Traffic Classification of Mobile Apps through Multi-classification

Giuseppe Aceto[1,2], Domenico Ciuonzo[2], Antonio Montieri[1], Antonio Pescapé[1,2]
[1]University of Napoli "Federico II" (Italy) and [2]NM2 s.r.l. (Italy)
{giuseppe.aceto, antonio.montieri, pescape}@unina.it, ciuonzo@nm-2.com

***Abstract*—The wide spreading and growing usage of smartphones are deeply changing the kind of traffic that traverses home and enterprise networks and the Internet. Tools that base their functions on the knowledge of the application generating the traffic (performance enhancement proxies, network monitors, policy enforcement devices) imply traffic classification, and are thus limited or impaired when dealing with the daily expanding set of mobile apps. Besides the moving-target nature of mobile apps traffic, the increasing adoption of encrypted protocols (TLS) makes classification even more challenging, defeating established approaches (DPI, statistical classifiers). In this paper we aim to improve the classification performance of mobile apps classifiers adopting a multi-classification approach, intelligently-combining decisions from state-of-art classifiers proposed for mobile and encrypted traffic classification. Based on a dataset of users' activity collected by a mobile solutions provider, our results demonstrate that classification performance can be improved according to all considered metrics, up to $+8.1\%$ F-measure score with respect to the best base classifier. Further room for improvements is also evidenced by the ideal combiner performance (oracle).**

***Index Terms*—traffic classification; mobile apps; Android apps; iOS apps; encrypted traffic; information fusion; classification combining; multi-classification.**

I. Introduction

Tools like security and quality-of-service enforcement devices and network monitors base their operations on the knowledge of the application generating the traffic. The process of associating (labeling) network traffic with specific applications or application types is known as Traffic Classification (TC) and has long-established application in several fields, backed by wide scientific literature [1]. The global spread and growing usage of smartphones is profoundly changing the kind of traffic that traverses home and enterprise networks and the Internet, as a consequence both the necessity and the difficulty of TC of mobile traffic have become very high nowadays. Indeed, in addition to the traditional drivers for TC, classification of mobile apps traffic has the potential of providing extremely valuable profiling information (to advertisers, insurance companies, security agencies, or malicious parties) about users. On the other hand, it certainly raises privacy issues, especially in regards to recognition of context-sensitive apps (such as health and dating ones). Unfortunately, TC comes with its own challenges and requirements that are even exacerbated in a mobile-traffic context, that is usually characterized by the presence of a large number of apps to discriminate from and an inadequate number of training samples per app. Moreover, the increasing adoption of encrypted protocols (TLS) makes classification even more challenging, defeating established approaches.

Moving from earlier port-based methods, to those based on payload inspection (termed Deep Packet Inspection methods, DPI [2]), approaches based on Machine Learning (ML) classifiers are deemed the most appropriate, especially in this context, since they suit also Encrypted Traffic (ET) analysis [3, 4, 5, 6]. Although earlier results have been published on this topic, the traffic of mobile apps is a *moving target* for classifiers due to its dynamic evolution and mix. Thus mobile TC constitutes an open and evolving research field.

In this paper, we aim to improve the classification performance of mobile apps by adopting a *multi-classification system* (MCS), that is intelligently-combining decisions from state-of-art classifiers specifically devised for mobile- and ET classification and currently considered the best approaches in such context [3, 4, 6]. To the best of authors' knowledge, this investigation is performed in the mobile context for the first time. The MCS framework has the potential of overcoming the deficiencies of each single classifier (not improvable over a certain bound, despite efforts in careful "tuning") and providing improved performance w.r.t. any of the base classifiers, also allowing for *modularity* of classifiers selection in the pool. For this reason, research has focused on MCSs in the last years [7, 8, 9, 10]. Based on these considerations, five types of combiners proposed in the literature [11, 12] are here compared, each with different complexity and training set requirements. Based on a dataset collected by a global mobile solutions provider[1] of true users' activity, our preliminary results show that MCS framework can improve classification performance with respect to the best base classifiers considered for the task. Specifically, F-measure can be improved by more than $+8\%$ on the best base classifier, thus showing promising results. Our results show that the presented approach is a viable path to improve classification of mobile apps traffic and that there is room for further research on combining algorithms applied to this goal (with evidence of over $+23\%$ improvement achievable by the ideal combiner).

