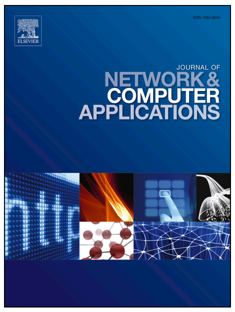
Accepted Manuscript



Mobile app traffic flow feature extraction and selection for improving classification robustness

Zhen Liu, Ruoyu Wang, Nathalie Japkowicz, Yongming Cai, Deyu Tang, Xianfa Cai

PII: S1084-8045(18)30340-0

DOI: https://doi.org/10.1016/j.jnca.2018.10.018

Reference: YJNCA 2241

To appear in: *Journal of Network and Computer Applications*

Received Date: 12 February 2018

Revised Date: 13 July 2018

Accepted Date: 24 October 2018

Please cite this article as: Liu, Z., Wang, R., Japkowicz, N., Cai, Y., Tang, D., Cai, X., Mobile app traffic flow feature extraction and selection for improving classification robustness, *Journal of Network and Computer Applications* (2018), doi: https://doi.org/10.1016/j.jnca.2018.10.018.

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Mobile App Traffic Flow Feature Extraction and Selection for Improving Classification Robustness

Zhen Liu[abe], Ruoyu Wang[cd*], Nathalie, Japkowicz[b], Yongming Cai[ae], Deyu Tang[ae], Xianfa Cai[ae]

[a]School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China

[b]Department of Computer Science, American University, Washington, DC, 20016, USA

[c]Information and Network Engineering and Research Center, South China University of Technology, Guangzhou 510041, China

[d]Communication & Computer Network Lab of Guangdong, Guangzhou 510041, China

[e]Guangdong precise medicine and big data engineering technology research center for traditional Chinese medicine, 510006, China

Abstract: In machine-learning based mobile app traffic classification, flow feature distributions can easily drift due to changes in network environments, user habits etc. Unstable features may negatively influence mobile app traffic classification robustness, so to remedy this problem, this paper investigates how to obtain optimal feature sets for improving classification robustness of mobile app traffic. Specifically, we develop a method to search for stable and discriminative features by jointly analyzing mobile app traffic characteristics and assessing the degree of feature drift. Along these lines, we first analyze the in-flow behavior characteristics of traffic flows, so as to extract a potential feature set for mobile app traffic data. Next, we present two new metrics to assess the degree of drift experienced by the flow features from different perspectives and design a composite metric to score these features by considering the degree of drift as a penalty factor of discrimination power. Based on these metrics, we further propose an algorithm to search for optimal features with high discrimination power but low degree of drift. Existing flow features and feature selection algorithms were implemented for our comparison experiments. Our experimental results on real mobile app traffic data demonstrate the effectiveness of our feature set and feature selection algorithm on improving classification robustness.

Keywords: mobile app traffic classification, flow feature selection, classification robustness, machine learning

1 Introduction

In the past decade, a dramatic growth in mobile app traffic has been witnessed, which results mainly from an increase in the number of mobile devices and cellular data networks. According to the *Cisco Visual Networking Index Forecast*, mobile devices, worldwide, are expected to consume more than 48.3 exabytes per month by 2021. The ability to classify mobile app traffic has significant implications in many domains, including traffic management, bandwidth allocation, and maintenance of user privacy.

A variety of mobile app traffic classification methods [1][2] have been proposed in the past. Since a significant fraction of mobile apps utilize the HTTP (or HTTPS) protocol for communication, most of the literature focuses on the identification of HTTP traffic [1] [3][4]. These approaches typically inspect packet payload and analyze HTTP headers to extract fingerprints [3][4] and, therefore, cannot handle encrypted traffic. Machine learning based traffic classification techniques do not rely on packet payload. However, they also face significant challenges. 1) Multiple kinds of flow features have been proposed to characterize traffic data. Which kind of feature is better for mobile app traffic data, however, has not been investigated, and, to date, the different feature sets that have been derived for mobile app traffic classification have not been appropriately compared. 2) The distribution of some flow features may change with network environment, user habits, app versions, leading to unstable classification performance. How to improve mobile app traffic classification robustness, thus, remains an open problem.

To address these challenges, this paper aims at developing a method to obtain an optimal feature subset including discriminative and robust features. The main contributions of this paper are as follows.

(1) In order to choose a set of flow features for characterizing mobile app traffic, we first explore the in-flow behavior of popular apps with respect to packet size and inter arrival time pattern, and then extract a set of potential features

[*] E-mail addresses: rywang@scut.edu.cn (Ruoyu Wang)

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

(named Hybrid-Feature) based on the analysis of the results. This feature set is composed of common statistical features and newly presented in-flow features.

(2) To further select discriminative and stable features, a feature selection algorithm named DDFS (Discrimination power and degree of Drift evaluation based Feature Selection) is presented. We adopt existing metric, such as Gain Ratio, to evaluate its discrimination power, and present two new metrics to assess the degree of drift of each feature. The degree of drift, here, refers to the relative degree of change experienced by a feature. We further devise a composite metric by considering the degree of drift as a penalty factor of discrimination power. Using this metric, DDFS searches for optimal features with high discrimination power and low degree of drift based on the feature evaluation results.

(3) We carry out a variety of experiments to evaluate the performance of Hybrid-Feature and DDFS on real mobile app traffic data. These data were collected by our *mobilegt* system [8] and were labeled in both coarse-grained and fine-grained.

i. To assess the usefulness of our Hybrid-Feature set, multiple kinds of feature sets (Basic-Feature [3], Moore-Feature [9], Joint-Feature [10], Service-Feature [11], Graph-Feature [2]) were also extracted from our mobile app traffic data, and compared to Hybrid-feature set. This is the first work to conduct comparison experiments of different kinds of feature sets in the field of mobile app traffic classification. Our classification results show that Basic-Feature, Service-Feature and Hybrid-Feature outperform all the other feature sets. Moreover, the performance of the Hybrid-Feature set was shown to be more stable than that of the others.

ii. To evaluate the performance of DDFS, it was compared against existing feature selection algorithms, including Information Gain[12], Gain Ratio[12], Relief[13], Chi-Square[14], GOA(Global Optimization Approach)[15] and SRSF(Selects the Robust and Stable Features)[16],on our mobile app traffic data. Our experimental results show that DDFS is able to improve the traffic classification robustness. Further, the experiments on the WLAN traffic data demonstrate the generalization capacity of DDFS.

The remainder of this paper is organized as follows. Section 2 overviews related works on mobile app traffic classification methods, flow features and feature selection algorithms. Section 3 analyzes in-flow behavior characteristics of the mobile app traffic and extracts flow features (named Hybrid-Feature). Section 4 presents metrics for assessing the degree of drift experienced by a feature and devises a feature selection algorithm named DDFS. Section 5 carries out a variety of experiments and discusses the experimental results. Section 6 concludes this paper.

2 Related work

2.1 Methods of mobile app traffic classification

Recently, mobile app traffic classification has attracted a lot of research. App identification methods mainly utilize the fields in HTTP header [3][4][17][18] [19], such as user-agent[17], host name, method (GET/PUT/GET)[3], URL[3], and URI[18] etc. However, these works only focus on HTTP traffic using DPI (deep packet inspection) methods and cannot handle encrypted traffic.

Alan and kaur [20] researched whether Android apps could be identified using only TCP/UDP headers by machine learning techniques. They carried out experiments on traffic flows characterized by packet size based features. They concluded that popular apps could be identified with 88% *accuracy* when training and testing data were collected from the same devices, but with 67% *accuracy* when testing data were from an unseen device with similar operating system/vendor. Mongkolluksamee et al. [2] combined packet size distribution and communication pattern based features for machine learning. Communication pattern based features are extracted from the connectivity graph built from the traffic in 3-minute. Experimental results on five popular mobile apps (Facebook, Line, Skype, YouTube and Web) show that their method obtains 0.95 in F-measure. Conti et al. [21] studied the identification of user actions without packet payload information. This method is based on the packet size with direction information, and DTW (Dynamic Time Warping) is used to measure the distance between different flows for clustering. The combination of agglomerative clustering and random forest is used to build a classification model. The experimental results show that it obtains more

ACCEPTED MANUSCRIPT

than 95% *recall/precision* on identifying user actions when the traffic app is known. Fu et al. [24] researched the in-App usages (such as text, picture etc.) classification method for messaging traffic. Their method first segments traffic flows into sessions with a number of dialog, and extracts flow features from packet length and time delay. Random Forest, Gradient Boosted Trees, Support Vector Machine, Naïve Bayes and KNN are used to build a classifier that identifies usage types. Before identifying the user actions of traffic flows, the message service apps of these traffic flows should be pre-identified. Recently, Aceto et al. [46] researched different kinds of fusion methods for multiple classification algorithms on mobile app traffic classification. Common features (such as overall statistics on packet size) were used on their mobile apps traffic data, and the results show that the fusion method can improve upon the performance of single models on identifying mobile apps.

The kinds of flow features, mobile traffic data types and classification methods in the above cited papers are summarized in Table 1 which shows, among other things, that the packet size distribution and the overall statistics on packet sizes are commonly used by existing works.

Table 1 summery of related work on mobile app traffic classification

Feature set	Mobile app in traffic data	Methods
packet size sequence with direction [20]	1595 apps	Gaussian NB, multinomial NB, Jaccard coefficient
packet size distribution, the graph based features set: the number of nodes in each column, the number of nodes with only one edge, the average degree, the maximum degree etc. [2]	Facebook, Line, Skype, YouTube, Web	Random forest
vector of packet size with direction; minimum, maximum, mean, median, absolute deviation, standard deviation, variance, skew, and kurtosis of packet size etc. [46]	QQ, SayHi, GooglePlay, eBay, HotSpot, PureVPN, QQReader, PaltalkScene, 6Rooms, etc.	multi-classification fusion approach
overall descriptive statistics, variances in backward and forward directions, percentages of packets with length in K equal-sized ranges etc. [24]	WeChat and Whatsapp	Random Forest, Gradient Boosted Trees, SVM, Naïve Bayes
packet size in sequence of a traffic flow [21]	Facebook, Gmail, Twitter, Tumblr, Dropbox, Google+, Evernote	agglomerative clustering and random forest

2.2 Statistical flow features for network traffic

Mobile app traffic is similar to traditional wired network traffic composed of IP packets. The existing traffic features extracted from wired network traffic could, therefore, also be used to characterize mobile app traffic. According to the measuring object, traffic features fall into the following three categories.

2.2.1 Packet header based features

Packet header based features are extracted from the headers of the frame layer, IP layer and transport layer [22] and include the packet size, frame size, capture length on the frame layer header, and source and destination ports on the transport layer header. Alshammari et al.[22] compared the performance of packet based features and flow based features on classifying encrypted traffic, and concluded that flow based features perform better. The packet header based features are seldom used for network traffic classification alone.

2.2.2 Flow based features

Statistical features of traffic flow are commonly used by machine learning based traffic classification techniques. And these features can be further categorized into single-flow features and multi-flow features [31] [5] [6].

In the case of single-flow features, the features are extracted from the IP packets of a flow. Mcgregor et al. [25] first extracted packet size, packet inter arrival time etc. on a flow. Moore et al. presented 248 flow features [9][23], including port number, flow length, the number of packets and duration etc. Some of these features are extracted from uni-directional flows, such as the total number of packets from client to server, the ACK packets from client to server etc. The traffic data described by these features were publically shared and widely used by other researchers [26][27]. Other kinds of flow features were also proposed, such as packet size distribution based features [10][28][30]. The most commonly used features are still the packet size statistics and inter arrival statistics based features [3][27][29][32].

ACCEPTED MANUSCRIPT

Recently, to identify services provided by apps, Fu et al. [24] presented dialog based features that included overall descriptive statistics (standard deviation, mean etc.), variances in backward and forward directions, percentages of packets with particular length in K equal-sized ranges etc.

In the case of multi-flow features, the features are extracted from the flows generated by an {IP, Port} peer [6] or a session [5]. Bermolen et al. [6] presented the {IP, Port} peer based features, including the number of peers that send K packets or bytes to a target peer. Lee et al. [5] aggregated all flows communicated between two hosts as a session, and extracted session based features, including session duration, number of flows, etc.

2.2.3 Connectivity graph based features

Connectivity graphs are built to exhibit communication behavior between hosts. The vertices of connectivity graphs fall into the following four categories: host [33], {IP, Port} peer [34], flow [35] and one element of *five tuple* {*source IP, destination IP, source Port, destination Port and Protocol*} [2]. The graph based features are utilized as traffic features, such as the number of vertices, the number of edges etc. Recently, five tuple based connectivity graphs were built for extracting features to classify mobile app traffic [2]. In such a connectivity graph, each node is an element of five tuples, and each edge denotes the association among the elements of a five tuple. The graph based feature set mainly includes five types of shape-related information: the number of nodes in each column, the number of nodes with only one edge, the average degree, the maximum degree etc. The authors combined the connectivity graph based features with packet size distribution feature. For the packets used for building a graph, the number of packets in the range of $(2^{k-1},2^{k}]$ (k=0,…,11) bytes is counted. And the TCP and UDP packets are individually counted. Hence, 24 features are extracted based on packet size distribution. Graph-Feature includes 59 (35+24) features in total.

2.3 Feature selection algorithms for traffic classification

Feature selection plays an important role in reducing the data dimension and removing irrelevant features, so as to improve the classification performance. Existing feature selection algorithms, such as Information Gain, Chi-Square, Gain Ratio etc. mainly focus on the discrimination power of each feature among classes. They ignore the influence of drift problems faced by mobile app traffic data. Recently, Zhang et al. [16] proposed the SRSF algorithm to select stable features. This algorithm firstly obtains initial feature subsets using the WSU_AUC algorithm on multiple training sets. And then, it further selects the features with high occurrence frequency, high mean of WSU_AUC and low variance of WSU_AUC among these selected subsets. Fahad et al.[15] presented an algorithm named GOA to select stable features by combining multiple feature evaluation metrics, such as Information Gain, Gain Ratio, Chi-square etc. Each metric is used to obtain an initial feature subset. They further search for the features with high outstanding occurrence frequency among these feature subsets. However, SRSF and GOA do not directly evaluate the degree of drift experienced by the different features. This paper proposes two metrics to quantify the degree of drift, which will be used to search for optimal features in mobile app traffic classification.

