MIMETIC: Mobile Encrypted Traffic Classification using Multimodal Deep Learning

Giuseppe Aceto[a,b], Domenico Ciuonzo[a], Antonio Montieri[a], Antonio Pescapé[a,b]

[a] *University of Napoli "Federico II", Italy*
[b] *Network Measurement and Monitoring (NM2) s.r.l., Italy*

Abstract

Mobile Traffic Classification (TC) has become nowadays the enabler for valuable profiling information, other than being the workhorse for service differentiation or blocking. Nonetheless, a main hindrance in the design of accurate classifiers is the adoption of encrypted protocols, compromising the effectiveness of deep packet inspection. Also, the evolving nature of mobile network traffic makes solutions with Machine Learning (ML), based on manually- and expert-originated features, unable to keep its pace. These limitations clear the way to Deep Learning (DL) as a viable strategy to design traffic classifiers based on automatically-extracted features, reflecting the complex patterns distilled from the multifaceted traffic nature, implicitly carrying information in "multimodal" fashion. Multi-modality in TC allows to inspect the traffic from complementary views, thus providing an effective solution to the mobile scenario. Accordingly, a novel multimodal DL framework for encrypted TC is proposed, named MIMETIC, able to capitalize traffic data heterogeneity (by learning both intra- and inter-modality dependences), overcome performance limitations of existing (myopic) single-modality DL-based TC proposals, and support the challenging mobile scenario. Using three (human-generated) datasets of mobile encrypted traffic, we demonstrate performance improvement of MIMETIC over (*a*) single-modality DL-based counterparts, (*b*) state-of-the-art ML-based (mobile) traffic classifiers, and (*c*) classifier fusion techniques.

Keywords: traffic classification; mobile apps; Android apps; iOS apps; encrypted traffic; deep learning; automatic feature extraction; multimodal learning.

1. Introduction

Thee efficacy of security and quality-of-service enforcement devices, as well as network monitors, is limited (or qualitatively hampered) when there is no accurate knowledge of the application generating the traffic. The process inferring such information, known as network Traffic Classification (TC), has a long-standing application in many fields [1] and is facing unprecedented challenges due to the users massive shift toward mobile devices (as witnessed by recent Internet traffic evaluations [2]), leading to a multifaceted and evolving composition of network traffic [3].

Hence, the appeal of mobile TC has bloomed nowadays, nurtured (other than usual TC drivers, e.g. service differentiation) by valuable profiling information (e.g., to advertisers, security agencies, and insurance companies), while also implying privacy downsides (e.g., recognition of context-sensitive applications, such as dating and health ones, and bring-your-own-devices policies). The effort towards the protection of privacy and security has fueled the widespread adoption of encrypted protocols (TLS). This shift, together with the use of dynamic transport ports or the clustering on a few well-known (and commonly unblocked) ports, resulted in the hampering of accurate TC, as both Packet Inspection (DPI) and port-based techniques become ineffective [1]. As a consequence these traditional approaches remain effective only in closed-world (e.g. enterprise) scenarios via application-level firewalls (analogous to man-in-the-middle attacks) [4]. Also, achieving targeted TC performance in mobile-traffic context is undermined by the presence of several (similar) apps to discriminate from and a scarce number of training samples per app.

In this context, Machine Learning (ML) classifiers have proved to be a good fit, since they suit also *encrypted traffic* while not expressly relying on port information [5, 6, 7, 8]. However, their usual form resorts to the process of obtaining handcrafted (domain-expert driven) features (e.g. packet sequence statistics), which is time-consuming, unsuited to automation, and it is unable to keep the pace of network traffic evolution. Therefore, Deep Learning (DL) is emerging as the stepping stone toward the fulfillment of high performance in the dynamic and challenging (encrypted) TC contexts, allowing to train classifiers directly from input data by *automatically distilling* structured (and complex) feature representations [9]. Accordingly, several works recently appeared tackling TC via DL [10, 11, 12, 13, 14, 15]. However, DL benefits should not be taken for granted and its naïve adoption to encrypted TC has been shown to imply misleading design choices and lead to *biased* conclusions, due to the peculiar (and tricky) nature of network traffic data [16]. Last but not least, the nature of traffic data is *heterogeneous* and its whole capitalization is yet to be achieved.

Indeed, most of these DL-based efforts have focused on one

Email addresses: giuseppe.aceto@unina.it (Giuseppe Aceto), domenico.ciuonzo@unina.it (Domenico Ciuonzo), antonio.montieri@unina.it (Antonio Montieri), pescape@unina.it (Antonio Pescapé)

type of input information (e.g. payload bytes or header fields), despite traffic data being naturally "multimodal"; this means that the same concept can be described by different data types (known as "views" or "modalities"). Hence, the main asset of multimodal DL is the ability to automatically learn a hierarchical representation exploiting jointly all the available modalities, instead of handcrafting modality-specific features for a given ML approach [17].

Summary of the Contributions

Based on the aforementioned motivations, the main contributions of this work are summarized as follows:

- We address the multi-view capitalization of traffic data via a novel **MultImodal DL-based MobilE TraffIc Classification (MIMETIC)** framework, having the capability of exploiting effectively the heterogeneous nature of the different views of a TC object, by capturing both intra- and inter-modalities dependence. Although the adoption of multimodal DL is obtaining a growing and wider interest in the scientific literature [17, 18, 19], no such approach has been proposed in (mobile) TC literature to date, up to our knowledge.

- Since the capitalization of multi-modality in DL architectures is *far from trivial* [19], it requires a thorough design which cannot ignore expertise from network traffic monitoring (e.g. the definition of a set of *unbiased* input views [16, 20]). Hence, MIMETIC approach is carefully defined herein in terms of (*i*) the general architecture and (*ii*) proposed training procedure.

- Our MIMETIC approach is compared with single-modality DL-based and state-of-the-art ML-based traffic classifiers. Such detailed comparison is performed (*a*) through the systematic performance evaluation groundwork provided in [16, 20] and (*b*) based on three datasets collected by *human users*, so as to draw close-to-general take-aways.

- Experimental results highlight a performance improvement of a MIMETIC instance (in terms of both concise and fine-grained measures) while reporting a lower training time (*more than three times*) with respect to existing (single-modality) DL-based traffic classifiers. Specifically, the proposed implementation outperforms the best baseline up to +8.58% in terms of F-measure (i.e. 82.99% when classifying traffic generated by iOS apps). The improvement is also observed with respect to classifier fusion [21] of best single-modality DL baselines, also unexplored to date.

- Finally, MIMETIC is enriched (to provide it with a finer performance control) with the option of censoring some classifications, while allowing to label only traffic aggregates for which the multimodal architecture emits a sure verdict (i.e. a "reject option"). Corresponding results report very high performance with a moderate (controllable) number of unclassified instances.