The paper is organized as follows. Sec. II discusses related literature; Sec. III describes the considered MCS framework for mobile TC; experimental results are reported in Sec. IV; finally, Sec. V provides conclusions and future directions.

[1]Due to NDA with the provider we can not report its name, details of its network, detailed information on the data set, nor release the data set.

II. State-of-art of techniques for mobile apps traffic classification

TC of mobile apps has been object of huge interest by several recent works, mainly based on ET assumption. Stöber et al. [13] propose a scheme for fingerprinting devices by learning their traffic patterns through background activities. Based on 3G transmissions, *bursts* of data are considered to evaluate statistical features. Then, using Support Vector Classifier (SVC) and K-Nearest Neighbors, a model of the traffic to be fingerprinted is built, being capable of identifying similar bursts. Results show that using ≈ 15 minutes of traffic testing (based on 6 hours of training) leads to an accuracy $\geq 90\%$. Wang et al. [14] propose a system for classifying app usage over encrypted 802.11 traffic (reporting results for 13 iOS apps from 8 distinct categories). Data frames are collected from target apps by running them dynamically for 5 minutes and training a Random Forest (RF) classifier with the proposed set of features. The need for an accurate ground-truth labeling is raised, highlighted by a counterintuitive behavior of some app performance with the training time. *AppScanner* is proposed in [6] as a framework for fingerprinting and identification of mobile apps. The fingerprints are collected by running apps *automatically* on an Android device and the network traces are pre-processed (to remove background traffic and extract features) to train an SVC and an RF. Statistical features are collected on sets of packets defined through timing criteria and destination IP address/port (see Sec. III-A). The results, evaluated on 110 most popular apps from Google Play Store, report 99% average accuracy in identifying single apps, and up to 86.9% in classifying them, *outperforming* state-of-art alternatives devised for the (conceptually-)similar website fingerprinting issue [3, 4]. These latter methods are used also by Alan and Kaur [15] to investigate whether Android apps can be identified from their launch-time traffic using only TCP/IP headers (i.e. sizes of the first 64 packets). They find that apps can be identified with 88% accuracy when training and test sets are collected on the same device. On the other hand, accuracy drops significantly (up to 26% for the best classifier) when the OS/vendor is different. State-of-art approaches [6, 16, 17] considered in this work outperform those analyzed in [15] also in terms of other performance metrics (see Sec. IV).

Other works aimed at identifying fine-grained user actions within mobile app traffic. Conti et al. [18] recognizes specific actions that users perform while running a certain app, based on packet direction/size info. This is achieved through *service burst* (see Sec. III-A) classification via RF approach, leading to $\geq 95\%$ accuracy for most of the considered actions within a set of 7 Android apps. *Netscope* [5] performs a similar task taking into account a set of 35 different activities (for both iOS and Android devices), based on statistics originated from IP headers. Assuming an eavesdropper on a Wi-Fi network, it is shown that even a small portion of ET is enough for a given app to be recognized. K-means clustering is employed for elementary-behavior discovery and then an SVC is trained/tested on activity-behaviors binary mapping, showing performance that vary with the device being tested, but reach 78.04% precision and 76.04% recall, on average.

Although not focused on mobile apps (though readily adaptable to this context), the work in [16] proposes a ML-based multi-level framework to identify services running within HTTPS connections without relying on specific header fields prone to alteration. The evaluation, based on real traffic, shows high identifiability of encrypted web services. Finally, Tongaonkar [19] reviews challenges and techniques for mobile TC and app identification. The presented approaches are mainly based on signature generation and fingerprint extraction from mobile traffic payloads and apps' metadata, as well as from 3rd-party services (e.g., advertisement and profiling traffic). Nevertheless, the problem of dissecting ET is there *bypassed* by considering man-in-the-middle solutions, suitable only in controlled environments such as enterprises.

III. Traffic Classification at work

In this section we introduce terms and concepts regarding traffic objects, define the features extracted from observed traffic and adopted for classification, describe the classification algorithms (considered as base classifiers) and the fusion techniques adopted for their combination.