3 Mobile app traffic feature extraction

In this section, we first explore the characteristics of traffic flows, and then extract flow features based on the results of our exploration.

3.1 Mobile app traffic flow characteristics

A traffic flow is composed of the packets with the same *five tuple* during a certain period of time [9]. It is represented by two sequences: (1) a sequence of packet size and (2) a sequence of inter arrival time [24]. The packet size is measured in bytes, and the inter arrival time denotes the arrival time distance between two nearby packets in a flow. Fig. 1 represents these two sequences graphically. The x-axis represents the packet inter arrival time and the y-axis represents the packet size corresponding to the given inter arrival time in a flow. In each sub-figure, the lower side represents incoming traffic, while the upper side represents outgoing traffic. Each vertical line denotes a packet, and packets of the same color belong to the same flow.

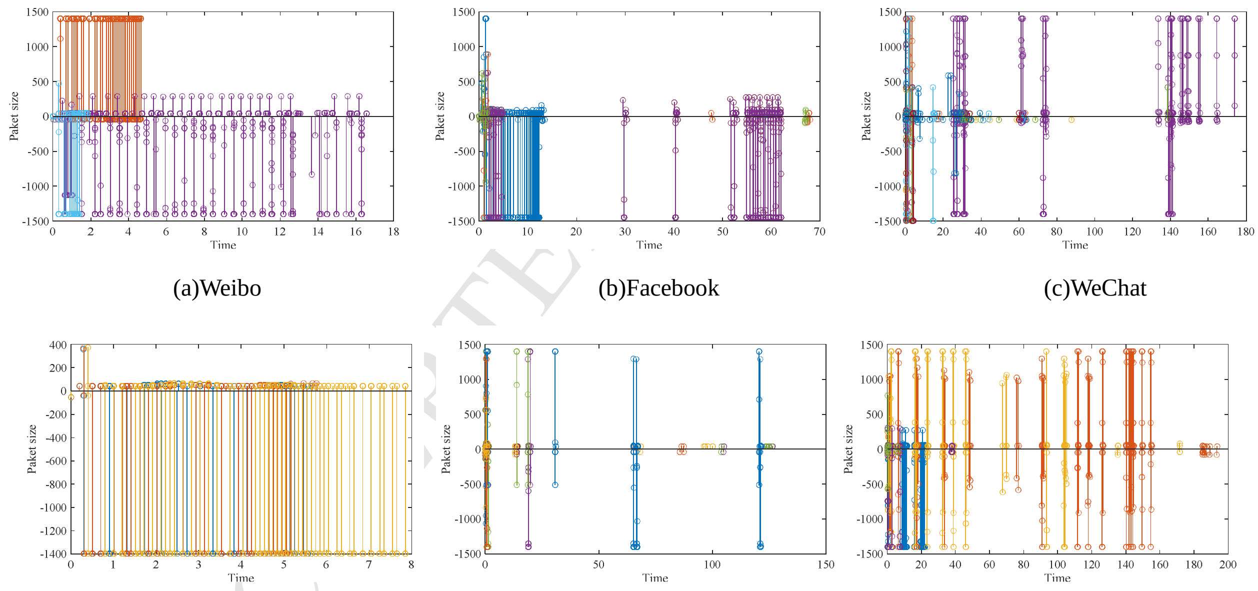
Fig.1 shows that different kinds of apps have different global characteristics (e.g., mean or standard deviation of packet

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

size etc.) and local characteristics (e.g., outgoing or incoming packet sizes, packet size distance between nearby packets etc.). For example, the flows in Fig.1(a) exhibit two kinds of in-flow behavior (or local characteristics): (1) outgoing packet sizes are much greater than incoming packet sizes (orange lines) in some flows, during which the user may upload pictures or a short video; (2) incoming packet sizes are also greater than outgoing packet sizes (purple lines) in some flows, during which the user may view pictures or short videos. In Fig.1(b), Facebook is also a popular social app, whose flows are similar to Weibo's flows in terms of packet sizes. Fig.1(c) exhibits different sizes among packets in a flow, which suggests that WeChat is used to chat with others by text, audio and video. Moreover, the packets exhibit some clusters by inter arrival time, because the inter arrival time between some nearby packets are much larger than others. The inter arrival time in each cluster is very small (<0.05ms). The inter arrival time between clusters are not constant, because the inter arrival time is influenced by the response time of remote devices or users. This indicates that the inter arrival time easily changes with network environment and user habits.

Streaming apps such as Youku in Fig.1(d) generate packets of maximum size and small inter arrival time when users watch videos. In general, the packets are of full size to accelerate the speed of videos. The Browser is mostly used for searching. Fig.1(e) shows that the incoming packets are of different sizes, as users obtain different search results such as text, pictures and videos. According to further analysis, we found that some continuous packets are of similar size, such as downloading a video in a short period of time. The inter arrival time between clusters is also dynamic and depends on the response time of the server/user. Mobile shopping represents another kind of apps. Users mostly search for goods or view pictures and introduction of goods. Fig.1(f) shows that for these apps, packets come in a variety of sizes, and continuous packets are all of the same size to accelerate the loading of pictures.



(a)Weibo (b)Facebook (c)WeChat

(d) Youku (e)Browser (f) VipShop

Fig.1 Packet size sequences with inter arrival time of mobile app traffic flows

3.2 Mobile app traffic flow features

To perform network traffic classification using machine learning, flow features are required to be extracted in order to build the flow samples. A flow sample is generally represented by a flow feature vector. According to the in-flow behavior analysis of mobile app traffic in Section 3.1, we now introduce the flow features extracted from four perspectives: (1) in-flow features, (2) packet header features (3) packet size and inter arrival time distribution features and (4) overall statistics.

To clearly describe the flow features, we first give the definitions of full flow, incoming flow, outgoing flow and subflow. Given a flow, the two peers of the flow are denoted by {IP_1, $Port_1$} and {IP_2, $Port_2$} respectively and the

ACCEPTED MANUSCRIPT

transport protocol is denoted by *pro*.

Definition 1 full flow: a full flow is composed of bi-directional packets coming from the same five tuple structure. These packets are denoted by {*Pkt*| (srcIP(*Pkt*)=IP$_1$& dstIP(*Pkt*)= IP$_2$ & srcPort(*Pkt*) = Port$_1$ & dstPort (*Pkt*) = Port$_2$) || (dstIP(*Pkt*)=IP$_1$& srcIP(*Pkt*)= IP$_2$ & dstPort(*Pkt*) = Port$_1$ & srcPort(*Pkt*) = Port$_2$) & Protocol(*Pkt*) = *pro*}.

Definition 2 incoming flow: the incoming flow is composed of the packets from the in-direction flow. These packets are denoted by {*Pkt*| dstIP(*Pkt*)=IP$_1$& srcIP(*Pkt*)= IP$_2$ & dstPort(*Pkt*) = Port$_1$ & srcPort(*Pkt*) = Port$_2$ & Protocol(*Pkt*) = *pro*}.

Definition 3 outgoing flow: the outgoing flow is composed of the packets from the out-direction flow. These packets are denoted by {*Pkt*| srcIP(*Pkt*)=IP$_1$& dstIP(*Pkt*)= IP$_2$ & srcPort(*Pkt*) = Port$_1$ & dstPort(*Pkt*) = Port$_2$ & Protocol(*Pkt*) = *pro*}.

Definition 4 subflow: a subflow is composed of a subset of packets found within a single direction of the flow. An example of a subflow is shown in Fig. 2.

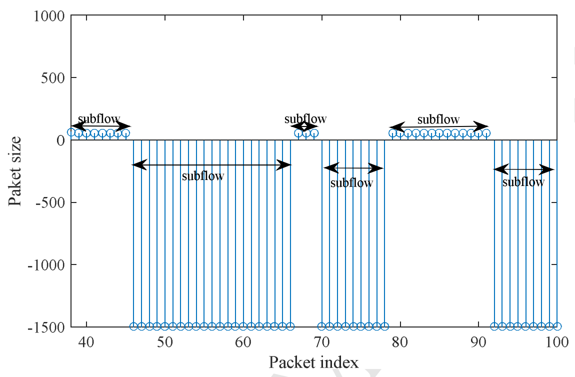


Fig.2 An example of subflows

(1) In-flow features

The in-flow features are shown in Table 2. The iIPS (incoming inter-packet size) and oIPS (outgoing inter-packet size) refer to inter-packet size with direction. Given a packet size sequence {z_1, z_2, z_3,…, z_n}, the inter-packet size sequence is {|z_1-z_2|,|z_2-z_3|,…,|z_{n-1}-z_n|}. The basic statistical calculations (min, max, mean, standard deviation) are performed on the inter-packet size sequence. For example, Streaming apps always generate packets of the same size (i.e., the maximum size). The inter-packet sizes are close to 0. Browser apps generate packets of various sizes but some nearby packets may be of similar sizes during data transmission. That is, some inter-packet sizes are close to 0 but others may be much larger than 0.

The SFPS (subflow packet size) and SFIAT(subflow inter arrival time) refer to subflow related features, i.e. statistical calculation of the flow size and inter arrival time of subflows. Each subflow contains multiple packets. The mean of packet size (*mps*) and the mean of inter arrival time (*miat*) are first calculated on each subflow, and the sequences {*mps*$_1$, *mps*$_2$,…,*mps*$_k$} and {*miat*$_1$,*miat*$_2$,…,*miat*$_k$} are built. From these sequences, the min, max, mean and standard deviations calculations are performed on the two sequences. The subflows of Streaming apps contain a lot of incoming bytes but a small number of outgoing bytes. The subflows of Social apps may generate a lot of incoming and outgoing bytes when viewing and posting pictures or short videos. The Browser subflows may generate incoming bytes of varying-size when searching for information. The inter arrival times in subflows is large on Browser traffic, but small on Streaming traffic. The subflow related features aim to describe the statistical information in each round of data transmission.

Table 2 In-flow features

Flow features	Description
iIPS(min, max, mean, standard deviation)	The minimum, maximum, mean, standard deviation of inter packet size distance of in-coming packets, i.e. the size distance between nearby packets on incoming flow
oIPS(min, max, mean, standard deviation)	The minimum, maximum, mean, standard deviation of inter packet size distance of out-going packets, i.e. the size distance between nearby packets on outgoing flow

ACCEPTED MANUSCRIPT

SFPS(min, max, mean, standard deviation)	The minimum, maximum, mean, standard deviation on the sequence {$mps_1, mps_2, .., mps_k$}, where mps_i denotes the mean of packet sizes in the *i*th subflow.
SFIAT(min, max, mean, standard deviation)	The minimum, maximum, mean, standard deviation on the sequence {$miat_1, miat_2, .., miat_k$}, where $miat_i$ denotes the mean of inter arrival time in the *i*th subflow.

(2) Packet header feature

According to our previous research [36], the port number is still effective when used with other statistical features for traffic classification. Inspired by the service IP and app correlation based identification method [37], this paper extracts the destination IP, destination port, source port and protocol in the packet header as flow features.

(3) Packet size and inter arrival time distribution related features

The distribution of packet size and inter arrival time in a full flow could be described as below. All features are calculated for the full flow.

The amount of packets in different ranges of sizes (PS-x): packets are counted in different ranges of sizes (measured in bytes). The packet size is generally in the range of 40-1500. Therefore, 6 ranges are used here, i.e. $[2^5,2^6)$, $[2^6,2^7)$,…,$[2^{10},2^{11})$.

The distribution of packet size along with inter arrival time (PSI-x), i.e. the number of packets in different ranges of inter arrival time. We used 9 ranges for inter arrival time (in microseconds), i.e. $[0,10)$, $[10,10^2)$,…,$[10^8,+\infty)$. The inter arrival time of Streaming traffic is small, i.e. most packets are distributed in the range of small inter arrival time. Most packets in Browser apps are distributed in the range of large inter arrival time.

Traffic ratio in different directions (IOPR(incoming and outgoing packets ratio) and IOBR(incoming and outgoing bytes ratio)): IOPR is computed by the ratio between the incoming #packets and outgoing #packets; IOBR is computed by the ratio between the incoming bytes and outgoing bytes e.g. outgoing traffic is greater than incoming traffic (IOBR<1) when users post pictures/videos using social apps.

Packet size frequency (FHPS(first highest packet size), SHPS (second highest packet size), THPS(third highest packet size)): the packet sizes with top three frequencies. For example, streaming apps generate a lot of max-size packets, leading to a value of FHPS larger than 1400 but values of SHPS and THPS much smaller than FHPS.

(4) Overall statistical flow features

The overall statistical flow features are obtained by performing the general statistical calculation on the packets of incoming or outgoing flows. All of them are shown in Table 3.The first 20 features are used as the Basic-Feature in our experiments of Section 5, which includes the Packet size statistics and inter arrival time statistics. They are frequently used in existing works [3][27].

Table 3 overall statistical flow features

Flow features	Description
oFS	outgoing flow size
iFS	incoming flow size
oNP	The number of outgoing packets
iNP	The number of incoming packets
oPS(mean, min, max, standard deviation)	The minimum, maximum, mean and standard deviation of packet size in outgoing flow
iPS(mean, min, max, standard deviation)	The minimum, maximum, mean and standard deviation of packet size in incoming flow
oIAT(min, max, mean, standard deviation)	The minimum, maximum, mean and standard deviation of inter arrival time in outgoing flow
iIAT(min, max, mean, standard deviation)	The minimum, maximum, mean and standard deviation of inter arrival time in incoming flow
iFD	The flow duration of incoming flow
oFD	The flow duration of outgoing flow
iSkewness	The skewness of incoming flow
oSkewness	The skewness on outgoing flow
ikurtosis	The kurtosis in incoming flow
okurtosis	The kurtosis in outgoing flow
iSE	The standard error of packet size in incoming flow
oSE	The standard error of packet size in outgoing flow

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

4 Feature selection algorithm

The Hybrid-Feature could be used to characterize mobile app traffic. However, some features may easily drift with changes in network environments, user habits, app versions etc. This section presents a filter feature selection algorithm to search for discriminative and stable features for mobile app traffic data. Let $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2),\ldots,(\mathbf{x}_N, y_N)\}$ denote a dataset. Each flow sample $\mathbf{x}_i$ is characterized by a feature set $F = \{f_1, f_2,\ldots, f_m\}$. The $y_i \in C=\{C_1, C_2,\ldots,C_q\}$ denotes the class label of $\mathbf{x}_i$ and it is from predefined q classes.