Table 1: List of the acronyms used in the manuscript.

Acronym	**Definition**
CNN	Convolutional Neural Network
CR	Classified Ratio
DL / ML	Deep/Machine Learning
FB / FBM	FaceBook / FB Messenger
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MIOB-C	Maximum Improvement Over Best - Classifier
MIOB-FT	Maximum Improvement Over Best - Fusion Technique
MLP	MultiLayer Perceptron
MV	Majority Voting
RF	Random Forest
RTPE	Run Time Per-Epoch
SOA	Soft-Output Average
TC	Traffic Classification
TLF	Trainable Late Fusion

We remark that the present study strikes a significant difference with respect to our recent work [16] (and its extended version [20]), wherein (*a*) a systematic dissection of "practical" *single-modality* DL-based TC approaches was put forward and (*b*) a performance evaluation benchmark was proposed to enable TC designers with a set of useful performance tools at the analysis stage. In this respect, the present work provides a constructive design-oriented contribution (namely, the MIMETIC framework). The unbiased single-modality DL baselines and the performance evaluation framework individuated and defined in [16, 20], respectively, are merely exploited in this work to show the appeal of MIMETIC from different standpoints.

The rest of the paper is organized as follows. Sec. 2 contains a literature background of (mobile) TC (including most relevant DL works), whereas Sec. 3 describes MIMETIC framework; the experimental setup and evaluation are given in Secs. 4 and 5, respectively; finally, Sec. 6 provides conclusions and future directions. Also, Tab. 1 summarizes the acronyms used in the text for readability.

2. Related Works

Various works have recently tackled mobile TC, mostly in presence of encrypted traffic and via usual ML techniques. Hence, ML-based approaches in mobile TC are described first (Sec. 2.1). Then DL-based Internet TC (i.e. mostly in *non-mobile scenarios* and resorting only to a *single-modality*) is discussed (Sec. 2.2). A wrap-up discussion, highlighting the limitations of current literature, ends the section (Sec. 2.3).

2.1. Mobile TC via standard ML

Stöber et al. [22] devise a *device-fingerprinting scheme* by learning traffic patterns of background activities. An accuracy $\geq$ 90% is obtained (among 20 users with different combinations of apps installed on Android OS only) using a Support Vector Classifier and K-Nearest Neighbors fed with statistical features from 3G data bursts. Differently, Wang et al. [23] investigate *app-usage classification* (among 13 iOS apps within 8 different categories) via a Random Forest (RF) classifier, whose features are extracted from Wi-Fi encrypted traffic. Results show some counterintuitive behavior with increasing training time, highlighting the impact of inaccurate ground-truth on classification performance.

Taylor et al. [7] propose *AppScanner*, a framework based on a ML classifier (namely, RF) to identify smartphone apps using packet sizes and directions (accessible also from encrypted traffic). Based on *bot-generated* traffic from the 110 most popular Android apps and considering also variation of app versions, employed devices, and fingerprint aging, results show app re-identification with up to 96% accuracy in the best case, outperforming baselines taken from website fingerprinting, with good tolerance to fingerprints aging. The same methods employed for website fingerprinting are also adopted in [24] to check out whether Android apps can be identified from their launch-time *(bot-generated)* traffic using payload sizes of the first 64 packets. Therein, the best classifier achieves 88% accuracy when training and testing are performed on the same device, with a drop to −26% when the OS version or vendor is different. Fu et al. [6] propose *CUMMA* approach to classify (and detect anomalous) service usages in mobile messaging apps, based on RF, hidden Markov models and clustering. Results, based on *human-generated data from 15 volunteers* using Whatsapp and WeChat, report ≥ 96% accuracy for both apps.

A two-tier hierarchical TC framework is proposed in [25] to identify services running within HTTPS connections. Statistics on inter-arrival times and packet/payload sizes are used to train/test C4.5 and RF classifiers. The evaluation, via real traffic traces, shows a recall within $[95, 100]\%$ in 50 out of the 68 HTTPS services considered. Recently, in [21] a *multi-classification approach* is devised leveraging state-of-the-art classifiers proposed for mobile (encrypted) TC, considering four combining classes differing in classifiers' outputs used, learning philosophy, and training requirements. Exploiting iOS and Android datasets of *real users' activity*, combination results produce a performance gain (up to +9.5% recall) w.r.t. the best state-of-the-art ML classifier.

2.2. State of the art on TC via DL

A first DL approach applied to *clear traffic identification* (seamlessly applicable also to encrypted traffic) is presented in [10], employing Stacked AutoEncoders and comparing them to standard neural networks. Results show that the Stacked AutoEncoders outperforms the latter and achieves ≥ 90% precision and recall in protocol identification (on 25 most popular protocols), and ≥ 80% class prediction probability on 6.7k out of 10k traffic samples unrecognizable via DPI. On the other hand, Wang et al. [11] propose a method for TC, explicitly devised for encrypted traffic, based on 1D Convolutional Neural Networks (CNNs). Experiments are conducted on a selection of the "ISCX VPN-nonVPN" (non-mobile) dataset [26] and consist of four different setups including VPN/nonVPN (binary) classification, encrypted TC, and TC of VPN-encapsulated data. Input data employed to feed the DL traffic classifier is characterized by the protocol layer ("ALL" vs. "L7") and the TC object ("Flow" vs. "Biflow") considered, with "Biflow + ALL" input combination achieving the best performance. Unfortunately, such design choice led to *biased results* (for a comparison taking into account this issue see [16]). The same dataset is used to test *Deep Packet* [12] and *Datanet* [27], two DL-based encrypted traffic classifiers working at packet-level and adopting a 1D/2D-CNN, a (deep) MLP or a Stacked AutoEncoder. In the former case, Deep Packet achieves an average 95% (resp. 97%) F-measure for the application identification (resp. traffic characterization) task, consisting of 17 applications (resp. 12 activities). In the latter case, Datanet reaches ≥ 96% F-measure for both the Stacked AutoEncoder and 2D-CNN in discriminating among (a subset of) 15 applications. In *both* studies the first 1480 bytes of L2 payload are used as the input, thus leading to *biased* performance.

The work in [28] tackles *malware TC* through a DL-based approach (exploiting both raw data and *handcrafted features*) which uses a "weighted" backpropagation (to deal with the issues of an imbalanced dataset) and adopts hierarchical learning. The proposed approach outperforms standard ML/DL alternatives (i.e. 99.63% accuracy and 85.44% precision on a self-generated dataset), performing real-time TC and unseen malware identification.