A. Traffic View

A common TC object is the *biflow*, defined as a sequence of packets sharing the values of the 5-ple *(transport protocol, source IP address, source transport port, destination IP address, destination transport port)*, where source and destination can be swapped [1]. On the other hand, in this paper, network traffic is decomposed into *service bursts*, leveraging the notions introduced in [13] and [6, 18] for mobile-phone identification and mobile-app classification, respectively. More specifically, a *burst* [13] is defined as a sequence of packets having an inter-packet time smaller than a given threshold (named *burst threshold*, here set to 1 second as suggested in [6]), irrespective of their source or destination addresses, as well as of the biflow they belong to. Accordingly, a *service burst* (SB) is then a set of packets, within a single burst, that belongs to biflows sharing the same transport protocol, destination IP address, and destination port number.[2] We remark that this notion has been used previously in [6, 18] under the name of *flow*. However, in order to avoid any ambiguity with the common and established definition of flow [1], we will refer to the decomposition used in [6, 18] as a SB.

B. Statistical Features

For the purpose of traffic classification, we will consider features which are either related or extracted (by statistical means) from the vector of packet lengths. For each SB, three packet series are considered: (i) *incoming* packets only, (ii) *outgoing* packets only, and (iii) *bidirectional* traffic.

[2]The direction of a biflow is defined according to its first packet: the packet source (destination) is chosen as source (destination) for the whole biflow. Criteria and heuristics for biflow start and end can be defined for both TCP and UDP, and in general a TCP biflow does *not* necessarily match with a TCP session (see [1]).

The following features can be identified for each of these series [6]:

- vector of packet lengths with sign indicating direction;
- minimum, maximum, mean, median, absolute deviation, standard deviation, variance, skew, and kurtosis;
- percentiles (from 10% to 90%, with 10% increments).

Additionally, for the incoming and outgoing packet series taken as a whole, the joint histogram of packet lengths in both directions can be considered [3, 4].

Finally, in the following, the set of M features adopted by each classifier will be generically indicated with $f_1, \ldots, f_M$ (or collectively as $\boldsymbol{f} \triangleq \begin{bmatrix} f_1 & \cdots & f_M \end{bmatrix}^T$) and the set of classes (apps) as $\Omega \triangleq \{c_1, \ldots c_L\}$.

C. Classification Algorithms adopted as Base Classifiers

In this section we list the state-of-art approaches adopted as *base* classifiers. We briefly describe their main properties and the motivations that guided us to their choice. For all of them we have carefully reproduced the exact implementation and executed them with the same parameters as described in the respective works, to which we refer for further details.

Lib_NB: In Liberatore and Levine [4], two classifiers were proposed, one based on the Jaccard similarity index and another based on the well-known *Naïve Bayes* (NB) learning technique. It was observed that the NB enjoys attractive performance and increased robustness than the Jaccard-based classifier, if IP packets are padded; therefore we consider NB-based approach as a base classifier (Lib_NB). The NB assumes class-conditional independence of the features $\boldsymbol{f}$ (being not the case for real-world problems but working well in practice) and evaluates the probability that an unlabeled test instance represented by $\boldsymbol{f}_T$ belongs to each class c_i, i.e. the posterior probability $P(c_i|\boldsymbol{f}_T)$ through the Bayes' theorem $P(c_i|\boldsymbol{f}_T) \propto P(c_i) \prod_{m=1}^{M} P(f_{T,m}|c_i)$, where "$\propto$" denotes proportionality. Each term $P(c_i)$ denotes the (prior) probability that a generic sample from the dataset will belong to c_i and is estimated from the training set population, while each PDF $P(f_m|c_i)$ is estimated through a (Gaussian-) kernel density estimation. The fine-grained feature there employed is the joint histogram of packet lengths in both incoming and outgoing directions.

Her_Pure, Her_TF, and Her_Cos: The study by Herrmann et al. [3] proposed the use of a *Multinomial NB* (MNB) classifier, adopting the same set of features as Lib_NB [4], but differing in the building assumption. Indeed, the MNB classifier treats the f_ms as frequencies of a certain value of a categorical random variable and compares the sample histogram of each test instance with the aggregated histogram of all training instances per class. Then, the evaluation of the conditional PMF $P(\boldsymbol{f}_T|c_i)$ is different from Lib_NB and equals $P(\boldsymbol{f}_T|c_i) \propto \prod_{m=1}^{M} (\rho_m)^{f_{T,m}}$ where ρ_m denotes the probability of sampling the m^{th} feature. This implementation is referred to as Her_Pure in our analysis. The MNB classifier was also employed in [3] with: (i) Term Frequency (TF) and (ii) Inverse Document Frequency (IDF) transformations (both with or without cosine normalization).[3] The variants adopting TF transformation without and with cosine normalization have been also compared in [6] and are referred in our analysis to as Her_TF and Her_Cos, respectively.