4.1 Feature evaluation metrics

According to the target of our feature selection algorithm, features have to be measured from two aspects: discrimination power and stability. In the case of discrimination power, a feature could be evaluated by a traditional metric such as Gain Ratio etc. In the case of stability, two new metrics are presented to assess the degree of drift of a feature. The degree of drift is defined as the relative degree of change experienced by a feature. Stability could be evaluated from two perspectives: feature value distribution in each class (local perspective) and overall feature value distribution (global perspective).

4.1.1 Degree of drift metric from a local perspective

This kind of metric is supervised. It is calculated on the labeled training data. On the training data, we pay more attention to the stability of a feature with respect to characterizing each class, i.e. the feature value uncertainty on each class. If the most frequent value of a feature is changed from v_1 on dataset1 to v_2 on dataset2, but the uncertainty of the feature value is still low, then the feature is still effective, as all values of the feature could be learned by the classification model. This suggests that the stability of uncertainty is more important than the stability of values when the observed data can also be learned by the classification model. Among multiple labeled datasets (the subset of training set), we hope to find the variation of uncertainty of a feature on each class**,** so as to expose the change in the stability of the feature on each class.

Information entropy essentially quantifies “the amount of uncertainty” contained in the data [40]. The empirical entropy of a random variable X on observed samples is defined as

$$H(X) = -\sum_{x_i \in X} p(x_i)\log p(x_i) \qquad (1)$$

where x_i is the *i-th* value of X, and $p(x_i)$ is the empirical probability of X taking the value of x_i on observed data.

On the observed samples, $0 \le H(X) \le H_{\max}(X)=\log_{\min}\{N_x, M\}$. N_x is the number of values taken by X, and M is the number of observed samples. $H_{\max}(X)$ is the boundary of $H(X)$ and is referred to as the maximum entropy [40]. Based on $H_{max}(X)$, the relative uncertainty is defined as Eq.(2). It puts $H(X)$ in a scale of 0-1, and provides an index of variety regardless of N_x and M, so as to be used to compare the uncertainty of different variables.

$$RU(X) = \frac{H(X)}{H_{\max}(X)} = \frac{H(X)}{\log_2 \min\{N_x, M\}} \qquad (2)$$

$RU(X)$ is in the range of [0,1]. $RU(X)$=0 implies that the observational randomness is completely absent, i.e. all observations of X are the same. $RU(X)$=1 implies that the observational randomness is the highest, i.e. the probability that all values of X are the same.

When RU is used to evaluate the uncertainty of a feature variable f_i (i=1,..,m), the X in Eq. (2) corresponds to the feature f_i, N_x is the number of values taken by f_i and M is the number of observed flow samples. Then the conditional relative uncertainty $RU(f_i|C_j)$ measures the randomness of f_i on the observed data from C_j (j=1,..,q). $RU(f_i|C_j)$ represents the ability of f_i on identifying C_j. When $RU(f_i|C_j)$=0, f_i only uses one value to characterize observed data from C_j, which suggests that f_i can easily identify the traffic of C_j using this value. When $RU(f_i|C_j) = 1$, f_i owns all kinds of values on observed data from C_j, which suggests that f_i has the largest randomness and can not identify the flow samples from C_j well. If the RU of a feature has large variance among multiple labeled data sets, the feature is not stable on characterizing traffic classes. Given multiple labeled data sets $\{D_1, D_2,..,D_K\}$, the degree of drift on $RU(f_i|C_j)$ (*DDRU* in short) is defined

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

as

$$DDRU(f_i \mid C_j) = Std(RU_1, RU_2, ..., RU_K) \qquad (3)$$

It is the standard deviation of *RUs* obtained on multiple labeled datasets. *The* $RU_1,\ldots,RU_K$ are the $RU(f_i|C_j)$ on $D_1, D_2, \ldots, D_K$ datasets, respectively. The smaller the *DDRU* of a feature, the more stable that feature on characterizing each class.

4.1.2 Degree of drift metric from a global perspective

This kind of metric is unsupervised. It evaluates the degree of drift of the feature value distribution, which does not need label information. Therefore, the unknown data that will be classified could be used for this calculation. The change in feature value distribution may degrade classification performance as these values have not been seen by the model. When classifying future unknown traffic data, the drift of each feature on the unknown data is important for searching for stable features and improving the mobile app traffic classification robustness.

To evaluate the degree of drift of the feature value distribution, the values of each feature are first discretized into multiple ranges $\{r_1, r_2, \ldots, r_k\}$, and the empirical probability of each range is assessed on observed flow samples. The probability of the *j-th* range r_j of a feature is defined as

$$p(r_j) = \frac{N_{r_j}}{N} \qquad (4)$$

where N_{rj} is the number of flow samples whose feature value falls into the range r_j, and N is the total number of flow samples on a given data.

Given the probability sequence on the training set $(p(r_1),.., p(r_k))$ and unknown dataset $(p'(r_1),.., p'(r_k))$, the distance between the two sequences can be evaluated by distance metrics, such as Hellinger, Euclidean distance etc. The two sequences represent feature value probability distributions. The Hellinger distance is a measure of the similarity between two probability distributions [41]. In addition, it is symmetric and in the scale of 0-1, so as to be used for comparing the probability similarity of difference features. Given two probability sequences $P=(p(r_1),.., p(r_k))$ and $P'=(p'(r_1),.., p'(r_k))$ of a feature f_i, the definition of the Hellinger distance of f_i is

$$HD_i(P, P') = \sqrt{\frac{\sum_{j=1}^{m} (\sqrt{p(r_j)} - \sqrt{p'(r_j)})^2}{2}} \qquad (5)$$

$HD_i(P,P')$ is in the range of [0,1]. $HD_i(P,P') = 0$ means that the distribution of feature f_i on unknown data is the same as that on the training data. $HD_i(P,P') = 1$ means that the distribution of feature f_i on training data is much different from unknown data. Therefore, the smaller the *HD* is, the feature is more stable on feature value distribution.

4.1.3 Composite metric

To search for discriminative and stable features, the degree of drift metric is used as a penalty factor of the discrimination metric. Based on the *exp* function, the composite metric is defined as

$$com(f_i) = \frac{dis(f_i)}{(\sqrt{e^{drift(f_i)}})^{\gamma}} \qquad (6)$$

Where $dis(f_i)$ is a discrimination power metric, such as GainRatio etc., and the $drift(f_i)$ is the degree of drift metric. The *dis* and *drift* are normalized by MIN-MAX way before calculating $com(f_i)$ if the value of the metric is not in the range of [0,1]. γ is used to control the power of *drift*. There are multiple ways to combine the *dis* and *drift* metrics. In this paper, the *exp* function is used to avoid zero value in the denominator and to enlarge the value of *drift*, because empirical results show that the value of *drift* may be too small, leading to the power of *dis* becoming weak and the algorithm tending towards selecting features with low degrees of drift but also with low discrimination power. It is an empirical way to combine the two metrics, and may not be the optimal one. As r is used to control the power of drift, the *exp* function could also be replaced by other exponential functions. Improving the effectiveness of combining the *dis* and *drift* metrics is left as interesting future work.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

4.2 Feature selection algorithm

4.2.1 Preliminary

In mobile app traffic classification, the features with high discrimination power play a key role in identifying different classes. The features with outstanding discrimination power are expected to be selected without considering the result of the composite metric. Also, with respect to the features with an outstanding degree of drift, this kind of feature is not stable and should be removed. Therefore, DDFS is based on the following rules. Rule 1 has higher priority than rule 2.

Rule 1: For the features with outstanding discrimination power, they perform well on traffic classification. These features should be selected for sure.

Rule 2: For the features with an outstanding degree of drift, they are not stable and would negatively influence traffic classification robustness. These features should be removed.

Therefore, we divide the features into three categories according to discrimination power and degree of drift.

Definition 5 Safety feature: a feature f_i is a safe feature if it has outstanding discrimination power, i.e. $dis(f_i) \in [\eta_{dis}, max_{dis}]$, where η_{dis} is the outstanding threshold of *dis*. Such features should be selected.

Definition 6 Danger feature: a feature f_i is a dangerous feature if it suffers from outstanding drift, i.e. $DDRU(f_i) \in [\eta_{ddru}, max_{ddru}]$ or $HD(f_i) \in [\eta_{hd}, max_{hd}]$, where η_{ddru} and η_{hd} are the outstanding threshold of *DDRU* and *HD* respectively. Such features should be removed.

Definition 7 Pending feature: a feature f_i is pending if it does not have outstanding discrimination power and does not suffer from outstanding drift. Such features will be further checked by the composite metric.

4.2.2 DDFS algorithm

Based on the feature evaluation metrics and the definitions of the three categories of features, our feature selection algorithm named DDFS (Discrimination power and degree of Drift based Feature Selection) is shown in Algorithm 1. *S* denotes the labeled training set, *T* denotes the unlabeled data set that will be classified, parameter β is the input parameter of **OutstandingThreshouldSearch** and parameter N_f represents the number of features that will be selected by DDFS. The two parameters will be further discussed in Section 5.5. The DDFS algorithm can be divided into three parts. The first part (lines 1-2) evaluates the discrimination power and the degree of drift of each feature. The degree of drift could be evaluated by *DDRU* and *HD*. The second part (lines 3-14) searches for the outstanding threshold for *dis[]* and *drift[]*, and selects the safety features and danger features according to their definitions. The third part (lines 15-26) computes the composite metric value based on *dis[]* and *drift[]*. Then it first selects the features in S_{safety}, and then selects the features with higher composite metric value and not in S_{danger}, until the number of selected features equals to N_f.

In Algorithm 1, the **Eval*dis*** function evaluates the discrimination power of each feature. Eval_{dis} could be solved by performing Gain Ratio. The **Eval** function evaluates the degree of drift of a feature and will be detailed in Section 4.2.3. The **OutstandingThresholdSearch** function searches for the threshold used in identifying the safety and danger features and will be detailed in Section 4.2.4.

Algorithm 1 Feature selection algorithm

Input: Training set *S*, Testing set *T*, β, N_f

Output: an optimal feature subset S_{best}

```
1 $dis[]=\mathbf{Eval}_{dis}(S)$;
2 $drift[]=\mathbf{Eval}_{\mathbf{drift}}(S,T)$;
3 $\eta_{dis}$ = OutstandingThreshouldSearch($dis[], \beta$);
4 for each feature $f_i$
5    if $dis[i] \geq \eta_{dis}$
6        $S_{safety} = S_{safety} \cup \{f_i\}$;
7    end if
8 end for
```

ACCEPTED MANUSCRIPT

```
9 $\eta_d$ = OutstandingThreshouldSearch(drift[], $\beta$);
10 for each $f_i$
11    if drift[i] $\geq \eta_d$
12       $S_{danger} = S_{danger} \cup \{f_i\}$;
13    end if
14 end for
15 for each feature $f_i$
16    $com[i] = \frac{dis[i]}{(\sqrt{e^{drift[i]}})^{\gamma}}$;
17 end for
18    $S_{best} = S_{best} \cup S_{safety}$;
19 while($|S_{best}| < N_f$)
20    $f_p$=firstFeature($S_{sort}$);       //in $S_{sort}$, features are sorted in descending order according to com[]
21      if $f_p \notin S_{best}$ && $f_p \notin S_{danger}$
22         $S_{best} = S_{best} \cup \{f_p\}$;
23         $S_{sort} = S_{sort} \setminus \{ f_p\}$;
24      end if
25 end while
26 return $S_{best}$;
```

4.2.3 Degree of drift evaluation process

The degree of drift can be evaluated by *DDRU* and *HD*. They are solved, respectively, by Algorithms 2 and 3. In Algorithm 2, in both *S* and *T*, the values of each feature is normalized by the min-max procedure and are discretized into multiple ranges. Algorithm 2 is a loop. For each feature, it first counts the empirical probability of each value range, and then evaluates the feature value distribution distance using Eq.(5).

Algorithm 2 Eval$_{\text{drift}}$(*S*,*T*) by *HD*

Input: Training set *S*, Testing set *T*

Output: degree of drift for each feature *drift[]*

```
1 for each feature $f_i$
2      $Prob_s$=ProbofEachRange($f_i$ |$S$);
3      $Prob_t$= ProbofEachRange($f_i$ |$T$);
4      drift[i]= HD($Prob_s$ , $Prob_t$);
5 end for
6 return drift[];
```

Algorithm 3 divides the degree of drift evaluation into two main parts. The first part (lines 1-7) calculates the *RU*(f_i) on each dataset. The second part (lines 8-14) first calculates *DDRU* using Eq.(3) and then calculates the average of *DDRU* for each feature over the results obtained on different classes.

Algorithm 3 Eval$_{\text{drift}}$(*S*,*T*) by *DDRU*

Input: Training sub sets S_1 ,S_2,..,S_K

Output: degree of drift for each feature *drift[]*

```
1 for each subset $S_k$
2    for each feature $f_i$
3       for each class $C_j$
4           r[i][j][k]=RU($f_i$|$C_j$);       //RU is acquired on the data of $C_j$ from $S_k$
5      end for
```

ACCEPTED MANUSCRIPT

```
6    end for
7  end for
8  for each feature $f_i$
9      for each class $C_j$
10         $ddru[i][j]= std(r[i][j])$;     //calculate the standard deviation on $r[i][j]$
11     end for
12     $drift[i]=Mean(ddru[i])$;     //calculate the average over $ddru[i]$
13 end for
14 return $drift$;
```

4.2.4 Outstanding threshold search

The outstanding threshold is used for selecting the safety and danger feature sets. The threshold is calculated based on *SSE*(sum of squared error). The *SSE* has been used to search for the K parameter in K-Means [40], the search loop for *K* stops when the *SSE* is smaller than a threshold (i.e., a small value β). Inspired from this, *SSE* is applied here to search for the outstanding threshold. Similarly, the loop stops when the *SSE* stops showing a significant decrease. Given an array of evaluation metric values $\{v_1, v_2, .., v_m\}$, *SSE* is defined as

$$SSE = \sum_i (v_i - \bar{v})^2 \tag{7}$$

The value range of *SSE* is $[0, +\infty)$. The search algorithm for the outstanding threshold is shown in Algorithm 4.The goal is to find the threshold to identify the outstanding values. The algorithm is divided into three main parts. The first part (line 1) calculates the *SSE* on the initial *val* array (denoted by φ_i). The second part (lines 2-9) sorts the *val* array in descending order and moves the first two elements from V_{sort} to a new array V_{large}. V_{small} is composed of the elements of V_{sort} after removing the largest two elements. The *SSE* is calculated on V_{large} and V_{small}, and their results are respectively denoted by φ_l and φ_s. The initial *SSE* φ_i subtracts the sum of φ_l and φ_s, and the result is checked if it is smaller than β. If it is not, the algorithm implements the third part (lines10-19). The next largest element in V_{small} will be repeatedly removed from V_{small} and added to the end of V_{large}, and the *SSE* is calculated on these two arrays, until the decrease in *SSE* is smaller than β. The threshold is the last value in V_{large}.