Different DL architectures for encrypted TC, based on hybrid compositions of Long Short-Term Memory (LSTM) and 2D-CNN layers, are proposed in [13]. The best-performing of these variants attains an accuracy (resp. F-measure) up to 96.32% (resp. 95.74%) on a dataset captured on Spanish academic backbone network and consisting of ≈ 266k biflows belonging to 108 distinct services. The analysis also highlights (*i*) a performance drop by including inter-arrival times in the input and (*ii*) that $5 \div 15$ packets are enough for satisfying results. As a further innovation, Huang et al. [29] propose a multi-task DL approach (with a 2D-CNN) to simultaneously solve: (*i*) malware (binary) detection, (*ii*) (binary) recognition of VPN-encapsulation, and (*iii*) Trojan classification (9 classes). Devised approach is successfully tested on data assembled from "CTU-13" (malware) and "ISCX VPN-nonVPN" traffic datasets.

Similarly, Chen et al. [30] propose *Seq2Img*, a pipeline made of reproducing kernel Hilbert space embeddings (producing an equivalent image) and a 2D-CNN architecture, suitable for early TC (namely, based on the first 10 packets), where three packet informative fields (e.g. the size difference, the inter-arrival time and the direction) and the server IP address are used as the input. The approach is validated on two self-generated datasets whose traffic is generated by five protocols and five Internet applications, respectively, achieving 99.84% and 88.42% accuracy. In [15] a flow-based TC approach based on a cascade of a RF (designed at flow-level) and a 1D-CNN (operating at packet level) combined with majority voting (to aggregate per-packet classification outcomes at the flow level) is designed for discriminating (encrypted) Google services running over QUIC protocol, based on the first 1400 bytes of its payload. The approach is tested on a *bot-generated* (via web-browser exploration) *non-mobile* dataset, and comprising five services using QUIC.

Differently, in [14] the *Byte Segment Neural Network* architecture, based on LSTM & GRU layers, is proposed for datagram-based classification, based on L4 payload. The experimental analysis, on a self-generated dataset made of 10 classes (protocols and applications), reports a $\geq 90\%$ F-measure for the applications/protocols considered.

Recently, Aceto et al. [16] define a *systematic framework to dissect the encrypted (mobile) TC problem using DL* and compare a number of the aforementioned techniques (including [10, 11, 13]) on three datasets of *real human users' activity*, highlighting their pitfalls, design guidelines, and challenges. Different viewpoints are taken into account, such as: (*i*) the TC object adopted, (*ii*) the type and the amount of input data, (*iii*) the DL architecture employed, and (*iv*) the required set of performance measures for a comprehensive evaluation. The survey of current DL-based traffic classifiers highlights the need for (*a*) input data that are *unbiased* (to avoid experiencing inflated performance), informative, and *heterogeneous* (composed of different complementary views), and (*b*) a rigorous performance evaluation workbench.

2.3. Limitations of Existing Literature

The above literature review highlights the following limitations, which are *addressed by the present work*.

First, all the works specifically dealing with mobile TC mostly either consider iOS- [23] or (bot-generated) Android-traffic [7, 22, 24], as opposed to our work (covering both mobile operating systems with human-generated traffic). Indeed, the different nature and history of the software and hardware ecosystems revolving around the two mobile OSes suggest that app traffic could inherit different properties as well. Consequently, app discrimination ability on one OS cannot be assumed as generalizable to the other. Similar concerns regard bot-generated traffic when extending results to actual (human-generated) mobile app traffic.

Secondly, the analysis of current literature highlights the lack of flexibility (and adaptation) of ML-based approaches in realistic mobile contexts. Moreover, it shows the inability of current DL approaches of consistently outperforming the former in these challenging scenarios [16]. We traced these deficiencies back to *the use of only a single-modality* in their end-to-end design [10, 11, 13, 12, 15]. As a consequence, in contrast to existing DL-based TC literature, *our MIMETIC framework is designed to exploit different modalities (viz. views or inputs) jointly*, and thus to reap DL promised benefits. Precisely, our proposal is shown to provide a performance improvement in the challenging mobile scenario with respect to (ML) state-of-the-art [7], single-mode DL baselines in [11, 13], and even classifier fusion attempts of their outputs, thus proving that the gain is due to our prescribed framework, and not the mere combination of DL algorithms.

Finally, most of the existing DL-based TC approaches are analyzed in terms of per-class or synthetic classification metrics [31, 11, 13, 12, 14], *without* analyzing their performance behavior at a finer-level. Differently, to draw firm conclusions, our performance evaluation resorts to the systematic evaluation framework developed in [16], allowing to compare and assess performance comprehensively, e.g. from complementary viewpoints (i.e. classification performance and complexity) and at different levels of granularity.

3. Description of MIMETIC framework

Herein, we describe the MIMETIC framework, starting from a high-level architectural description in Sec. 3.1. We then focus, in Sec. 3.2, on the general procedure adopted for training it. Both the architectural description and the training procedure are shown in Fig. 1 from a conceptual standpoint. Finally, in Sec. 3.3, we focus on the specific instance of MIMETIC (see Fig. 2) chosen and evaluated in later Sec. 5. Table 2 describes the mathematical notations used in the following.

Table 2: List of the mathematical notations used to define the MIMETIC framework.

Symbol	Definition
M	Number of training samples
M_ℓ	Number of training samples of the ℓ^{th} app
P	Number of different inputs (modalities)
J_p	Number of single-modality layers
$\boldsymbol{x}_{(m)}$	m^{th} sample of the training set
$\ell_{(m)}$	Label (true class) of $\boldsymbol{x}_{(m)}$
$\boldsymbol{t}_{(m)}$	One-hot representation of $\ell_{(m)}$
$\boldsymbol{c}_{(m)}$	Predicted class confidences of $\boldsymbol{x}_{(m)}$
$\mathrm{CE}(\boldsymbol{t}, \boldsymbol{c})$	Categorical cross-entropy between $\boldsymbol{t}$ and $\boldsymbol{c}$
w_m	Weight assigned to $\boldsymbol{x}_{(m)}$
$\boldsymbol{\theta}_p$	Parameters of the p^{th} single-modality layers
$\boldsymbol{\theta}_p^{\uparrow}$	Parameters optimized in pre-training and fine-tuning
$\boldsymbol{\theta}_p^{\downarrow}$	Parameters optimized only in pre-training
$\boldsymbol{\theta}_p^{\text{stub}}$	Parameters of the p^{th} "stub" layer
$\boldsymbol{\theta}_0$	Parameters of the shared representation layers
$\hat{\boldsymbol{\theta}}_p$	Pre-trained parameters of the p^{th} single-modality layers
$\hat{\boldsymbol{\theta}}_p^{\text{stub}}$	Trained parameters of the p^{th} "stub" layer
$\mathcal{L}_p(\cdot)$	Loss function minimized in pre-training of p^{th} modality
$\mathcal{L}(\cdot)$	Loss function minimized in fine-tuning
$\boldsymbol{h}[t]$	State vector of recurrent layers