Tay_RF and Tay_SVC: In Taylor et al. [6], four (resp. two) approaches for mobile-app traffic classification (resp. identification) were proposed, leveraging both an SVC and an RF. An SVC is a supervised model that represents the training samples as points in a feature (vector) space. The aim of the training phase is to find a set of hyperplanes, dividing this space, which provides the best class separation. Then, during the classification phase, the SVC simply classifies new points according to the portion of space they fall into. On the other hand, an RF is an ensemble classification method taking advantage of several decision trees built at training time in order to form a stronger classifier. An RF combines the ideas of "bootstrap aggregating" (bagging) and random-feature selection to construct a collection of trees with controlled variance, thus avoiding over-training. In [6], these classifiers were fed with either (i) raw vectors of packet lengths or (ii) statistical features, with the latter approach leading to the best classifier (RF with statistical features). For this reason, we consider both RF (Tay_RF) and SVC (Tay_SVC) based on the set of 40 statistical features extracted and selected as in [6].

CART: Several works performed TC by means of decision trees (e.g., C4.5, C5.0, and their variants) [7, 16, 17]. In this paper, we leverage *Classification And Regression Tree* (CART), a very similar variant of the C4.5 algorithm, constructing binary trees exploiting the features and thresholds that ensure the maximum information gain at each node and allowing to perform both classification and regression tasks (i.e. with categorical and numerical target variables, respectively).

D. Classifier Fusion Techniques

Different classifier fusion rules have been proposed in the literature [7, 11]. In this study, we will focus on those based on *Type 1 classifiers* (i.e. those that output only the predicted class), implying the least requirements for designers [11].

Before proceeding, we recall the definition of confusion matrix [11]. Specifically, the $(i,j)^{th}$ entry of the confusion matrix of k^{th} classifier $\boldsymbol{E}^k$, denoted with $e^K_{i,j}$, represents $P(\widehat{d}_k = c_j|c_i)$, i.e. the probability of the k^{th} classifier of deciding for j^{th} class when the i^{th} class is being observed. Clearly, the matrices $\boldsymbol{E}_k$ (and the priors $P(c_i)$ employed by combiners) are typically estimated using a set of data (namely, a validation set) different from both the training and the test sets. In this study, we will consider the following combiners (whose decision will be generically denoted with $\hat{d}_0$):

1) *Majority Voting (MV)*: the guess class is the one voted by the relative majority of the classifiers.[4]

[3]Exploiting the raw occurrence frequencies, the decisions of the MNB classifier are biased towards classes which contain many packets and/or packets with high frequencies. These issues are alleviated by means of the TF and IDF transformation, respectively [3].

[4]Ties are broken by using e^k_{ii}, i.e. the vote of each classifier is weighted by the number representing the confidence degree of that classifier when it assigns a sample to the class it is voting for [7].

2) *Weighted Majority Voting (WMV)*: the vote of each classifier is weighted by its relative confidence. The combiner output is $\hat{d}_0 \triangleq \arg\max_{i\in\Omega}\left\{ f_i + \sum_{k\in I_+^i} w_k \right\}$, where I_+^i denotes the subset of classifiers having decided for i^{th} class, $f_i \triangleq \left[\log P(c_i) + |I_+^i| \cdot \log(L-1)\right]$, and $w_k \triangleq \log(p_k/(1-p_k))$, where p_k denotes the (estimated) accuracy of k^{th} classifier [12].
3) *Naïve Bayes (NB)*: the guess class is the one which maximizes the *a posteriori* probability $P(c_i|\hat{d}_1,\ldots,\hat{d}_K)$ based on conditional independence of classifiers, that is $\hat{d}_0 \triangleq \arg\max_{i\in\Omega} P(c_i)\left\{\prod_{k=1}^{K} P(\hat{d}_k|c_i)\right\}$.
4) *Behavior-Knowledge Space method (BKS)*: this approach removes the conditional independence assumption of NB combiner via multinomial counting on the joint classifiers' space $\hat{d}_1,\ldots,\hat{d}_K$ [20]. More specifically, the validation set is used to estimate the *a posteriori* probability $P(c_i|\hat{\boldsymbol{d}})$ for each c_i and for *each value* of $\hat{\boldsymbol{d}}$ (the space complexity is thus $\mathcal{O}(L^K)$, which requires a large validation set for training). This in turn allows labeling each possible value of $\hat{\boldsymbol{d}}$ with the most likely class (according to $\arg\max_i P(c_i|\hat{\boldsymbol{d}})$) and constructing a look-up (BKS) table. Then, during the testing phase, each new $\hat{\boldsymbol{d}}_T$ provides an index to retrieve from BKS table the estimated class $\hat{d}_0$.[5]
5) *WERnecke's method (WER)*: WER constructs the same table as BKS but, to reduce over-fitting, considers the 95% confidence intervals of the frequencies in each unit [11]. If there is overlap among the intervals, there is no dominating class for labeling the test instance $\hat{\boldsymbol{d}}_T$. In this case, the "least wrong" classifier among the K members of the pool is identified (based on confusion matrices) and authorized to assign the class to that unit.[6]