Algorithm 4 OutstandingThresholdSearch

Input: metric evaluation array *val*[], β

Output: best threshold η

```
1  $\varphi_i = SSE(val)$;
2  $V_{sort} = Sort(val)$;
3  $V_{large} = \{v_1, v_2\} = firstTwoValues(V_{sort})$;
4  $V_{small} = V_{sort} \setminus \{ V_{large} \}$;
5  $\varphi_l = SSE(V_{large})$;
6  $\varphi_s = SSE(V_{small})$;
7    if($\varphi_i - \varphi_l - \varphi_s \leq \beta$)
8      return $null$;
9    end if
10 while ($\varphi_i - \varphi_l - \varphi_s > \beta$)
11     $\varphi_{I} = \varphi_l + \varphi_s$;
12     $\eta = lastValue(V_{large})$;
13     $v = firstValue(V_{small})$;
14     $V_{large} = V_{large} \cup \{v\}$;
15     $V_{small} = V_{small} \setminus \{v\}$;
```

ACCEPTED MANUSCRIPT

```
16    $\varphi_l = SSE(V_{large})$;
17    $\varphi_s = SSE(V_{small})$;
18 end while
19 return $\eta$;
```

4.2.5 Parameter γ solution

There is a balance problem between the discrimination power and degree of drift in Eq.(6) and the balance is adjusted by parameter γ. If the degree of drift punishes the discrimination power too much, the features with high discrimination power may be wrongly removed. In order to guarantee the rank of safety features, Algorithm 5 handles the balance problem by searching for parameter γ. The smaller γ is, the lesser the punishment for degree of drift. γ is initialized as 1, and it is decreased until the index of every safety feature is at least smaller than 2*$|S_{safety}|$ in the sorted feature sequence in descending order according to *com[]*. This means that the safety features could be in the top area of the rank list with the final γ value. In algorithm 5, $|S_{Safety}|$ is the number of safety features that are searched for by the outstanding threshold solved by Algorithm 4.

Algorithm 5 Search parameter γ

Input: S_{safety}, *dis[], drift[]*

Output: γ

```
1 while γ >0
2     for each feature $f_i$
3         $com[i] = \frac{dis[i]}{(\sqrt{e^{drift[i]}})^{\gamma}}$;
4     end for
5     $flg$=true;
6     $k$=0;
6     for each feature $f_i$ in $S_{safety}$
7       if the $Rank(f_i) > 2*|S_{safety}|$
8           $flg$=false;
9       end if
10      if($flg$)
11          break;
12      else
13        $k$++;
14        $\gamma=2^{-k}$;
15      end if
16    end for
17 end while
```

5 Experiments

This section first introduces the mobile traffic data used in our experiments and the classification performance evaluation metrics, and then discusses the performance of the Hybrid-Feature set and of the DDFS algorithm by comparing them against existing feature sets and feature selection algorithms respectively.

5.1 Experimental data and performance evaluation metrics

5.1.1 Experimental data

Our mobile traffic data were collected through deploying the *mgtClient* [8] on 10 volunteers' smartphones and the *mgtServer* on a remote server during the October 2016 -March, 2017 period. Volunteers launched *mgtClient* when they chose to share their data. During data collection, they used popular apps as usual, such as WeChat, Weibo etc. The session

ACCEPTED MANUSCRIPT

information was collected on the volunteer's smartphone while running *mgtClient* and the raw traffic was captured at the remote server. Each item of the session information records the association between a *five tuple* and an app name so that, the apps of collected mobile traffic can be labeled by *mgtServer* with the aid of the session information. As a whole, the dataset is made up of 4050 traffic traces, and each trace covers about 16mins on average. The traffic on the most popular apps shown in Table 4 were selected as our experimental data.

Existing works show that network traffic can be labeled at the category level (Web, Streaming, Social etc.)[5], application level (WeChat, Facebook, Youku etc.)[6], and protocol level (HTTP, HTTPs etc.)[7]. Existing research only focuses on traffic classification performance at one level. To validate the effect of feature selection on the mobile app traffic data labeled at different levels, our mobile app traffic data were labeled at the coarse grained (category) and fine-grained (app) levels. Since most mobile apps utilize HTTP/HTTPS as their application layer protocol, we were mainly concerned with the app at the fine-grained level. Mobile apps were grouped into the following 5 categories in order to carry out the experiments at the coarse-grained level.

Table 4 The relationship between mobile app traffic category and mobile apps

Traffic category	Apps
Social	QQ, WeChat, Facebook, Weibo
Streaming	Youku, Tencent, MgTV
Web	Browser
Shopping	JdShop, VipShop
Mail	QQMail, YahooMail

The flow and byte distributions of each app category are shown in Table 5. Web and Social apps have many flows, as the popular usages of Browser, Weibo, WeChat etc shows. With respect to byte distribution, Social and Streaming apps account for more bytes than others. The details of the distributions of mobile apps are shown in Table 6. The number of flow samples among classes is imbalanced on both coarse-grained and fine-grained traffic data. Mobile app traffic classification may, therefore, suffer from class imbalance problem [38]. This paper mainly focuses on robust and discriminative feature selection. The class imbalance problem will be further studied as a future work.

Table 5 The number of flows, packets and bytes of each mobile app traffic category

Category	#Flows	#Packets	#Bytes
Social	57 K	10,548K	7.90G
Streaming	21 K	6,948 K	4.95G
Web	25 K	2,068 K	1.30G
Shopping	8 K	826 K	0.48G
Mail	4K	169 K	0.07G

Table 6 The number of flows, packets and bytes of each mobile app

Category	Classes	#Flows	#Packets	#Bytes
Social	QQ	17,104	3,213K	2.30GB
	WeChat	13,631	807K	0.57GB
	Facebook	1,400	440K	0.29GB
	Weibo	25,407	6,086K	4.75GB
Streaming	Youku	5,825	1,544K	1.09GB
	Tencentvideo	1,593	382K	0.29GB
	MgTV	14,046	5,021K	3.57GB
Web	Browser	25,512	2,068K	1.30GB
Shopping	Jdshop	4,008	323K	0.20GB
	Vipshop	4,577	503K	0.28GB
Mail	QQMail	1,432	25K	0.01GB
	YahooMail	3,485	143K	0.07GB

Another network traffic data set was collected at an edge router of a WLAN network of SCUT (South China University of Technology). Two traffic traces were captured at different time periods, respectively, 11:00 and 15:00 of January 8, 2016. Each trace covers about 10 mins. We obtained two datasets (named SCUT1 and SCUT2) through labeling the two traffic traces. We utilized nDPI [45] to build the ground truth for SCUT data. Carela-Español et al. [42] compared the *precision* of six tools (i.e., PACE, OpenDPI, L7-filter, nDPI, Libprotoident, and NBAR) for building

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

ground truth on network traffic. The results in [42] show that nDPI performs better than the other open-source tools [42]. To compare the traffic classification performance of our methods to others on the WLAN network, we grouped the protocols or applications into a higher level (category level) as shown in Table 7. The numbers of flows and bytes in each category of the SCUT data are shown in Table 8.

Table 7 The relation between protocols and traffic categories

Traffic category	nDPI labels
Social	SSL.QQ, HTTP.QQ, SSL.Facebook, HTTP. Sina(Weibo), SSL.Sina(Weibo), Skype
Streaming	HTTP.QuickTime, HTTP.Flash
Web	HTTP.Amazon, HTTP.Google, SSL.Google, MPEG, SOCKS, SSL.IFLIX, HTTP.IFIX, NETBIOS, Web, EAQ, SSL, HTTP.HTTP, HTTP_Proxy,
Download	Download, Thunder
Mail	SSL.Yahoo

Table 8 The number of flows and bytes of each category on SCUT data

Category	SCUT 1			SCUT 2		
	#Flows	#Packets	#Bytes	#Flows	#Packets	#Bytes
Social	1453	301K	200MB	1475	483K	298 MB
Streaming	987	1574 K	1282 MB	1551	1386 K	1072 MB
Web	2879	293 K	161 MB	3792	373 K	200 MB
Download	1175	1382 K	1108 MB	1291	1276 K	1047 MB
Mail	12	412	0.14 MB	6	166	0.04 MB

5.1.2 Performance evaluation metric

Mobile app traffic classification performance will be evaluated from *flow accuracy*, *byte accuracy*, *flow g-mean, byte g-mean, flow f-measure*, and *byte f-measure*. The flow can also be counted in bytes, hence the performance metrics in terms of bytes are also of concern in this paper.

Flow (or *byte*) *accuracy* is calculated as the ratio between the number of correctly classified *flows* (or *bytes*) and the number of classified *flows* (or *bytes*) on a flow sample set.

*Flow(*or *byte) g-mean* is generally used to measure the classification performance in class imbalance situation. It is the geometry mean of *flow* (or *byte*) *recalls* measured separately on each class. It is defined as Eq.(8), where R_i (i=1,2,…,q) is the *recall* of *i-th* class and q is the number of classes on a flow sample set.

$$g-mean=(\prod_{i=1}^{q} R_i)^{1/q} \qquad (8)$$

Flow (or *byte*) *f-measure* of a class C_i is the composition metric of *flow* (or *byte*) *recall* and *flow* (or *byte*) *precision*, and it is calculated as $2*P_i*R_i/(P_i+R_i)$. The definitions of *recall* and *precision* are respectively as below.

Flow (or *byte*) *recall* of a class C_i is calculated as the ratio between the correctly classified *flows* (or *bytes*) in C_i and the total *flows* (or *bytes*) of C_i on a flow sample dataset.

Flow (or *byte*) *precision* of a class C_i is calculated as the ratio between the correctly classified *flows* (or *bytes*) in C_i and the *flows* (or *bytes*) being classified as C_i on a flow sample set.

5.2 Comparison experiments on feature sets

In this section, Hybrid-Feature will be compared against five feature sets. These feature sets were extracted from different perspectives. Moreover, these features are clearly detailed in references and were easily re-built.

(1) Basic-Feature: the common 20 flow features [3][27] are used as the basic features for comparison.

(2) Moore-Feature: the fullstat [39], shared by Moore et al., was implemented on the raw traffic to build the flow samples characterized by 248 features.

(3) Graph-Feature: a connectivity graphlet is built over 3 minutes of traffic according to the suggestion in [2]. 59-dimension feature vector were extracted on mobile app traffic data.

(4) Joint-Feature: Based on the 9 bins (in bytes) for packet size: {2,5,10,100,200,500,1000,1400,∞}, and the 9 bins (in microseconds) for inter-packet time: {10,100,1000,10000,100000, 500000,1000000,10000000,∞}, the joint distributions on upload stream and download stream are respectively obtained. Therefore, 9*9*2 features can be

ACCEPTED MANUSCRIPT

extracted, and each feature represents a probability of packets which fall into the joint bins. In [10], the authors combined these features with the upstream packets divided by the sum of upstream and downstream packets, and the duration of the bi-flow. In this paper, the 164 features were extracted to characterize mobile app traffic data.

(5) Service-Feature: Fu et al. [24] recently presented a set of features for service usage classification in mobile messaging apps. These features are based on packet size and time delay. The details could be found in [24]. All of them were also extracted on our mobile app traffic.

Each feature set was used to characterize our mobile app traffic data and build a flow sample set. On the flow sample set, 10-fold cross validation was performed to compare the performance of different feature sets. In order to evaluate the robustness of each feature set, the classification model was trained on one dataset and tested on the dataset from unknown mobile devices or unknown environment. In the following, Random Forest [43] is used to train a classification model. It is implemented by the Weka toolkit [44]. The performance of all the feature sets will be evaluated on both coarse-grained data and fine-grained data. Among the results obtained by different feature sets, the best one is highlighted in bold, and the second best is highlighted in underline.

5.2.1 Cross validation results of the full feature set

(1) Results on the mobile app traffic data labeled in coarse-grained

The classification results on the coarse-grained data are shown in Table 9. This table shows that the Basic-Feature, Service-Feature and Hybrid-Feature outperform others. The classifiers trained with the three feature sets acquire more than 90% *flow accuracy* and *byte accuracy*. And the classifier trained with Hybrid-Feature performs the best. It obtains 94.6% *flow accuracy* and 97% *byte accuracy*. Moreover, its *flow g-mean* and *byte g-mean* are higher than 90%, whereas the others' g-means are lower than 90%. The Graph-Feature is based on the assumption that mobile users normally interact with an application around 10-250 seconds per session. But this assumption cannot be held when multiple apps run simultaneously on a mobile device. The Joint-Feature is mainly composed of the joint distribution of packet size and inter arrival time. However, the inter arrival time is easily influenced by the network environment and user habits. For example, the analysis in section 3.1 shows that the inter arrival time is closely correlated with the users' response time when using the interactive apps. The redundant and irrelevant features in Moore-Feature may lead to the worse classification performance on mobile app traffic data.