3.1. Architectural Overview

The mobile TC problem consists in assigning a label among L applications within the set $\{1, \cdots, L\}$ to each TC object [1] observed. Specifically, *traffic object segmentation* defines how raw traffic is segmented into multiple discrete units [1]—a given subset of network packets—that will constitute the object of label assignment. Virtually all works tackling (encrypted) TC using DL considered as classification objects either *flows* or *biflows* (except for [12, 14], working on single packets/datagrams, cf. Sec. 2), with the latter achieving better performance. In detail, a *flow* (resp. *biflow*) is a stream of packets sharing the 5-tuple (i.e. source IP and port, destination IP and port, and transport-level protocol), taking into account (resp. irrespective of) their directions. We mention that other noteworthy choices of TC object—so far adopted in contexts different from mobile encrypted TC—are the *TCP connection* [1] (differing from the biflow only for the termination criterion) and the *service burst*

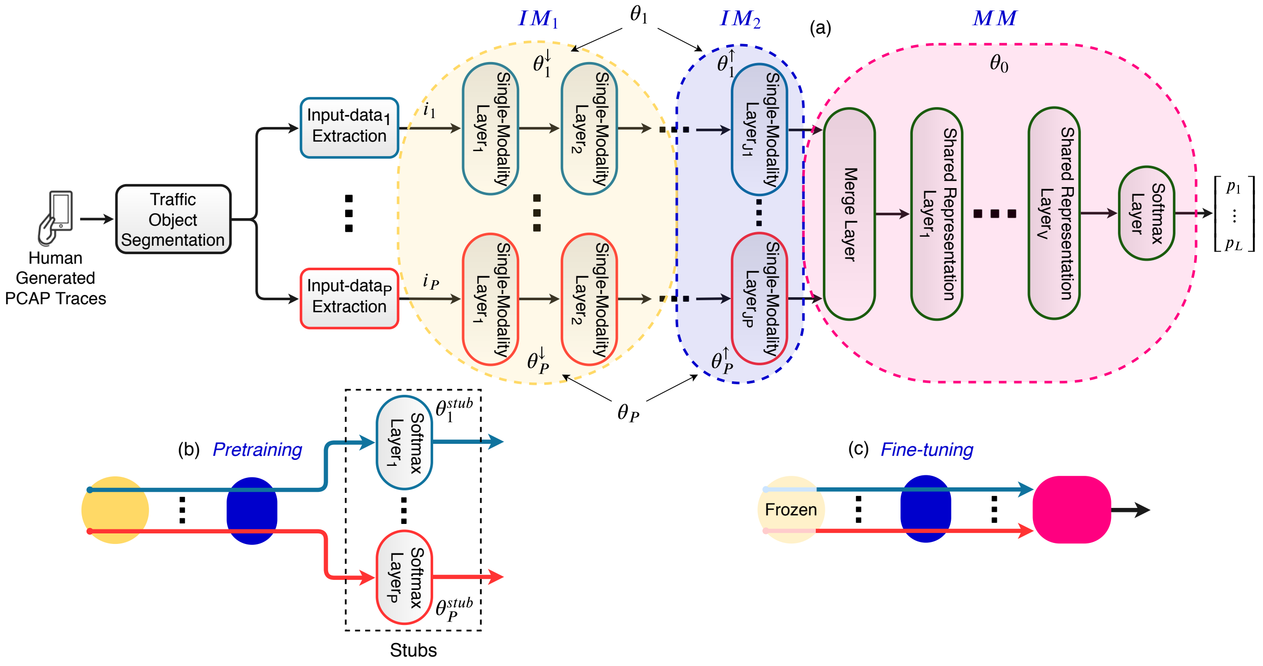


Figure 1: General illustration of MIMETIC framework. (*a*) depicts the architecture by highlighting single-modality representation layers, differentiated as those that are only pre-trained (IM_1) and those that are also fine-tuned (IM_2), and shared representation-layers (MM), along with the corresponding parameter set. (*b*) and (*c*) depict the proposed training procedure based on pre-training and fine-tuning.

[7] (aggregating packets toward the same destination IP & port pair).

We remark that ML-based, as well as DL-based traffic classifiers, rely on a *training set* to learn the distinctive fingerprint of each app. Therefore, for notational convenience, we define the m^{th} TC object of the training set (made of M samples, with M_ℓ being the number of samples belonging to ℓ^{th} app) as $\boldsymbol{x}_{(m)}$ while the corresponding label as $\ell_{(m)}$, belonging to one out of the L different classes considered. As opposed to ML-based TC, DL approaches are able to learn app fingerprints in a *end-to-end* fashion, i.e. *directly* from the type of input selected, thus defeating the tedious and lowly-adaptable process of feature design [16]. However, the traffic data is highly-structured by design, as it contains information referring to the whole protocol stack. As a result, a monolithic DL architecture taking the whole information coming from a TC object in bulk—*early* (or *data*) *fusion*—is likely to be suboptimal, since the parameter set would overfit to one input subset while underfitting the others. Differently, capitalization of score-results (*late* fusion) of DL-based traffic classifiers built on different modalities, although effective in some cases [21], is not able to fully exploit the benefits of multi-modality (as also shown experimentally in Sec. 5). Based on these reasons, multimodal DL is here foreseen as an appealing alternative toward a sophisticated form of information fusion, named *intermediate fusion* [19], overcoming both the limitations of the *early* (or *data*) *fusion* and *late* (or *score/decision*) *fusion*, offering a truly-flexible tool for practical mobile TC enjoying multi-modality. The description of MIMETIC framework is provided hereinafter.

As sketched in Fig. 1(a), at an abstract level, the architecture of MIMETIC is fed with P different inputs (*modalities* or *views*) for each TC object to be classified, with p^{th} modality provided from *Input-data$_p$* extraction block. Such deep network architecture is first composed of J_p *single-modality* (input-specific) *layers*, allowing to extract in an increasingly-abstract fashion the discriminative features pertaining to p^{th} view, capitalizing intra-modality dependence. Specifically, the set of parameters referring to single-modality layers of p^{th} modality is referred to as $\boldsymbol{\theta}_p$. On the top of these layers, the abstract features are joined via a *merge layer* (cf. Fig. 1), which represents the first layer channeling the modality-specific distilled information toward a joint multimodal representation. Although the most general (and common) choice is represented by a *concatenation operation*, other approaches may be pursued in case the abstract features originating from different modalities have the *same size*, e.g. averaging, entry-wise maximum, etc.