For a complete comparison, we will consider also an *ORAcle combiner* (ORA), i.e. an ideal upper bound on the performance corresponding to a combiner correctly classifying a test sample if *at least one* of the base classifiers provides the correct decision.

IV. Experimental Results

A. Dataset Description

The proposed MCS is tested on real-traffic traces, provided by an international mobile solutions provider and generated from a total of 49 apps (resp. 45) on Android (resp. iOS) devices, run *separately*. Accordingly, ground truth is obtained by labeling each trace with the generating application. Moreover, these traces have been provided already anonymized and cleaned from background traffic, in order to reflect only relevant application traffic.[7] Network traffic is processed using

[5]Ties are resolved by using a MV (with random tie-breaking) between the elements of $\hat{\boldsymbol{d}}_T$ [11].

[6]To calculate the 95% confidence intervals we adopt the normal approximation of the Binomial distribution [11].

[7]Furthermore, we underline that we do not cleanse traffic traces from TCP retransmissions, as we have observed that this does not affect classification performance in a substantial way.

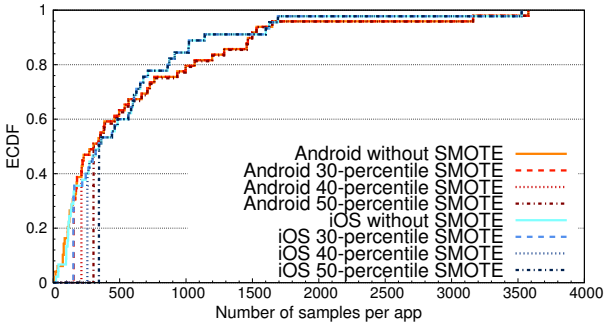


Figure 1: ECDFs of the number of samples per app for Android and iOS datasets before and after SMOTE application with different thresholds.

the approach introduced in Sec. III-B. The minimum SB length considered in this paper is 7 (as suggested in [6]), since it is the shortest sequence of packets representing a meaningful data transfer which includes a TCP handshake and an HTTP request/response with corresponding ACKs. Then, after *burstification*, we obtained about 29670 (resp. 24714) labeled SBs composing our dataset.

Additionally, due to severe imbalance in the number of instances for each app (this is especially true for the least observed ones, see Fig. 1, reporting the empirical CDF of the number of SBs per app), an oversampling procedure has been applied to the dataset. More specifically, we applied the *Synthetic Minority Oversampling TEchnique* (SMOTE) [21] to apps with a number of SBs less than 30^{th} percentile of the distribution in Fig. 1, in order to obtain a reasonable number of samples per app. SMOTE is one of the most popular approaches for data-based class-minority oversampling. We remark also that the results obtained with different percentages of SMOTE (e.g., corresponding to the 40^{th} and 50^{th} percentiles) have shown no discrepant relative performance among the classifiers/combiners considered in what follows, thus underlining the *stability* of the considered dataset. Finally, after applying SMOTE, we obtained 30680 (resp. 25465) SBs composing our dataset, with the least populated classes composed by 155 (resp. 152) SBs.