Table 9 Cross validation results of different feature sets on the data labeled in coarse-grained

Feature sets	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
Basic-Feature	0.90750.004	0.931±0.016	0.857±0.004	0.841±0.011
Service-Feature	<u>0.934±0.002</u>	<u>0.959±0.015</u>	<u>0.897±0.002</u>	<u>0.898±0.022</u>
Graph-Feature	0.778±0.050	0.781±0.065	0.566±0.046	0.539±0.055
Moore-Feature	0.813±0.006	0.913±0.038	0.735±0.008	0.823±0.039
Joint-Feature	0.852±0.002	0.918±0.010	0.775±0.007	0.812±0.028
Hybrid-Feature	**0.946±0.003**	**0.970±0.009**	**0.911±0.004**	**0.915±0.022**

(2) Results on the mobile app traffic data labeled in fine-grained

The classification results on the fine-grained data are shown in Table 10. The ranks of the 6 feature sets are similar to those in Table 9. Compared to the results in Table 9, the performance on fine-grained data in Table 10 is worse. This is because the classification among the apps is more difficult than that among the categories. The similar apps (such as the QQ, WeChat in social category) are easily misclassified as each other. This indicates that the classification object (i.e., the traffic class) would influence the performance of the classifier. The Basic-Feature, Service-Feature and Hybrid-Feature sets still perform better than the other three feature sets. And Hybrid-Feature performs the best, Service-Feature the second best, and following them, the Basic-Feature. The classifier with Hybrid-Feature still acquires >93% *flow* and *byte accuracies,* and >84% *flow* and *byte g-means*. Besides the features in Basic-Feature, Hybrid-Feature adds extra features based on in-flow behavior and packet headers information. The experimental results on the data in coarse-grained and fine-grained further prove that the extra features benefit the performance of Basic-Feature.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table 10 Cross validation results of different feature sets on the data labeled in fine-grained

Feature sets	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
Basic-Feature	0.885±0.003	0.910±0.034	0.837±0.006	0.774±0.065
Service-Feature	0.912±0.004	0.924±0.024	0.885±0.006	0.838±0.054
Graph-Feature	0.588±0.020	0.671±0.055	0.362±0.034	0.325±0.043
Moore-Feature	0.679±0.011	0.868±0.065	0.525±0.278	0.530±0.287
Joint-Feature	0.817±0.003	0.879±0.023	0.735±0.009	0.717±0.054
Hybrid-Feature	**0.931±0.007**	**0.950±0.018**	**0.894±0.007**	**0.841±0.056**

5.2.2 Robustness evaluation results

When a traffic classification model is deployed on the Internet, the model has to classify the future data from unknown mobile devices or unknown network environment. This means that the data distribution of the training data may be much different from the future unknown data. The classification performance of the classifier with these features may gradually become ineffective. In order to evaluate the robustness of different feature sets, this section evaluates the classification performance on testing data from known/unknown mobile devices and environment.

According to the results in section 5.2.1, the Hybrid-Feature, Basic-Feature and Service-Feature perform better than others. They are more suitable for mobile app traffic classification. Therefore, this section mainly evaluates the robustness of the three feature sets. Experiments were carried out from the following three cases. The three cases are mainly designed by considering that the mobile devices and the *mobilegt* server can be factors that change over time. As our *mobilegt* system could be deployed at multiple servers, a change of server may also lead to a change in data distribution as the network environment of the data collection has changed. Also, the mobile device is the impact factor on the data distribution.

1) In the first case, the classifier is trained and tested on the data collected from the same mobile devices and *mobilegt* server. The training data and testing data do not overlap.

2) In the second case, the classifier is trained on the data from a set of mobile devices (M_1) and is tested on the data from another set of mobile devices (M_2), where $M_1 \cap M_2 = \emptyset$ and the *mobilegt* servers are the same for the training and testing data

3) In the third case, the classifier is trained on the data from one *mobilegt* server and tested on the data from another *mobilegt* server. The training and testing data come from the same mobile device set.

Table 11 Robustness evaluation results of three feature sets in the three cases

Case 1: Training and testing data collected at same mobile device set and server									
Coarse-grained traffic data					Fine-grained traffic data				
Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*	Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*
Service-Feature	0.622	0.704	0.250	0.127	Service-Feature	0.592	0.575	0.511	0.344
Basic-Feature	0.799	0.882	0.700	0.690	Basic-Feature	0.774	0.870	0.726	0.640
Hybrid-Feature	**0.841**	**0.902**	**0.748**	**0.716**	Hybrid-Feature	**0.824**	**0.888**	**0.785**	**0.680**
Case2: Training and testing data collected at same server but different mobile devices									
Coarse-grained traffic data					Fine-grained traffic data				
Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*	Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*
Service-Feature	0.604	0.418	0.497	0.268	Service-Feature	0.506	0.362	0.376	0.244
Basic-Feature	0.693	0.550	0.486	0.326	Basic-Feature	0.637	0.482	0.539	**0.369**
Hybrid-Feature	**0.793**	**0.630**	**0.547**	**0.367**	Hybrid-Feature	**0.743**	**0.491**	**0.620**	0.367
Case 3: Training and testing data collected at the same mobile device set but different server									
Coarse-grained traffic data					Fine-grained traffic data				
Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*	Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*
Service-Feature	0.565	0.489	0.361	0.176	Service-Feature	0.539	0.205	0.000	0.000
Basic-Feature	0.662	0.573	0.582	0.415	Basic-Feature	0.609	0.353	**0.448**	0.181
Hybrid-Feature	**0.718**	**0.578**	**0.587**	**0.417**	Hybrid-Feature	**0.656**	**0.478**	0.417	**0.191**

The robustness evaluation results of the three feature sets in the above three cases are shown in Table 11. *Facc*, *Bacc*, *Fgmean* and *Bgmean* respectively denote *flow accuracy*, *byte accuracy*, *flow g-mean* and *byte g-mean*. The results both on the coarse-grained data and fine-grained data are shown. The experimental results analysis follows.

1) Comparing the results obtained on the three cases.

We can see that the performance in case 1 (the first part of Table 11) is much better than the one obtained in case 2 (the

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

second part of Table 11) and case3 (the third part of Table 11), and that case 3 is the worst (only 71.8% *flow accuracy* and 57.8% *byte accuracy* on coarse-grained data, 65.6% *flow accuracy* and 47.8% *byte accuracy* on fine-grained data). This suggests that the data collection environment (*mobilegt* server) has the largest influence on the data distribution. This is because that the two *mobilegt* servers in our experiment are deployed in different countries. Their network environment is much different from one another.

2) Comparing the results of the three feature sets

Among the three feature sets, Service-Feature performs the worst on the three cases of coarse-grained and fine-grained data. This may result from the fact that the features in the Service-Feature set are unstable. Through further analysis of the features in Service-Feature, we found that three features are unstable and their *HD* values are much higher than that of the others. Taking the dataset in Case 1 as an example, the *HD* value of each feature is shown in Fig.3. This figure shows that the *HD* values of the three feature sets is equal to 1. This indicates that severe drift occurs to the three feature sets. As a result, the classifier trained with the three feature sets is over-fitting the data.

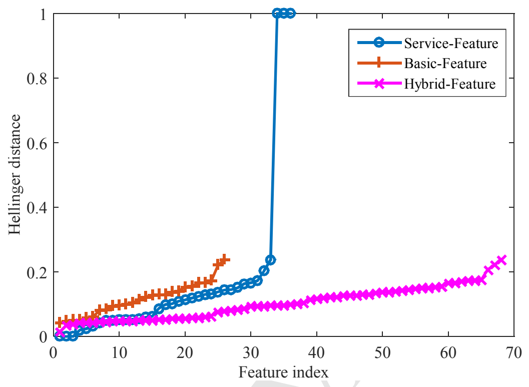


Fig. 3 *HD* of each feature from different feature sets

When Hybrid-Feature is compared to Basic-Feature, Hybrid-Feature performs better. The *flow accuracy*, *byte accuracy*, *flow g-mean* and *byte g-mean* improve by an average of about 9.38%, 5.89%, 6.76% and 5.61% respectively on the data labeled in a coarse-grain, and improve by an average of about 10.27%, 13.12%, 5.41% and 3.74% on the data labeled in a fine-grain. Hybrid-Feature is the extended version of Basic-Feature. This demonstrates that the newly added features are able to improve the classification robustness.

5.3 Comparison experiments on the feature selection algorithms

This section aims to evaluate the performance of DDFS. The experimental results discussions consider the following three aspects: overall classification performance, per class classification performance and selected features.

5.3.1 Overall classification performance

To evaluate the robustness of the selected features, experiments are carried out on the three cases discussed in section 5.2.2. Hybrid-Feature is used to build the flow sample set for the training classification model. DDFS is compared with existing feature selection algorithms, including Information Gain, Gain Ratio, Chi-Square, Relief, SRSF and GOA. The first four filter feature selection algorithms are implemented in the Weka toolkit [44]. SRSF and GOA are reproduced according to their algorithms described in [16] and [15]respectively. In the DDFS algorithm, the input parameter β is the parameter in Algorithm 1, and it is set to 0.1; the N_f parameter is set to 30. The influence of the parameter values will be discussed in Section 5.5. Also, the experiments are carried out both on coarse-grained and fine-grained data.

The classification results in cases 1, 2 and 3 are shown in Tables 12, 13 and 14, respectively. The DDFS(DDRU) and DDFS(HD) refer to the feature subset selected by DDFS with *DDRU* and *HD* metrics respectively. The results obtained with the full feature set (denoted by Fullset) is given as the base line for comparison.

ACCEPTED MANUSCRIPT

Table 12 The performance of the classifier with different feature selection algorithms on coarse-grained and fine-grained data in Case 1

Coarse-grained data					Fine-grained data				
Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*	Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*
Fullset	0.841	0.902	0.748	**0.716**	Fullset	0.824	0.888	0.785	0.680
Information Gain	0.857	0.913	0.792	0.696	Information Gain	0.793	0.868	0.750	0.615
CHI	0.837	0.908	0.774	0.704	CHI	0.806	0.888	0.768	**0.685**
Relief	0.823	0.906	0.716	0.596	Relief	0.831	**0.894**	0.790	0.653
SRSF	**0.874**	0.905	**0.820**	0.617	SRSF	**0.851**	0.877	**0.822**	0.623
GOA	0.843	0.914	0.777	0.700	GOA	0.805	0.879	0.773	0.673
Gain Ratio	0.845	0.916	0.738	0.686	Gain Ratio	0.831	**0.894**	0.790	0.653
DDFS(DDRU)	0.852	**0.931**	0.750	0.697	DDFS(DDRU)	0.843	0.893	0.821	0.672
DDFS(HD)	0.859	0.924	0.788	0.690	DDFS(HD)	0.833	0.887	0.795	0.663

Table 13 The performance of the classifier with different feature selection algorithms on coarse-grained and fine-grained data in Case 2

Coarse-grained data					Fine-grained data				
Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*	Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*
Fullset	0.793	**0.630**	0.547	0.367	Fullset	0.743	0.491	0.620	0.367
Information Gain	0.790	0.548	0.617	0.366	Information Gain	0.706	0.424	0.615	0.355
CHI	0.762	0.588	0.586	0.347	CHI	0.709	0.431	0.633	0.379
Relief	0.741	0.590	0.555	0.344	Relief	0.709	0.455	0.623	0.403
SRSF	0.787	0.559	0.529	0.331	SRSF	0.747	0.589	0.605	0.422
GOA	0.779	0.624	0.567	0.354	GOA	0.698	0.447	0.623	0.365
Gain Ratio	0.796	0.563	**0.655**	0.404	Gain Ratio	0.725	0.529	**0.664**	0.433
DDFS(DDRU)	0.798	0.585	0.613	0.372	DDFS(DDRU)	0.725	0.529	**0.664**	0.433
DDFS(HD)	**0.807**	0.621	0.647	**0.408**	DDFS(HD)	**0.762**	**0.624**	0.662	**0.475**

The classifiers in cases 2 (Table 13) and 3(Table 14) perform worse than that in case 1 (Table 12). This is because the drift of feature value distribution is more severe in cases 2 and 3. This is similar to the results shown in Table 11. The results demonstrate that DDFS is able to further improve the classification performance in most situations after performing feature selection. When compared to fullset, the *flow accuracy, byte accuracy, flow g-mean* and *byte g-mean* are improved on average by about 1.8%, 6.9%, 8.9% and 10.1%, respectively. However, some of the other feature selection algorithms may degrade the classification performance after feature selection, such as Relief and CHI.

Table 14 The performance of the classifier with different feature selection algorithms on coarse-grained and fine-grained data in Case 3

Coarse-grained data					Fine-grained data				
Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*	Features	*Facc*	*Bacc*	*Fgmean*	*Bgmean*
Fullset	0.718	0.578	0.587	0.417	Fullset	0.656	0.478	0.417	0.191
Information Gain	0.711	0.573	0.581	0.392	Information Gain	0.660	0.376	0.412	0.162
Chi-Squire	0.711	0.573	0.581	0.392	CHI	0.647	0.362	0.422	0.159
Relief	0.699	0.564	0.581	0.371	Relief	0.642	0.427	0.406	0.180
SRSF	0.707	0.641	0.584	0.391	SRSF	0.652	0.425	0.395	0.154
GOA	0.711	0.607	0.583	0.382	GOA	0.666	0.329	0.418	0.134
Gain Ratio	0.723	**0.642**	0.607	0.411	Gain Ratio	0.658	0.403	0.443	0.172
DDFS(DDRU)	**0.726**	0.633	0.607	0.407	DDFS(DDRU)	0.668	0.479	0.459	0.194
DDFS(HD)	0.724	0.640	**0.627**	**0.468**	DDFS(HD)	**0.672**	**0.491**	**0.481**	**0.218**

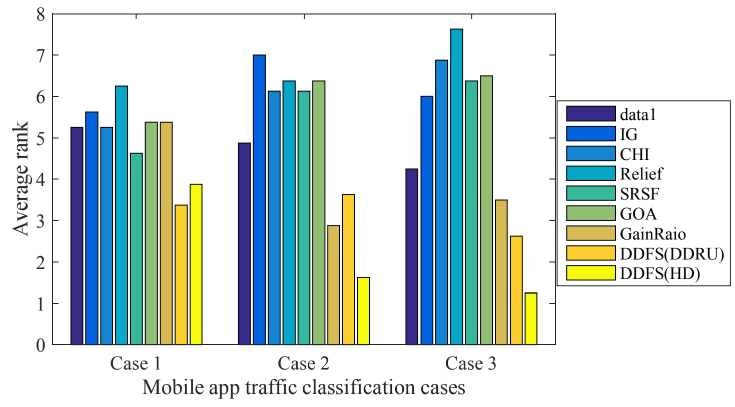


Fig. 4 the average rank of different feature selection algorithms

ACCEPTED MANUSCRIPT

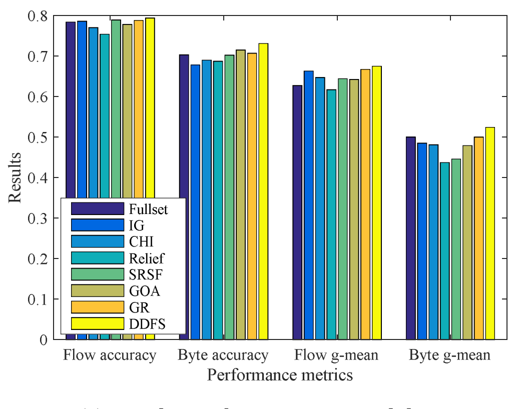
Among the 8 feature selection algorithms, none performs best all the time. For example, SRSF performs best in terms of *flow g-mean* in case 1, where it obtains 87.4% and 85.1% *flow g-mean* on coarse-grained data and fine-grained data respectively. But, its performance is worse than Fullset in terms of some performance metrics in cases 2 and 3. It is hard to see which one is the best when only observing the numbers in the three tables. To clearly compare the classification performance of these feature selection algorithms, the average rank index of each feature selection algorithm is shown in Fig.4. The results of each column from Tables 12 to 14 are first sorted in descending order and each feature selection algorithm gets a rank index in each column. After that, the rank indexes of each feature selection algorithm are averaged and shown in Fig.4.