Finally, the architecture is completed with a few *shared representation layers*, distilling features capturing inter-modality dependencies, and the usual *softmax layer*, returning the soft-output vector for the mobile TC task considered. Hereinafter, the set of parameters referring to shared representation layers (plus final softmax) is referred to as $\boldsymbol{\theta}_0$. Also, to promote *regularization* (so as to avoid overfitting), dropout between successive layers and early-stopping techniques are adopted [9]. We now briefly recall some common choices for single-modality and shared-representation layers, i.e. *dense*, *convolutional*, *pooling*, and *recurrent layers*.

Dense layers are the simplest atoms of DL architectures, made of a linear transformation and an entry-wise activation. Differently the *Convolutional layers* are the basic blocks of *CNNs*, made of a set of translation-invariant (1D or 2D, based on the specific input nature) filters which aim at extracting features of a certain region. *Pooling layers* are other key elements of CNNs, down-sampling intermediate representations from convolutional layers, with the aim of complexity and overfitting mitigation. Finally, *recurrent layers* present loopy connections and have in Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) their most popular variants [9]. These are in charge of *recalling* values over time, via a state vector $\boldsymbol{h}[t]$, and accept as input a vector sequence. Differently, they output either the final state $\boldsymbol{h}[T]$ or its entire time-evolution $\boldsymbol{h}[1], \cdots, \boldsymbol{h}[T]$. LSTM/GRU layers can be also conceived in an improved "bidirectional" form, i.e. their internal representation is split into forward and backward directions.

Given the general definition of MIMETIC architecture, we next describe the algorithmic procedure for its training.

3.2. Proposed Training Procedure

The high-level procedure suggested for training a classifier in the MIMETIC architecture is shown as pseudocode in Algorithm 1 and described hereafter.

```
  /* pre-training */
1 for Modality $p \in [1, P]$ do /* parallelizable */
2     ( $\hat{\boldsymbol{\theta}}_p, \hat{\boldsymbol{\theta}}_p^{\text{stub}}$ ) ← trainSingleM($\boldsymbol{\theta}_p, \boldsymbol{\theta}_p^{\text{stub}}$, TrainingSet)
      /* $\hat{\boldsymbol{\theta}}_p^{\text{stub}}$ is discarded */
3     $[\hat{\boldsymbol{\theta}}_p^{\downarrow}\ \hat{\boldsymbol{\theta}}_p^{\uparrow}] \leftarrow \hat{\boldsymbol{\theta}}_p$ ;                /* $\hat{\boldsymbol{\theta}}_p$ is split */
4 end
  /* fine-tuning */
5 $\boldsymbol{\theta}^{\downarrow} \leftarrow \hat{\boldsymbol{\theta}}^{\downarrow}_{1,\ldots,P}$ ;                            /* frozen */
6 $\boldsymbol{\theta}^{\uparrow} \leftarrow \hat{\boldsymbol{\theta}}^{\uparrow}_{1,\ldots,P}$ ;                       /* initialized */
7 ( $\boldsymbol{\theta}_0, \boldsymbol{\theta}^{\uparrow}$ ) ← trainMultiM($\boldsymbol{\theta}_0, \boldsymbol{\theta}^{\downarrow}, \boldsymbol{\theta}^{\uparrow}$, TrainingSet)
```

Algorithm 1: Pseudo-code of MIMETIC Training Procedure.

The architecture of MIMETIC is trained via a *two-stage phase*, made of *pre-training* and *fine-tuning* [9]. The reason for a preliminary pre-training procedure is to correctly distill discriminative information from each modality so as to capitalize the advantage of the multimodal traffic representation. Before proceeding, we recall that training of DL approaches resort to the "one-hot" representation [9] of each label $\ell_{(m)}$, namely $\boldsymbol{t}_{(m)} \triangleq [t_{1,(m)}, \cdots, t_{L,(m)}]$, whose entries are all zero, save from a single "1" corresponding to $\ell_{(m)}^{th}$ class.

Specifically, each single-modality stack is first (pre-)trained independently, i.e. *without* the shared representation layers and by topping each modality chain with a softmax layer "stub" (whose parameters are collected within $\boldsymbol{\theta}_p^{\text{stub}}$), see Alg. 1 lines 1–4 and Fig. 1(b). Specifically, p^{th} "stubbed" chain is trained to minimize the classification loss function $\mathcal{L}_p(\cdot)$ with the intent of promoting p^{th} modality capability to solve the TC task *alone*, defined as:

$$\mathcal{L}_p\left(\boldsymbol{\theta}_p, \boldsymbol{\theta}_p^{\text{stub}}\right) = \sum_{m=1}^{M} w_m \, \text{CE}(\boldsymbol{t}_{(m)}, \, \boldsymbol{c}_{(m)}[\boldsymbol{\theta}_p, \boldsymbol{\theta}_p^{\text{stub}}]) \quad (1)$$

Herein, the vector $\boldsymbol{c}_{(m)} \triangleq [c_{1,(m)}, \cdots, c_{L,(m)}]$ collects the predicted class confidences of DL classifier (which depend on the network parameters) for the label of m^{th} training sample. These confidences should be as close as possible to the (ground-truth originated) one-hot vector $\boldsymbol{t}_{(m)} \triangleq [t_{1,(m)}, \cdots, t_{L,(m)}]$. Such distance is measured via $\text{CE}(\boldsymbol{t}, \boldsymbol{c}) \triangleq -\{\sum_{\ell=1} t_\ell \log c_\ell\}$, denoting the *categorical cross-entropy* of m^{th} training sample. Furthermore, MIMETIC includes the minimization of a general *weighted* form of the categorical cross-entropy, with w_m denoting the weight of m^{th} sample, enabling *cost-sensitive learning* [9]. Indeed, the weight w_m allows penalizing/favoring (during training phase) the discrimination capability toward some app(s) and/or mitigating the class-imbalance problem. The learned parameters from the above optimization are indicated with $(\hat{\boldsymbol{\theta}}_p, \hat{\boldsymbol{\theta}}_p^{\text{stub}})$.