B. Classification Results

In this section, we show results pertaining to experiments on the dataset described in Sec. IV-A, obtained by application of the proposed MCS (see Sec. III). Our comparison will be based on the following performance measures [11]: overall accuracy, precision, and recall. Since the latter two are defined on a per-app (per-class) basis, we will employ their arithmetically averaged (viz. macro) versions. Additionally, we will consider the *F-measure* ($\mathrm{F} \triangleq (2\cdot \mathrm{prec}\cdot \mathrm{rec})/(\mathrm{prec}+\mathrm{rec})$), so as to account for both the effects of precision (`prec`) and recall (`rec`) in a concise fashion. Furthermore, we will also consider confusion matrices of classifiers (resp. combiners) to provide their complete performance "picture" and identify the most frequent misclassification patterns. Finally, the considered setup will employ a random training-validation-test set splitting (with corresponding percentages $50\%-25\%-25\%$).

Table I: Performance of state-of-art classifiers considering Android (iOS) traffic.

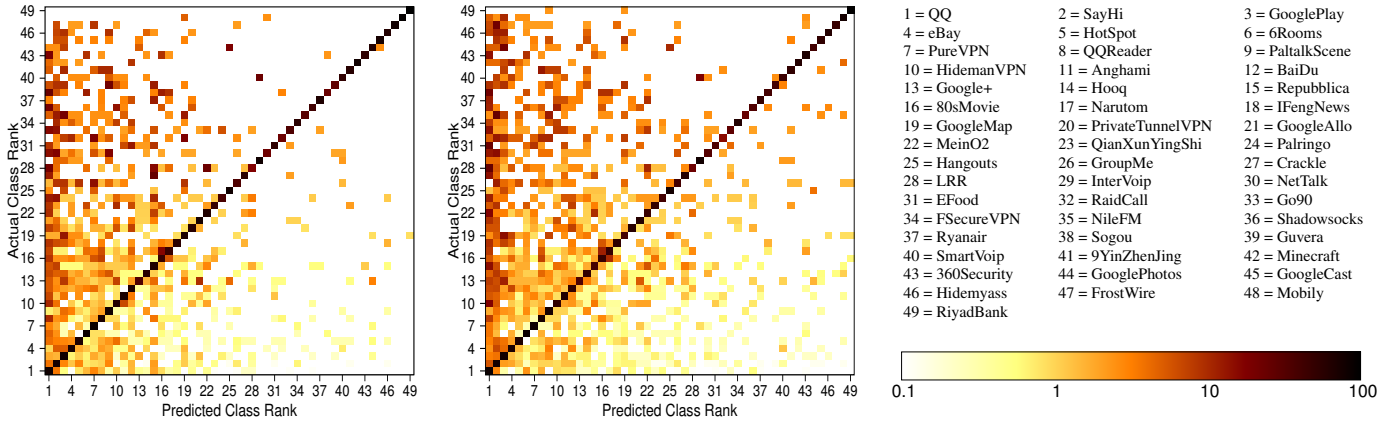
Classifier	Her_Pure	Her_TF	Her_Cos	Lib_NB	Tay_RF	Tay_SVC	CART	ORA
Accuracy	48.2 (45.8)	58.8 (56.0)	59.1 (58.9)	31.1 (29.8)	**64.5 (62.2)**	28.5 (26.5)	51.0 (45.7)	83.5 (82.3)
Macro Precision	48.2 (48.3)	73.5 (**70.0**)	**74.3** (69.0)	61.5 (50.5)	64.2 (63.0)	35.1 (38.1)	41.3 (36.8)	-
Macro Recall	47.9 (44.3)	46.1 (42.1)	45.7 (45.2)	36.2 (31.9)	**54.0 (51.3)**	13.2 (12.7)	40.0 (36.5)	76.9 (75.1)
Macro F-Measure	48.2 (47.5)	65.7 (61.8)	**66.1 (62.4)**	53.9 (45.2)	61.8 (60.3)	26.4 (27.2)	41.0 (36.8)	-

Table II: Performance of combiners and *Maximum Improvement Over Best Classifier* (MIOBC), considering Android (iOS) traffic.

Combiner	MV	WMV	NB	BKS	WER	ORA	*MIOBC*
Accuracy	63.0 (61.3)	63.5 (61.3)	**67.9 (64.8)**	67.7 (64.5)	65.2 (62.6)	83.5 (82.3)	+3.4 (+2.6)
Macro Precision	**80.2** (75.6)	79.9 (**75.8**)	72.5 (70.1)	74.0 (69.1)	64.6 (62.7)	-	+5.9 (+5.8)
Macro Recall	51.6 (48.4)	51.4 (47.7)	**59.3 (55.6)**	57.6 (53.5)	54.7 (52.0)	76.9 (75.1)	+5.3 (+4.3)
Macro F-Measure	**72.2 (68.0)**	71.9 (67.9)	69.4 (66.6)	70.0 (65.3)	62.3 (60.2)	-	+6.1 (+5.6)

Table III: F-measure of combiners and *Maximum Improvement Over Best Classifier* (MIOBC), as function of the pool of selected classifiers considering Android (iOS) traffic. Highlighted values: **maximum per pool**, *maximum per combiner*, overall maximum.