Fig.4 shows that the average rank index of DDFS(HD) is close to 1 in cases 2 and 3, this means that the results obtained by DDFS(HD) are generally at the top on both the coarse-grained and fine-grained data. This indicates that DDFS cannot always perform best, but its performance is close to the best even when it is not exactly the best. In addition, the average over the results of the three cases are shown in Fig.5. DDFS performs better than the others on average. It obtains, on average, about 80% *flow accuracy*, 72.4% *byte accuracy*, 67.1% *flow g-mean*, 50.5% *byte g-mean* on coarse-grained data, and 76.9% *flow accuracy*, 68.5% *byte accuracy*, 64.7% *flow g-mean*, 44.0% *byte g-mean* on fine-grained data. The effectiveness of DDFS comes from removing the features with an outstanding degree of drift while keeping the features with high discrimination power. The features selected by DDFS perform better on traffic from unknown devices or environment. The classification results in Tables 12 to 14 and Figs.3 and 4 demonstrate that DDFS is able to improve classification robustness.

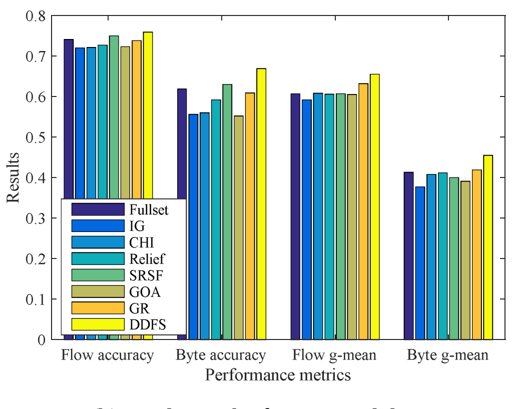
DDFS applies Gain Ratio as the discrimination power metric, and HD or DDRU as the degree of drift metrics. It aims to search for discriminative and stable features. Next, we will take a deeper look at DDFS by comparing it against Gain Ratio and discussing how suitable the two metrics used in DDFS are.

1) The results in Tables 12 to 14 show that DDFS performs better than Gain Ratio, especially in cases 2 and 3. When compared to the results of Gain Ratio, the *flow accuracy, byte accuracy, flow g-mean* and *byte g-mean* are respectively improved, on average, by about 0.8%, 3.35%, 1.2% and 4.8% on the data labeled in coarse-grained, and are improved by about 2.84%, 9.97%, 3.53% and 8.51% respectively on the data labeled in fine-grained.

2) When HD is compared to DDRU in DDFS, DDRU performs better than HD in case 1, but HD performs better in cases 2 and 3 according to the results in Fig.4. The unlabeled data that will be classified are used by HD to evaluate the degree of drift of each feature whereas, only the training data are used by DDRU to evaluate the degree of drift of each feature. In case 1, the training and testing data are from the same mobile devices, the degree of drift of each feature is not obvious. In such a case, the supervised degree of drift metric is more effective. In cases 2 and 3, the data distribution on the training data is much different from that in the testing data. In such a case, the unsupervised degree of drift metric is more effective.



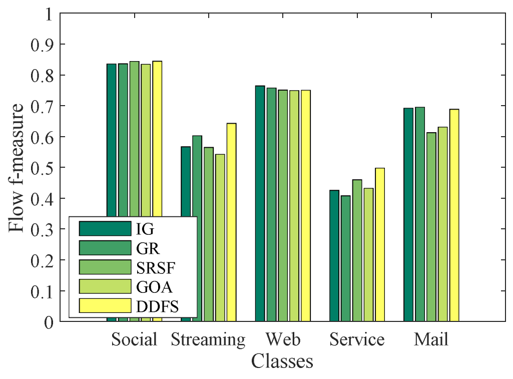
(a) Results on the coarse-grained data



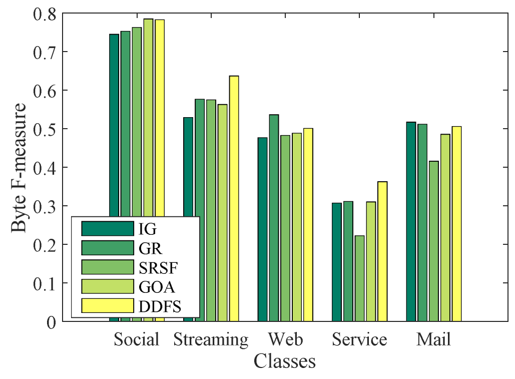
(b)Results on the fine-grained data

Fig. 5 Average flow/byte accuracy, flow/byte g-mean obtained by each feature selection algorithm in the three cases

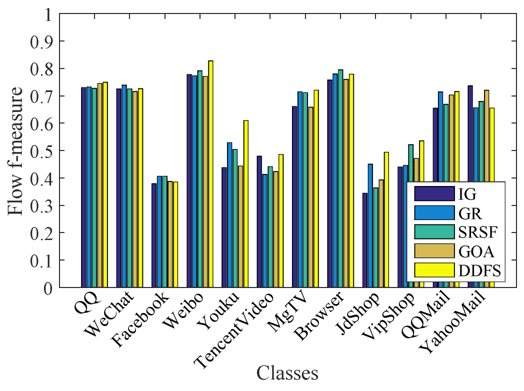
ACCEPTED MANUSCRIPT



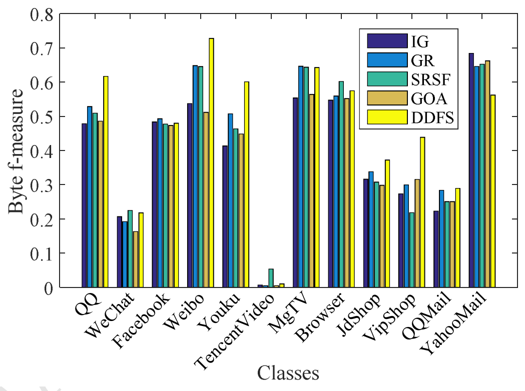
(a) *Flow f-measure* on coarse-grained data



(b) *Byte f-measure* on coarse-grained data



(c) *Flow f-measure* on fine-grained data



(d) *Byte f-measure* on fine-grained data

Fig. 6 per class classification results of IG, GR, SRFS, GOA and DDFS in terms of flow/byte *f-measure*

5.3.2 Per class classification performance

According to the results in section 5.3.1, IG(Information Gain), GR(Gain Ratio), SRSF, GOA and DDFS (DDRU in case 1 and HD in cases 2 and 3) perform better than the other methods on the mobile app traffic data. They are further compared in terms of per classification performance in this section. The *flow f-measure* and *byte f-measure* of each feature selection algorithm are averaged over the results of the three cases, and results are shown in Fig. 6. On the coarse-grained data, DDFS performs the best on some categories (such as Social, Streaming and Service), and its performance is close to the best when it is not the best (such as the *flow/byte f-measure* of Mail, and *byte/byte f-measure* of Web). On the fine-grained data, DDFS performs significantly better than the other methods on some apps (such as QQ, Weibo, Youku and VipShop). It obtains worse *byte f-measure* for TencentVideo and YahooMail, this may result from the smaller number of flow samples in the two classes. The HD is based on the feature value distribution, it may not perform well when the flow samples are scarce. When DDFS is compared to Gain Ratio, it indeed improves its per class classification performance for some categories (such as Social, Streaming)/apps(such as WeChat, QQ, Weibo etc.).

5.3.3 Selected features

To further illustrate the effect of DDFS on improving the performance of Gain Ratio (the basic discrimination power metric), Table 15 shows the features selected by DDFS and Fig.7 exhibits the degree of drift of these selected features. In the experiments reported in Tables 12 to 14, there are 6 independent experiments (coarse-grained data and fine-grained data in three cases) and 6 feature subsets obtained by each feature selection algorithm on the 6 experiments. The selected features with occurrence frequency ≥ 6 are given in Table 15.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Table 15 High frequent features selected by Gain Raito and DDFS

Feature selection algorithm	Selected features
DDFS	{dstport, dstIp, proto, iIAT(max) oPS (min, max, std), iIPS(mean, min), oIPS(min), SFPS(min, max), SFIAT(max), FHPS, THPS, PSI-3, PSI-4, PSI-6, PSI-8, oSE}
Gain Ratio	{dstport, dstIp, proto, iIAT(mean, max), oFS, iPS(max), oPS (min, max, std), iIPS(min), oIPS(max), SFPS(min, mean, max), SFIAT(max), FHPS, SHPS, THPS, PSI-0, PSI-3, PSI-8, PSI-6, oSE}

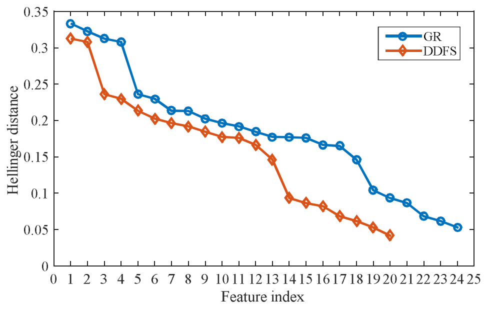


Fig. 7 HD of each selected feature

Table 15 shows that the two feature subsets are similar to each other. The packet header based features are all selected by both Gain Ratio and DDFS. Other selected features are the in-flow behavior features and distribution features. The HD of each feature in the two feature subsets are shown in Fig. 7, where the features are sorted in descending order according to the value of HD. The x-axis is the feature index in the sorted feature sequence, y-axis is the HD value. It shows that the features obtained by DDFS have much smaller HD value than those selected by Gain Ratio. This further demonstrates that DDFS is able to improve Gain Ratio in terms of classification robustness.

5.4 Experimental results on WLAN data

To evaluate the generality of Hybrid-Feature and DDFS, this section carries out experiments on the traffic data collected from the WLAN network of SCUT. The cross validation results, and robustness evaluation results are shown in Table 16. HD is used as the degree of drift metric of DDFS as the wireless devices on the testing data may be unseen on the training data of SCUT data.

The cross validation classification results (the first subpart of Table 16) show that Hybrid-Feature outperforms Basic-Feature in terms of *flow accuracy* and *byte accuracy*. The classifier trained on Hybrid-Feature obtains 92.2% *flow accuracy* and 95.4% *byte accuracy*. To further analyze the per class classification performance, we found that Download obtains worse *recall* when the results of Hybrid-Feature are compared to those obtained on Basic-Feature, leading to worse *flow g-mean* and *byte g-mean*.

Table 16 Experimental results on WLAN traffic data

Cross validation results on the data with full feature set				
Features	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
Basic-Feature	<u>0.889±0.005</u>	<u>0.929±0.016</u>	**0.697±0.254**	**0.726±0.272**
Service-Feature	0.891±0.006	0.924±0.016	0.466±0.402	0.488±0.422
Hybrid-Feature	**0.922±0.004**	**0.954±0.010**	<u>0.666±0.004</u>	<u>0.675±0.261</u>
Robustness evaluation results on the data with full feature set (SCUT1 vs. SCUT2)				
Features	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
Basic-Feature	<u>0.800</u>	<u>0.796</u>	**0.532**	**0.546**
Service-Feature	0.777	0.747	<u>0.391</u>	<u>0.378</u>
Hybrid-Feature	**0.891**	**0.861**	0.000	0.000
Robustness evaluation results of feature selection algorithm(SCUT1 vs. SCUT2)				
Feature selection algorithms	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
IG	<u>0.904</u>	0.849	<u>0.488</u>	<u>0.476</u>
SRSF	**0.905**	<u>0.862</u>	0.000	0.000
GOA	0.901	**0.865**	0.000	0.000
GR	**0.905**	0.857	0.000	0.000
DDFS(HD)	0.903	0.853	**0.608**	**0.588**

On the robustness evaluation results with a full feature set (the second sub part of Table 16), Table 16 shows that Hybrid-Feature still obtains the highest *flow accuracy* (89.1%) and *byte accuracy* (86.1%) but the worst *flow g-mean* (0%)

ACCEPTED MANUSCRIPT

and *byte g-mean* (0%). The bad g-mean results from the 0% *recall* of Download.

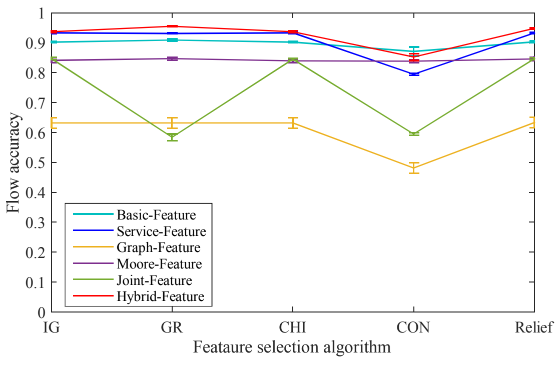
On the results obtained by different feature selection algorithms (the third subpart of Table 16), DDFS performs the best in terms of *flow g-mean* (60.8%) and *byte g-mean* (58.8%) while maintaining high *flow accuracy* (90.3%) and *byte accuracy* (85.3%). DDFS removes unstable and irrelevant features and improves the *recall* of the minority class (Download) from 0% to 100%. In addition, the experiments further demonstrates that our feature selection algorithm is able to improve on the performance of the algorithm based only on the discrimination power metric (i.e., Gain Ratio).