Then, during the *fine-tuning* phase (Fig. 1(c) and Alg. 1 lines 5–7), the above softmax stubs are removed (i.e. $\hat{\boldsymbol{\theta}}_1^{\text{stub}}, \cdots, \hat{\boldsymbol{\theta}}_P^{\text{stub}}$ are discarded from the optimization) and training of the *whole* MIMETIC architecture is performed (i.e. including both the parameters of single-modality layers $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_P$ and of shared representation layers $\boldsymbol{\theta}_0$, associated to block *MM*). However, as a result of the pre-training phase, a share of single-modality layers (i.e. those corresponding to low-layers in DL hierarchy, named IM_1) are typically *frozen* when fine-tuning phase is performed. This is due to the fact that low-level layers refer to *intra-modality automatic* feature extraction [18]. In other terms, within $\boldsymbol{\theta}_P \triangleq \left[\boldsymbol{\theta}_p^{\downarrow} \quad \boldsymbol{\theta}_p^{\uparrow}\right]$ only the subset $\boldsymbol{\theta}_p^{\uparrow}$ is (further) optimized during fine-tuning (i.e. those corresponding to IM_2), while $\boldsymbol{\theta}_p^{\downarrow}$ is kept *fixed* to the value learned during pre-training, i.e. $\boldsymbol{\theta}_p^{\downarrow} = \hat{\boldsymbol{\theta}}_p^{\downarrow}$. As a result, the following *weighted* form of the *categorical cross-entropy* is minimized:

$$\mathcal{L}\left(\boldsymbol{\theta}_1^{\uparrow}, \cdots, \boldsymbol{\theta}_P^{\uparrow}, \boldsymbol{\theta}_0\right) \triangleq \sum_{m=1}^{M} w_m \, \text{CE}(\boldsymbol{t}_{(m)}, \boldsymbol{c}_{(m)}[\boldsymbol{\theta}_1^{\uparrow}, \cdots, \boldsymbol{\theta}_P^{\uparrow}, \boldsymbol{\theta}_0]) \quad (2)$$

The loss functions concerning pre-training and fine-tuning phases ($\mathcal{L}_p(\cdot)$ and $\mathcal{L}(\cdot)$, respectively) are minimized via standard first-order local optimizers (e.g., SGD, ADAM, etc.), resorting to the usual back-propagation for gradient evaluation [9]. We now present the specific instance obtained from MIMETIC framework and used for experimental evaluation.

3.3. Implementation of a Traffic Classifier based on MIMETIC

The specific implementation[1] of the proposed MIMETIC framework (see Fig. 2) operates at *biflow level* (aiming at a consistent comparison with earlier works–except for [12]–employing single-modality DL for TC) and is made of $P = 2$ *modalities*. These are fed with the corresponding two types of input, that are naturally suited for "early" TC [34] and have been already employed successfully in most related works performing TC via single-modality DL [10, 11, 13, 16]: (I) the

[1]We highlight that we have leveraged DL models provided by *Keras* [32] (Python) API running on top of *TensorFlow* [33] to implement and test the MIMETIC instance described in this section and DL-based classifiers used as baselines.

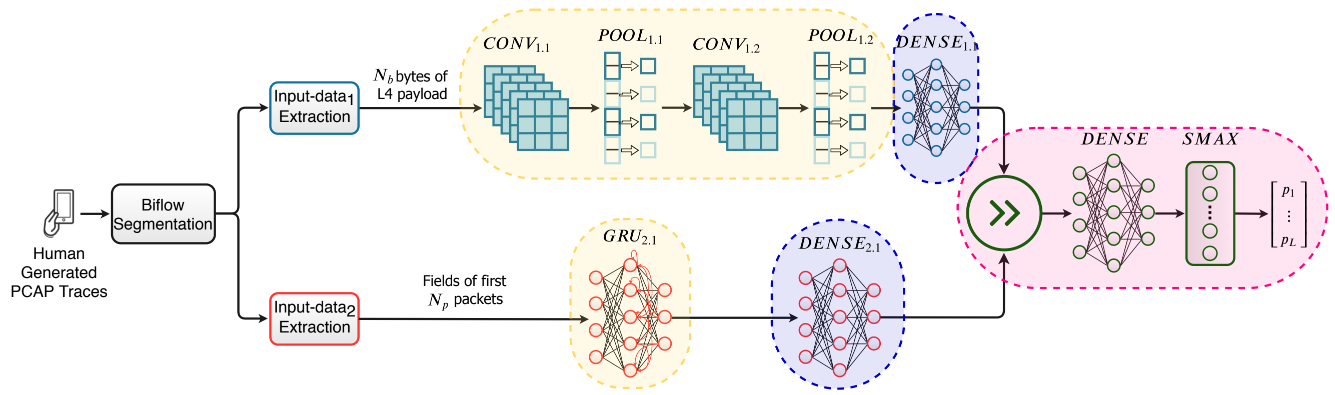

Human Generated PCAP Traces
Biflow Segmentation
Input-data$_1$ Extraction
N_b bytes of L4 payload
$CONV_{1.1}$
$POOL_{1.1}$
$CONV_{1.2}$
$POOL_{1.2}$
$DENSE_{1.1}$
Input-data$_2$ Extraction
Fields of first N_p packets
$GRU_{2.1}$
$DENSE_{2.1}$
$DENSE$
$SMAX$
p_1
p_L

Figure 2: Implementation of considered traffic classifier based on MIMETIC framework. Background colors specify the groups of layers IM_1 (in yellow), IM_2 (in blue), and MM (in purple).

first N_b bytes (normalized within $[0, 1]$) of payload [10, 11]; (II) informative protocol fields, not pertaining to the explicit inspection of the encrypted payload—namely: number of bytes in transport-layer payload, TCP window size (set to *zero* for UDP packets), inter-arrival time, and packet direction $\in \{0, 1\}$—of first N_p packets [13]. We remark that we focus on payload at application-layer in TCP/IP stack in (I) and, also, we do not consider port info in (II), as otherwise these may both lead to biased and inflated performance, as shown in [16]. Finally, in the case of instances with inputs longer (resp. shorter) than the considered fixed-length data formats, these inputs are truncated (resp. padded with zeros) to the designed length of bytes (N_b) or (N_p) packets. We underline that the above "traffic-originated" modalities refer to different levels of abstraction (packet vs. biflow depth) and standpoints (encryption-dependent vs. encryption independent) for the observed traffic. This inspired the adoption of a multi-modal architecture to improve classification performance, as later supported by the experimental validation in Sec. 5. For the mentioned reasons, we parallel the use of both modalities as for audio and video modalities in natural language understanding.