Pool of classifiers							**Combiners**					
Her_Pure	Her_TF	Her_Cos	Lib_NB	Tay_RF	Tay_SVC	CART	MV	WMV	NB	BKS	WER	*MIOBC*
✓	✓	✓	✓	✓	✓	✓	**72.2 (68.0)**	71.9 (67.9)	69.4 (66.6)	*70.1 (65.1)*	62.3 (60.2)	+6.1 (+5.6)
✓	✓	✓	✓	✓		✓	**71.6** (66.6)	71.5 (67.1)	69.6 (**67.4**)	68.7 (64.0)	62.7 (59.9)	+5.5 (+5.0)
✓	✓	✓	✓	✓			69.6 (65.2)	69.5 (66.0)	**70.0 (66.6)**	66.7 (62.8)	*63.6* (61.1)	+3.9 (+4.2)
	✓	✓	✓	✓			70.5 (66.3)	70.5 (65.9)	***71.7* (68.0)**	65.0 (59.9)	63.0 (61.1)	+5.6 (+5.6)
		✓	✓	✓			*72.8 (69.6)*	*72.1 (70.5)*	70.0 *(68.8)*	63.3 (57.3)	63.0 *(61.2)*	+6.7 (+8.1)
		✓		✓			67.0 (65.1)	67.2 (64.2)	**67.9 (65.5)**	64.0 (60.3)	63.1 (60.9)	+1.8 (+3.1)



(a) Best combiner (NB). (b) Best base classifier (Tay_RF). (c) Scale (%) and labels.

Figure 2: Confusion matrices of best combiner (a) and best classifier (b). Note that the labels (c) are ranked according to decreasing abundance of samples and the logarithmic scale (c) is used to evidence small errors.

First, in Tab. I, we report the performance of all the classifiers considered in Sec. III-C, in terms of the these measures. Also, for completeness, we report the accuracy and recall achieved by the ORA.[8] First Tay_RF, Her_Cos and Her_TF achieve the highest performance w.r.t. the considered measures, thus qualitatively agreeing with the results in [6]. Nonetheless, as apparent from ORA results, the best accuracy (resp. recall) of the base classifiers can be virtually improved by means of the proposed MCS up to 19.0% (resp. 22.9%) for Android and up to 20.1% (resp. 23.8%) for iOS, respectively. These results confirm the appeal of adopted fusion techniques.

To this end, in Tab. II we show (and compare) the performance of the considered 5 combiners. Additionally, to assess the improvement achieved by combining techniques we have reported in the column *MIOBC* the maximum improvement over the best classifier for each metric. Overall, the set of combiners is *always* able to provide an improvement, ranging from +2.6% (accuracy on iOS traffic) to (+6.1%) (F-measure on Android traffic), by means of diversity principle, representing the milestone for adoption of classifier fusion techniques. However, different approaches result best according to different performance metrics. Specifically, NB is able to provide a good improvement with respect to the best base classifier for accuracy and recall. Differently, MV and WMV result appealing because of the remarkable improvement in terms of precision and F-measure over the best base classifier (between +5.6% and +6.1%). Interestingly, MV, WMV, and NB collectively provide the highest performance; this is explained as they are less prone to over-fitting (and have less training requirements) and, last but not least, they also enjoy lower complexity (w.r.t.

[8]Precision (and consequently F-measure) for the ORA cannot be evaluated since its error patterns are not defined [7].

WER and BKS). The comparison of confusion matrices of NB (Fig. 2a) and Tay_RF (Fig. 2b) also reveals a homogeneously-reduced occurrence of misclassification patterns.