5.5 Discussion

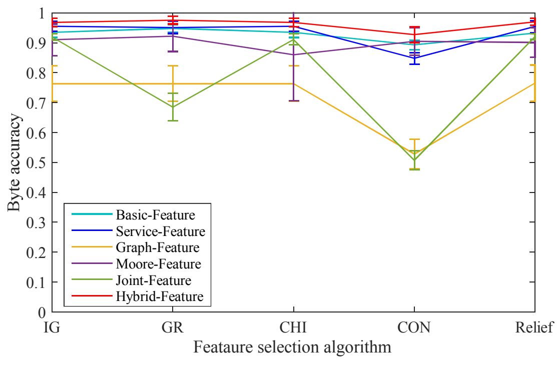
5.5.1 Discussion about the feature sets

(1) The impact of the feature selection algorithms

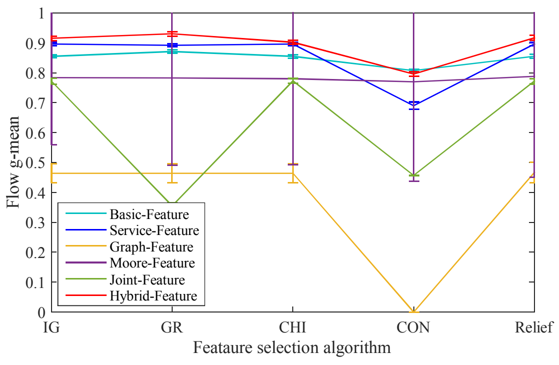
This paper presents a feature set named Hybrid-Feature and compares it to 5 existing feature sets that are extracted from different perspectives. In this section, we further discuss the impact of the feature selection on these feature sets. A good feature set is expected not to be sensitive to the feature selection algorithm. Multiple filter feature selection algorithms (Information Gain[12], Gain Ratio[12], Chi-square[14], CON[11] and Relief[13]) are run on the datasets characterized by different feature sets, so as evaluate the influence of feature selection algorithms on each feature set. The *flow accuracy*, *byte accuracy*, *flow g-mean* and *byte g-mean* results on the coarse-grained data are shown in Fig.8(a)~(d), and that on the fine-grained data are shown in Fig.8(e)~(h) . The x-axis represents the feature selection algorithms, and the y-axis represents the classification performance results.



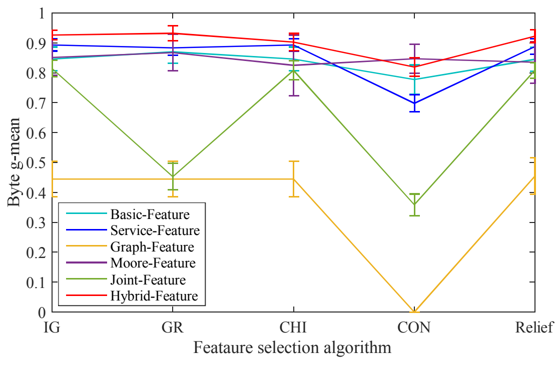
(a) *flow accuracy* on coarse-grained data



(b) *byte accuracy* on coarse-grained data

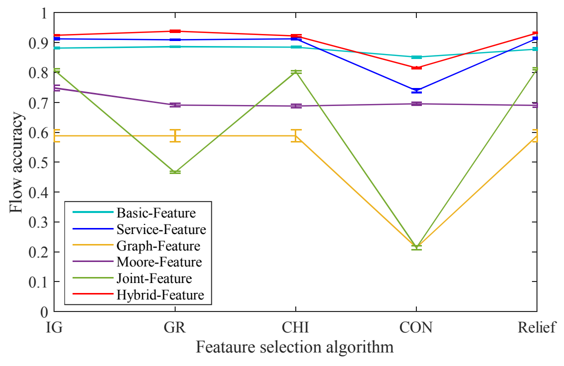


(c) *flow g-mean* on coarse-grained data

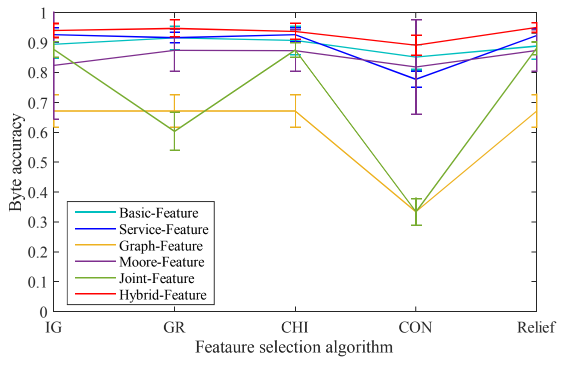


(d) *byte g-mean* on coarse-grained data

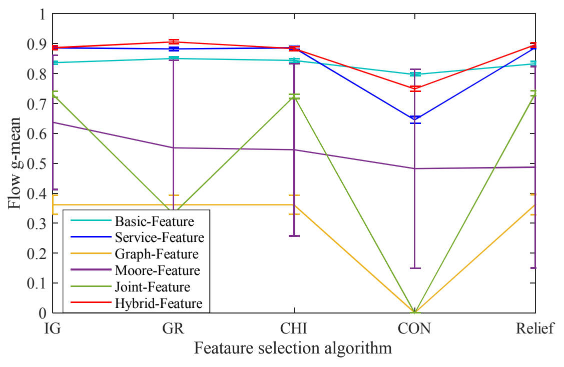
ACCEPTED MANUSCRIPT



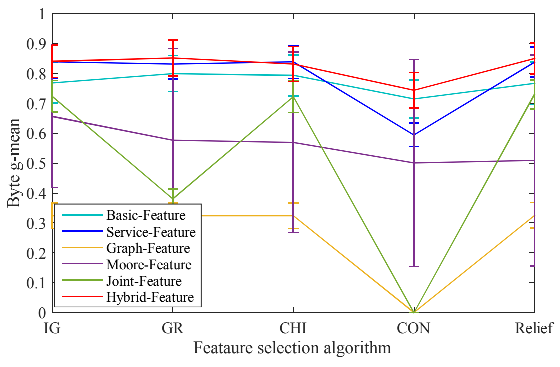
(e) *flow accuracy* on fine-grained data



(f) *byte accuracy* on fine-grained data



(g) *flow g-mean* on coarse-grained data



(h) *byte g-mean* on coarse-grained data

Fig. 8 Classification results of the model with different feature sets and different feature selection algorithms

On the coarse-grained and fine-grained data, Basic-Feature, Service-Feature and Hybrid-Feature still perform better and are more stable than the other feature sets. The Graph-Feature and Joint-Feature sets perform much worse in some cases such as when using the CON algorithm (shown in Figs.7(c), (d), (g) and (h)). These feature sets are, thus, not as effective as the others and may include some unstable features, which are easily influenced by network environment, such as the time related features. The Joint-Feature set closely correlated with the inter packet arrival time. The Graph-Feature set assumes that the traffic in a time period comes from the same app. However, this assumption cannot hold since the traffic may come from multiple apps during a time period. As a result, some traffic may be wrongly identified. Using different feature selection algorithms, the classifier that uses the Hybrid-Feature set always obtains the best results except on CON (though the results are still close to the best, such as in Fig.8(a), (e) and (g)). This further illustrates that Hybrid-Feature is more suitable for mobile app traffic data.

(2) The impact of the fold of cross validation

In the cross validation experiments from section 5.2.1, we applied 10-fold cross validation. To check whether the results vary from fold to fold, the 5-fold cross validation results on coarse-grained and fine-grained data are shown in Tables 17 and 18 respectively. The results obtained with 5-fold cross validation change a little when compared to the results in Tables 9 and 10. For example, using Hybrid-Feature on coarse-grained data, *the flow accuracy, byte accuracy, flow g-mean*, *byte g-mean* are changed from 94.6%, 97%, 91.1%, 91.5 to 94.3%, 96.6%, 90.6%, 90.6%. The largest difference is only 0.9% (byte g-mean performance). In addition, the classifier that uses Hybrid-Feature still performs best, and is followed by the Basic-Feature and Service-Feature sets.

ACCEPTED MANUSCRIPT

Table 17 the 5-fold cross validation results on coarse-grained data

Feature sets	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
Basic-Feature	0.922±0.021	0.935±0.032	0.891±0.018	0.885±0.033
Service-Feature	0.915±0.001	0.936±0.012	0.873±0.001	0.868±0.019
Graph-Feature	0.794±0.014	0.802±0.018	0.570±0.036	0.545±0.045
Moore-Feature	0.815±0.004	0.918±0.031	0.735±0.007	0.832±0.030
Joint-Feature	0.847±0.002	0.913±0.010	0.767±0.005	0.799±0.022
Hybrid-Feature	**0.943±0.002**	**0.966±0.012**	**0.906±0.005**	**0.906±0.013**

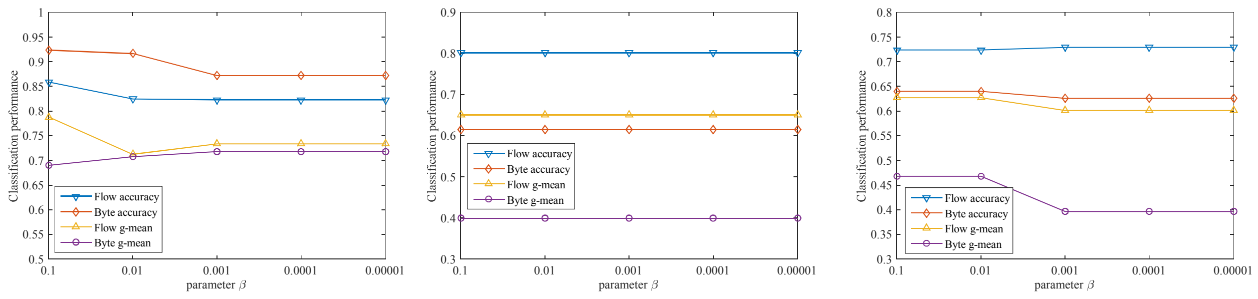
Table 18 the 5-fold cross validation results on fined-grained data

Feature sets	*Flow acc.*	*Byte acc.*	*Flow g-mean*	*Byte g-mean*
Basic-Feature	0.893±0.028	0.889±0.033	0.856±0.027	0.773±0.060
Service-Feature	0.889±0.000	0.918±0.007	0.858±0.005	0.816±0.021
Graph-Feature	0.676±0.037	0.672±0.044	0.364±0.023	0.317±0.030
Moore-Feature	0.686±0.009	0.873±0.043	0.518±0.291	0.533±0.300
Joint-Feature	0.812±0.003	0.876±0.025	0.728±0.005	0.711±0.055
Hybrid-Feature	**0.927±0.001**	**0.932±0.038**	**0.889±0.001**	**0.822±0.064**

5.5.2 Discussion about the parameters of DDFS

(1) The influence of parameter β

The influence of parameter β in Algorithm 1 is discussed in this section. It is set in the range of {0.1, 0.01, 0.001, 0.0001}. The classification results obtained with different values of β in the three cases are shown in Fig.9. The x-axis represents the value of β, and the y-axis represents the classification performance. In case 2 (Fig.9(b)), the classification performance is not influenced by β. In this case, the *SSE* does not decrease with further searching. This is because the values in V_{large} are much higher than those in V_{small}. The performance in cases 1 and 3 (Fig.9(a) and Fig.9(c)) degrades when decreasing β. When decreasing β, the threshold for selecting the 'danger' features is smaller, which means that more features would be selected into S_{danger}. We found that the features with small degree of drift (HD<0.3) may be selected into S_{danger} when β becomes smaller in case 1. Therefore, β=0.1 is more suitable for searching for the outstanding threshold for S_{danger}.



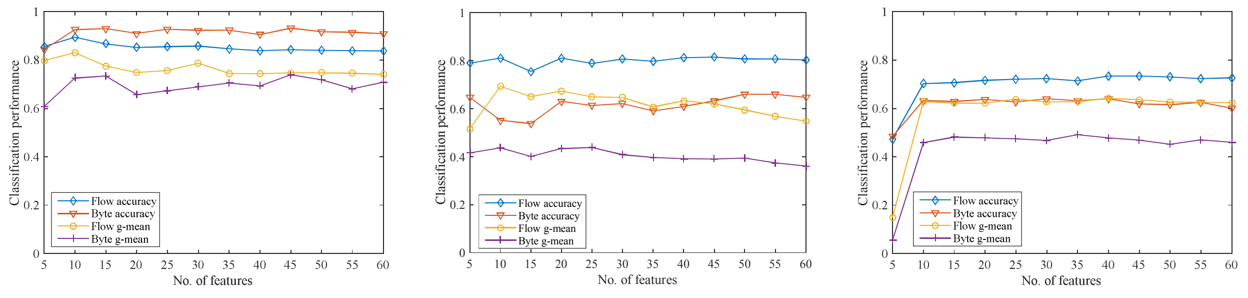
(a) Case 1 (b) Case 2 (c) Case 3

Fig. 9 the classification performance with different values of β in DDFS

(2)The influence of parameter N_f

To discuss the influence of the number of selected features (N_f of Algorithm 1) in DDFS, we carry our experiments using different values of N_f. The N_f is set in the range of {5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60}. The classification results obtained with different values of N_f in the three cases are shown in Fig. 10. The x-axis represents the value of N_f, and the y-axis represents the classification performance. There are some fluctuations when the number of features is less than 30 (in Fig.10(a) and (b)). When continuously increasing the number of selected features, *flow accuracy* becomes more stable. The *flow g-mean* tends to decrease. The possible reason is that the extra features selected favor the majority class on the class imbalanced mobile app traffic data.

ACCEPTED MANUSCRIPT



(a) Case 1 (b) Case 2 (c) Case 3

Fig. 10 The classification performance with different values of N_f in DDFS

5.5.3 Time consumption

DDFS combines the discrimination power metric and the degree of drift evaluation metric. In this section, we take Gain Ratio as the discrimination power metric, and compare the time consumption of DDFS against that of Gain Ratio. The time consumption of Gain Ratio, DDFS(DDRU) and DDFS(HD) are shown in Fig. 11, which are the average over 10 independent runs. The x-axis represents the experiment cases, and the y-axis represents the time consumption. For example, Case1_CG refers to the experiment in case 1 on coarse-grained data, and Case1_FG refers to the experiment in case 1 on fine-grained data. DDFS mainly includes two parts: discrimination power evaluation and degree of drift evaluation for each feature. Hence, it consumes more time than the algorithm based on the discrimination power metric only. For example, DDFS(HD) consumes about 40 s on the data with 88, 763 flow samples, whereas Gain Ratio consumes about 16 s. The results show that DDFS(DDRU) consumes much more time on the fine-grained data, because it calculates the DDRU for each app, and the number of apps (on fine-grained data) are larger than the number of traffic categories (on coarse-grained data). As the discrimination power and the degree of drift could be evaluated independently, the time consumption could be reduced by performing the two tasks simultaneously.