Hereinafter, we refer to the building blocks of Fig. 2 to describe the specific implementation of proposed MIMETIC framework. The single-modality layers of the *first view* (the "payload" modality) are two 1D convolutional layers ($CONV_{1.1}$ and $CONV_{1.2}$, made of 16 and 32 filters, respectively, with kernel size of 25, unit stride, and Rectified Linear Unit activations), each followed by a 1D max-pooling layer ($POOL_{1.1}$ and $POOL_{1.2}$, with unit stride and spatial extent equal to 3) and, finally, by one dense layer ($DENSE_{1.1}$, with 256 nodes). The reason for this choice is the ability of 1D convolutional layers to extract spatially-invariant (discriminative) patterns from the payload. On the other hand, the single-modality layers of the *second view* (the "protocol fields" modality) are, in order, a bidirectional GRU ($GRU_{2.1}$, with 64 nodes and return-sequences behavior) and one dense layer ($DENSE_{2.1}$, with 256 nodes). Such choice was driven by GRU ability to capture long-term dependencies pertaining to the initial segments of the biflow, while requiring slightly-less parameters with respect to a more common LSTM [9]. The intermediate features of the two branches are then concatenated via a *merge layer* (>>), and fed to a *dense (shared representation) layer* ($DENSE$, with 128 nodes), before the softmax ($SMAX$). In all the layers, the outputs are obtained via Rectified Linear Unit activations. Finally, 20% dropout is applied after (a) each dense layer (including the merge layer) and (b) after flattening the 2D representation of both the stack of convolutional/pooling layers and GRU.

The considered architectural instance is *trained* via the *two-stage phase* described in Sec. 3.2. Specifically, all the classification loss functions ($\mathcal{L}_1(\cdot), \cdots, \mathcal{L}_P(\cdot)$ and $\mathcal{L}(\cdot)$) include cost-sensitive learning, here exploited to mitigate natural class imbalance found in mobile traffic [21]. To this end, the weight $w_m \triangleq M/M_{\ell(m)}$ is assigned to m^{th} sample, being inversely proportional to the number of training set samples labeled with $\ell_{(m)}$, thus magnifying (resp. decreasing) the contribution of apps with a few (resp. high) number of samples. Concerning the *pre-training* phase, each single-modality stack is first (pre-)trained independently for 25 epochs each by topping a softmax layer stub and by minimizing the loss $\mathcal{L}_p(\cdot)$ (cf. Eq. (1)), so that mobile TC could be performed on either (transport-layer) payload or protocol layer fields. Then, *fine-tuning* of the whole multimodal DL architecture is performed (for 40 epochs) after freezing IM_1 (low-layers, namely, the convolutional and recurrent layers, $CONV_{1.1}/CONV_{1.2}$ and $GRU_{2.1}$, respectively) and by minimizing the loss $\mathcal{L}(\cdot)$ (cf. Eq. (2)). For both phases, ADAM optimizer (batch size of 50) and early-stopping technique (to prevent overfitting) measured on the training accuracy have been employed. We underline that the overall number of epochs ($25 \times 2 + 40 = 90$) has been chosen by considering the values suggested in more-related works [11, 13] so as to keep the complexity low, while properly training the architecture.

4. Experimental Setup

Hereinafter, we describe the three mobile traffic datasets, along with the performance evaluation framework, used for assessment of proposed MIMETIC framework.

4.1. Dataset Description

We considered *both Android and iOS* mobile OSes (instead of only one, as usually done in related works), with an ample representation of apps. Moreover all the datasets have been collected by real *human users* using the mobile apps (instead of relying on automatically-generated traffic, as done in related works). The collection of first dataset (FB/FBM) has been recommended by a global mobile solution provider, while the other two (namely Android and iOS) have been produced and handed to us directly by the same provider[2]. The envisioned usage scenarios for such datasets are related to key network management tasks, e.g. service prioritization, billing differentiation, censorship. For all the datasets, the ground truth has been obtained by labeling each traffic trace with the generating app, since each app has been run *separately*, thus limiting the presence of background traffic.

The *first (binary) dataset* was collected in the ARCLAB laboratory at the University of Napoli "Federico II", during May '17 - Mar. '18. In detail, the capture sessions have been run on a Xiaomi Mi5 (with Android OS 6.0.1 and CyanogenMod 13.0 distro) and pertain to either Facebook (FB) or Facebook Messenger (FBM) traffic data, to analyze the capability of classifiers to discriminate between similar apps' fingerprints for e.g., *billing differentiation*. More than 100 users have been involved in its construction on a voluntary basis for sittings lasting less than 2 hours, being required to perform different activities for both the apps (to explore their diversity), in union with login/registration/logged-in use cases. Each traffic-capture session lasted from 5 to 10 minutes, with > 1100 traffic traces collected. During each capture session, a user executed only one app in foreground. Background traffic was then removed from the collected traces in post-processing, based on the system-calls traced on the mobile device. The whole dataset contains $\approx$ 31k instances, with 13.5k (resp. 17.5k) biflows from FBM (resp. FB) app and a 44%/56% share.[3] The encrypted biflow ratio corresponds to 91%.

The *other two (multi-class) datasets*, obtained from the global mobile solutions provider and generated from 49 (resp. 45) apps on Android (resp. iOS) devices, are explored for *prioritization purposes*. The corresponding Android (resp. iOS) traces have been collected during Apr. '15 - Jan. '17 (resp. Sept. '14 - Jan. '17), generated by users with different devices and OS/app versions, and provided already anonymized and cleaned from background traffic.[4] As a whole, the dataset is made up of 607 (resp. 419) traffic traces, with an average duration of 282 (resp. 296) seconds and 1 to 60 (resp. 1 to 48) traces per app in Android (resp. iOS), leading to a non-negligible class imbalance. Then, after *biflow* segmentation, 55.5k (resp. 37.2k) labeled biflows are obtained for the Android (resp. iOS) dataset, with 47% (resp. 60%) encrypted biflow ratio. Finally, a detailed report of per-class biflow statistics can be found in [21], where both datasets were employed for ML-based (handcrafted) mobile TC.

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

[3]The current dataset constitutes a larger version w.r.t. that considered in [16], in terms of both depth and diversity, while improving also FB/FBM samples balance (i.e. 44%/56% vs. 38%/62% share of [16]).

[4]In detail, $\approx$ 89% (resp. $\approx$ 85%) of Android (resp. iOS) traces has been captured during 2016.

4.2. Performance Evaluation Framework

Our evaluation includes the following well-known performance measures [1]: *accuracy* (the fraction of correctly classified instances), *precision* (prec, i.e. the proportion of classifier decisions for a given class which are actually correct), and *recall* (rec, i.e. the app-conditional accuracy). For the latter two (defined on a per-app basis), the usual *F-measure* $\mathrm{F} \triangleq (2 \cdot \mathrm{prec} \cdot \mathrm{rec})/(\mathrm{prec} + \mathrm{rec})$ is considered for conciseness, and its arithmetically-averaged (viz. macro) version is adopted. In addition, the "global behavior" of DL-based TC is evaluated by means of *Top-K accuracy*, defining a correct classification event if the actual class is within the top K predicted apps ($K < L$ is a free parameter and $K = 1$ coincides with the standard accuracy) and allowing to investigate the soft-output of a DL classifier. Besides, the *confusion matrices* are used to highlight fine-grain misclassification patterns.