Finally, in Tab. III we preliminarily investigate the effect of subset selection on the F-measure of the 5 combiners to improve performance further (possibly reducing complexity). Having different optimization criteria (combiners), we adopt an heuristic approach informed by diversity of classification methods and iteratively removing the worst performing classifier. From inspection of results, it is apparent that subsets considered are helpful in (slightly) improving the performance of all combiners (and, equally important, to reduce complexity), by a judicious selection of the base classifiers, except for BKS, which does not generally benefit from subset selection. Interestingly, the best performance is achieved by combiners which employ a reduced set of base classifiers.

V. Conclusions and Future Directions

We tackled TC of mobile apps by means of a MCS, employing 5 well-known combining methods in conjunction with a pool of 7 state-of-art classifiers specific or suitable for mobile traffic. The evaluation of combining techniques has been performed on an actual dataset describing traffic from 49 (resp. 45) apps in Android (resp. iOS) devices. Results have shown improvements of the performance by MCS over the best base classifier up to 8.1% in terms of F-measure. Also the ORA and the (preliminary) subset selection performance have underlined that there is room for further improvement toward optimal (low-complexity) combination of the base classifiers. Future directions will include: (i) a deeper analysis with an enlarged pool, (ii) advanced fusion techniques possibly based on Type 2-3 (rank- and measurement-level) classifiers [11], (iii) an intelligent pool subset selection, (iv) the evaluation of the sampling impact [22], and (v) the implementation of classifiers and combination techniques in TIE [8].

References

[1] A. Dainotti, A. Pescapé, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, 2012.

[2] G. Aceto, A. Dainotti, W. De Donato, and A. Pescapé, "PortLoad: taking the best of two worlds in traffic classification," in *IEEE INFOCOM'10*, pp. 1–5.

[3] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial Naïve-Bayes classifier," in *Proc. of the ACM CCSW'09*, pp. 31–42.

[4] M. Liberatore and B. N. Levine, "Inferring the source of encrypted HTTP connections," in *Proc. of the 13th ACM CCS'09*, pp. 255–263.

[5] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian, "Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic," in *Proc. USENIX WOOT'16*.

[6] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *IEEE EuroS&P'16*, pp. 439–454.

[7] A. Dainotti, A. Pescapé, and C. Sansone, "Early classification of network traffic through multi-classification," in *TMA'11*. Springer, pp. 122–135.

[8] W. De Donato, A. Pescapé, and A. Dainotti, "Traffic identification engine: an open platform for traffic classification," *IEEE Network*, vol. 28, no. 2, pp. 56–64, 2014.

[9] H. He, C. Che, F. Ma, J. Zhang, and X. Luo, "Traffic classification using en-semble learning and co-training," in *WSEAS Proc. of the 8th AIC'08*, pp. 458–463.

[10] G. Szabo, I. Szabo, and D. Orincsay, "Accurate traffic classification," in *IEEE WoWMoM'07*, pp. 1–8.

[11] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[12] L. I. Kuncheva and J. J. Rodríguez, "A weighted voting framework for classifiers ensembles," *Knowledge and Information Systems*, vol. 38, no. 2, pp. 259–275, 2014.

[13] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic, "Who do you sync you are? smartphone fingerprinting via application behaviour," in *Proc. of the 6th ACM WISEC'13*, pp. 7–12.

[14] Q. Wang, A. Yahyavi, B. Kemme, and W. He, "I know what you did on your smartphone: Inferring app usage over encrypted data traffic," in *IEEE CNS'15*.

[15] H. F. Alan and J. Kaur, "Can android applications be identified using only tcp/ip headers of their launch time traffic?" in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 2016, pp. 61–66.

[16] W. M. Shbair, T. Cholez, J. Francois, and I. Chrisment, "A multi-level framework to identify HTTPS services," in *IEEE/IFIP NOMS'16*, April, pp. 240–248.

[17] T. Bakhshi and B. Ghita, "On internet traffic classification: A two-phased machine learning approach," *Journal of Computer Networks and Communications*, 2016.

[18] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 1, pp. 114–125, 2016.

[19] A. Tongaonkar, "A look at the mobile app identification landscape," *IEEE Internet Computing*, vol. 20, no. 4, pp. 9–15, July 2016.

[20] Y. S. Huang and C. Y. Suen, "A method of combining multiple experts for the recognition of unconstrained handwritten numerals," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 1, pp. 90–94, 1995.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[22] D. Tammaro, S. Valenti, D. Rossi, and A. Pescapé, "Exploiting packet-sampling measurements for traffic characterization and classification," *International Journal of Network Management*, vol. 22, no. 6, 2012.