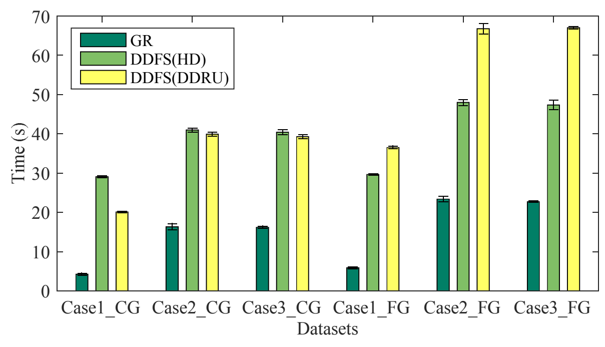


Fig.11 Time consumption of feature selection algorithms

6 Conclusion and future work

This paper explores the in-flow behavior of mobile app traffic flows in terms of packet size and inter arrival time. Based on our exploratory experiments, we extract a feature set named Hybrid-Feature for mobile app traffic data and propose a feature selection algorithm named DDFS to select a stable and discriminative feature subset for improving mobile app traffic classification robustness. The Hybrid-Feature set is an extended version of Basic-Feature set which adds the in-flow behavior features, distribution features and packet header features. In the feature selection algorithm, two new degree of drift metrics are proposed from two perspectives. The first metric (a supervised metric) is based on relative uncertainty and evaluates the stability of a feature's ability to characterize each class. The second metric (an unsupervised metric) is based on the Hellinger distance and evaluates the change of a feature with respect to the feature value distribution.

Our experiments were carried out on the data labeled in a coarse-grained and a fine-grained manner, so as to evaluate

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

the performance of different feature sets and feature selection algorithms on data carrying different label information. The effectiveness of Hybrid-Feature and DDFS was validated by comparing them against existing feature sets and feature selection algorithms. To evaluate the classification robustness of our proposed feature set and feature selection algorithm, the trained classifier was used to classify unknown data from different mobile devices and environment. The experimental results can be summarized as follows.

(1) Hybrid-Feature performs best in most cases on both the cross validation results and the robustness evaluation results. This suggests that the newly added features help improve classification performance over the Basic-Feature.

(2) The feature selection algorithm validation results show that DDFS is able to improve the classification performance of traditional feature selection algorithms. Furthermore, DDFS(HD) performs better than DDFS(DDRU) when severe drift happens to the traffic data.

(3) The discussion regarding the influence of feature selection algorithms on the feature sets show that Hybrid-Feature, Basic-Feature and Service-Feature are more stable than others.

The classification results also exhibit some of the shortcomings of our work. DDFS requires to pre-set the number of selected features and consumes more time calculating the evaluation metrics of the features. One of our proposed future work is to optimize our feature selection algorithm for online feature searching [47]. The robustness evaluation results in section 5.3.1 also show that the classifier does not perform well in case 3. It only obtains 72.4% *flow accuracy* and 64% *byte accuracy* in that case. These results are consistent with the results in [20], which show that the classifier identifies the apps with 67% *accuracy* when the testing data comes from an unseen devices. Improving the classification performance obtained on traffic from unseen devices remains an open problem which must be studied in the future, especially as more mobile apps and user behaviors are identified.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the National Natural Science Foundation of China under Grant No. 61501128, Natural Science Foundation of Guangdong Province (Nos.2017A030313345, 2016A030310300, 2015A030310267, 2014A03031358), the Specialized Fund for the Basic Research Operating expenses Program of Central College (No. x2rj/D2174870), and Guangdong Province Youth Innovation Talent Project (2014KQNCX139).

References

[1] A. Tongaonkar. A Look at the Mobile app identification landscape. IEEE Internet Computing, 2016, 20(4):9-15.

[2] S. Mongkolluksamee, V. Visoottiviseth, K. Fukuda. Enhancing the performance of mobile traffic identification with communication patterns. IEEE Computer Software and Applications Conference, 2015:336-345.

[3] S. Dai, A. Tongaonkar, X. Wang, et al. NetworkProfiler: Towards automatic fingerprinting of Android apps. Proceedings of IEEE INFOCOM, 2013:809-817.

[4] S. Miskovic, G. M. Lee, Y. Liao, et al. AppPrint: Automatic fingerprinting of mobile applications in network traffic. 2015.

[5] S. Lee, J. Song, S. Ahn, et al. Session-based classification of internet applications in 3G wireless networks. Computer Networks, 2011, 55(17): 3915 - 3931.

[6] P. Bermolen, M. Mellia, M. Meo, et al. Abacus: Accurate behavioral classification of P2P-TV traffic. Computer Networks, 2011, 55(6):1394-1411.

[7] A. Hajjar, J. Khalife, J. Díaz-Verdejo. Network traffic application identification based on message size analysis. Journal of Network & Computer Applications, 2015, 58:130-143.

[8] Z. Liu, R. Wang. Mobilegt: A system to collect mobile traffic trace and build the ground truth. Proceedings of IEEE Telecommunication Networks and Applications, 2017:142-144.

[9] A.Moore, D. Zuev, M. Crogan. Discriminators for use in flow-based classification. Queen Mary and Westfield

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

College, Department of Computer Science, 2005.

[10] A. Dainotti, A. Pescape, Kim H C. Traffic classification through Joint distributions of packet-level statistics. Proceedings of IEEE Global Telecommunications, 2011:1-6.

[11] H. Liu, H. Motoda, Feature selection for knowledge discovery and data mining, Springer, 1998.

[12] J. Han, M. Kamber, Data mining: concepts and techniques, Morgan Kaufmann,2006.

[13] K.Kira, L. Rendell, The feature selection problem: traditional methods and new algorithm. Proceedings of AAAI. San Jose, CA, July 1992.

[14] H. Liu, R. Setiono, Chi2: feature selection and discretization of numeric attributes, Proceedings of the International Conference on Tools with Artificial Intelligence, 1995: 388–391.

[15] A.Fahad, Z.Tari, I.Khalil, A.Almalawi, An optimal and stable feature selection approach for traffic classification based on multi-criterion fusion, Future Generation Computing Systems 36(2014): 156–169.

[16] H. Zhang, G. Lu, M. T. Qassrawi, et al. Feature selection for optimizing traffic classification. Computer Communications, 2012, 35(12):1457-1471.

[17] Q. Xu, J. Erman, A. Gerber, et al. Identifying diverse usage behaviors of smartphone apps. Proceedings of the ACM SIGCOMM conference on Internet measurement conference, 2011: 329-344.

[18] J. Sun, L. She, H. Chen, et al. Automatically identifying apps in mobile traffic. Concurrency & Computation Practice & Experience, 2016, 28(14):3927-3941.

[19] G. Ranjan, A. Tongaonkar, Torres R. Approximate matching of persistent LExicon using search-engines for classifying Mobile app traffic. Proceedings of IEEE INFOCOM, 2016:1-9.

[20] H. F. Alan, J. Kaur. Can android applications be identified using only TCP/IP headers of their launch time traffic? ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2016:61-66.

[21] M. Conti, L. V. Mancini, R. Spolaor, et al. Analyzing android encrypted network traffic to identify user actions. IEEE Transactions on Information Forensics & Security, 2016, 11(1):114-125.

[22] R. Alshammari, A. N. Zincir-Heywood. Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?. Computer Networks, 2011, 55(6):1326–1350.

[23] A. W. Moore, D. Zuev. Internet traffic classification using bayesian analysis techniques. Acm Sigmetrics Performance Evaluation Review, 2005, 33(1):50-60.

[24] Y. Fu, H. Xiong, X. Lu, et al. Service usage classification with encrypted Internet traffic in mobile messaging apps. IEEE Transactions on Mobile Computing, 2016, 15(11):2851-2864.

[25] A. Mcgregor, M. Hall, P. Lorier, et al. Flow clustering using machine learning techniques. International Passive and Active Network Measurement International Workshop, 2004:205-214.

[26] F. Ertam, E. Avcı. A new approach for internet traffic classification: GA-WK-ELM. Measurement, 2017, 95:135-142.

[27] B. Schmidt, A. Al-Fuqaha, A. Gupta, et al. Optimizing an artificial immune system algorithm in support of flow-Based internet traffic classification. Applied Soft Computing, 2017, 54(C):1-22.

[28] Y. D. Lin, C. N. Lu, Y. C. Lai, et al. Application classification using packet size distribution and port association. Journal of Network and Computer Applications, 2009, 32(5):1023-1030.

[29] L. Z. Peng, B. Yang, Y. Chen. Effective packet number for early stage internet traffic identification. Neurocomputing, 2015, 156(C):252-267.

[30] T. Qin, L. Wang, Z. Liu, et al. Robust application identification methods for P2P and VoIP traffic classification in backbone networks. Knowledge-Based Systems, 2015, 82(C):152-162.

[31] S. Valenti, D. Rossi. Identifying key features for P2P traffic classification. IEEE International Conference on Communications, 2011:1-6.

[32] Y. N. Dong, J. J. Zhao, J. Jin. Novel feature selection and classification of Internet video traffic based on a

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

hierarchical scheme. Computer Networks, 2017.

[33] M. Iliofotou, H. C. Kim, M. Faloutsos, et al. Graph-based P2P traffic classification at the internet backbone. INFOCOM Workshops. IEEE, 2009:1-6.

[34] Y. Jin, N. Duffield, P. Haffner, et al. Inferring applications at the network layer using collective traffic statistics. Teletraffic Congress. IEEE, 2010:1-8.

[35] H. Asai, K. Fukuda, H. Esaki. Traffic causality graphs: Profiling network applications through temporal and spatial causality of flows. Teletraffic Congress. IEEE, 2011:95-102.

[36] Z. Liu, Q. Liu. Balanced feature selection method for Internet traffic classification. IET Networks, 2012, 1(2):74-83.

[37] P. Casas, P. Fiadino, A. Bar. IP mining: Extracting knowledge from the dynamics of the Internet addressing space. Teletraffic Congress. IEEE, 2013:1-9.

[38] N. Japkowicz, S. Stephen. The class imbalance problem: A systematic study. Intell. Data Anal. 2002, 6(5): 429-449.

[39] http://www.cl.cam.ac.uk/research/srg/netos/projects/brasil/downloads/index.html

[40] K. Xu, Z. L. Zhang, Bhattacharyya S. Profiling internet backbone traffic. Acm Sigcomm Computer Communication Review, 2005, 35(4):169.

[41] L. Yin, Y. Ge, K. Xiao, et al. Feature selection for high-dimensional imbalanced data. Neurocomputing, 2013, 105(3):3-11.

[42] V. Carela-Español, T. Bujlow, P. Barlet-Ros. Is our ground-truth for traffic classification reliable?. Passive and Active Measurement. 2014: 98-108.

[43] L. Breiman. Random Forest. Machine Learning, 2001, 45:5-32.

[44] https://www.cs.waikato.ac.nz/ml/weka/

[45] L. Deri, M. Martinelli, T. Bujlow, etc. nDPI: Open-source high-speed deep packet inspection. International Computing Conference on Wireless Communications and Mobile (IWCMC), 2014: 617-622).

[46] G. Aceto, D. Ciuonzo, A. Montieri, et al. Multi-classification approaches for classifying mobile app traffic. Journal of Network and Computer Applications, 2018, 103: 131-145.

[47] G Haralabopoulos, I Anagnostopoulos. Real time enhanced random sampling of online social networks. Journal of Network and Computer Applications, 2014, 41: 126-134.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Highlights (for review)

- A flow feature set named Hybrid-Feature is extracted to characterize mobile traffic.
- A flow feature selection algorithm named DDFS is devised to improve mobile app traffic classification robustness.
- Two new metrics are proposed to assess degree of drift for each feature from different perspectives.
- Different kinds of flow feature sets were rebuilt to do comparison experiments on mobile traffic data.
- Experimental results on the real mobile app traffic data demonstrate the effectiveness of our feature set and feature selection algorithm on improving classification robustness.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Zhen Liu received the Ph.D. degree from the School of Computer Science and Technology of South China University of Technology, China, in 2013. She received her Bachelor's degree from Department of Computer Science and Technology of South West University, China in 2008. She is now a Lecturer in the School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China. Her research interests are in the areas of machine learning, mobile traffic classification .

Ruoyu Wang received the Ph.D. degree from the school of Computer Science and Engineering, South China University of Technology, China in 2015. He is now an engineer at the Information and Network Engineering and Research Center, South China University of Technology, China. His research interests are in the areas of machine learning and complex network.

Nathalie Japkowicz has been a full professor of Computer Science at American University, in Washington D.C, since August 2016. Prior to that, she directed the Laboratory for Research on Machine Learning applied to Defense and Security at the University of Ottawa in Canada. She trained over 30 graduate students and collaborated with a number of Canadian governmental agencies and private industry. Her publications include a co-authored book entitled Evaluating Learning Algorithms at Cambridge University Press, one edited book in the Springer Series on Big Data and over 100 book chapters, journal articles and conference or workshop papers. Her main research interests are in the area of machine learning.

Yongming Cai received of Ph.D. in biomedical engineer in 2012 at Sun Yat.sen University, China. He is now a professor in College of Medical Information and Engineering at GuangDong Pharmaceutical University, Guangzhou, China. His main research interests are data mining and bioinformatics.

Deyu Tang received the Ph.D. degree from the School of Computer Science and Technology of South China University of Technology, China, in 2015. He is now an associate professor in the School of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China. He research interests are in the areas of swarm intelligence, machine learning, bioinformatics.

Xianfa Cai spent his undergraduate days at Jiangxi Normal University from 1998 to 2002, went on to get his master's degree from SunYat-Sen University from 2003 to 2006, and a Ph.D. candidate at the School of Computer Science and Engineering, South China Uni-versity of Technology. He is now an associate professor at the School of Medical Information Engineering, Guangdong Pharmaceutical University. His current research interests include machine learning, pattern recognition.

ACCEPTED MANUSCRIPT