The classifiers are also tested when a *reject option* (viz. a censoring policy of "unsure" outcomes) is adopted, i.e. the classification is performed only if the highest class prediction probability ($\max_{\ell=1,\ldots,L} p_\ell$) exceeds a threshold γ, thus emitting a confident verdict. Its adoption has been justified in the context of mobile traffic [7]. Indeed, since apps typically establish multiple flows when they are running, there remains high chance to identify them from their more distinctive flows, without the need to classify all the instances. Hence, tuning γ provides further (useful) *flexibility* to mobile TC, since classification performance can be improved while incurring a negligible drawback, i.e. a *decreased* ratio of classified instances (CR).

For completeness, we also investigate the computational complexity of (multimodal) DL-architectures by reporting their training phase runtime. The latter is a key aspect in mobile TC, where frequent re-training is required, due to aging of training data as a result of apps and OS updates [7, 24]. Precisely, since training is performed on multiple epochs [9], we report such information in a terse way, by providing the *Run-Time Per-Epoch (RTPE)*. We note that, for each analysis, our evaluation is based on a *(stratified) ten-fold cross-validation*, (*i*) representing a stable evaluation process and (*ii*) mantaining the same share of class imbalance in both training and test sets within each fold. Thus, we report both the mean and the standard deviation of each performance measure by evaluating them on the ten different folds.

5. Experimental Evaluation

This section examines the performance (from *both* classification and complexity standpoints) of MIMETIC approach and methodically compares it to existing (single-modality) DL alternatives [11, 13], approaches to fuse their information [21], and ML-based state-of-the-art in mobile TC [7].

In the following, for compactness we will refer with "L7-N_b" to the first N_b bytes of payload data [11] and with "MAT-N_p" to the $N_p \times 4$ matrix of protocol fields extracted from each biflow [13]. We point out that, with a view to limit complexity, preliminary analyses (omitted for brevity) of different input combinations for each single-modality branch have been performed to select the values of N_b and N_p. Specifically, varying

Table 3: Accuracy and F-measure [%] comparison of MIMETIC with the four groups of baselines: (I) best single-modality DL classifiers, (II) shallow neural networks, (III) state-of-the-art ML-based mobile-traffic classifier, (IV) classifier fusion techniques. Results are in the format *avg. (± std.)* obtained over 10-folds. The last group reports the *Maximum Improvement Over Best - Classifier* (MIOB-C) and the *Maximum Improvement Over Best - Fusion Technique* (MIOB-FT) [%] of MIMETIC framework. Highlighted values: **overall best classifier**, best baseline classifier (♦), and best baseline fusion technique (‡) for each dataset and performance measure.

	Architecture	*FB/FBM*		*Android*		*iOS*	
		Accuracy	*F-measure*	*Accuracy*	*F-measure*	*Accuracy*	*F-measure*
	MIMETIC	**79.98 (± 0.49)**	**79.63 (± 0.51)**	**89.49 (± 0.32)**	**81.51 (± 0.93)**	**89.14 (± 0.82)**	**82.99 (± 1.14)**
I	1D-CNN [11] (L7-784)	75.92 (± 0.76)	75.45 (± 1.12)	85.44 (± 0.45) ♦	78.13 (± 1.73) ♦	83.29 (± 0.62) ♦	74.41 (± 1.28) ♦
I	LSTM + 2D-CNN [13] (MAT-20)	74.26 (± 0.98)	73.23 (± 0.95)	75.24 (± 0.58)	64.35 (± 0.87)	70.80 (± 1.06)	57.87 (± 1.15)
II	MLP-1 (L7-784)	74.46 (± 0.88)	73.89 (± 0.86)	78.71 (± 0.65)	69.79 (± 1.17)	77.16 (± 0.63)	67.61 (± 1.07)
II	MLP-1 (MAT-20)	68.93 (± 1.32)	67.86 (± 0.94)	64.94 (± 0.47)	48.26 (± 0.96)	54.42 (± 0.63)	40.86 (± 1.04)
III	RF (flow-based) [7]	79.56 (± 0.62) ♦	78.73 (± 0.62) ♦	84.78 (± 0.30)	75.49 (± 0.89)	80.77 (± 0.84)	72.39 (± 1.39)
IV	MV	75.13 (± 0.92)	74.48 (± 1.14)	80.41 (± 0.40)	71.28 (± 0.85)	77.24 (± 0.62)	66.49 (± 0.97)
IV	SOA	78.86 (± 0.79) ‡	78.37 (± 1.00) ‡	87.08 (± 0.29) ‡	80.07 (± 0.81) ‡	84.68 (± 0.55) ‡	75.94 (± 1.10) ‡
IV	TLF	74.61 (± 1.57)	73.60 (± 1.80)	68.87 (± 1.05)	48.82 (± 1.92)	62.01 (± 0.97)	39.07 (± 1.52)
	MIOB-C	+ 0.42 (± 0.65)	+ 0.90 (± 0.68)	+ 4.04 (± 0.67)	+ 3.38 (± 1.79)	+ 5.84 (± 0.97)	+ 8.58 (± 1.52)
	MIOB-FT	+ 1.12 (± 0.89)	+ 1.26 (± 1.14)	+ 2.40 (± 0.48)	+ 1.44 (± 1.56)	+ 4.46 (± 1.01)	+ 7.05 (± 1.43)

$N_b \in \{256 - 2034\}$ and $N_p \in \{4 - 32\}$, employed values proved to achieve the best performance, keeping both the complexity low and also allowing an "earlier" TC. In detail, our MIMETIC instance is fed with the same two input types (cf. Sec. 3) of single-modality DL classifiers used as baselines, but with different (i.e. shorter) amounts of data, i.e. $N_b = 576$ bytes and $N_p = 12$ packets, respectively.

5.1. Description of Baselines

Precisely, *four types of baselines* are included for the sake of a complete analysis.

- The *first* type of baseline is represented by the best single-modality DL classifiers fed with each of the $P = 2$ inputs considered in the MIMETIC approach, that is, the 1D-CNN (with L7-784 as input) in [11] and the LSTM + 2D-CNN (with MAT-20 as input) in [13]. We remark that the number of payload bytes (resp. packets) used in 1D-CNN (resp. LSTM + 2D-CNN) differs from that used for MIMETIC implementation: the reason was to report performance for corresponding input-optimized versions proposed in respective works [11, 13]. Nonetheless, preliminary experimental results (reported in [20]) on these architectures did not demonstrate an appreciable gain when using $N_b = 576$ bytes (resp. $N_p = 12$ packets). Both architectures are trained for a total of 90 epochs, following the suggestions in related studies [11, 13], with ADAM optimizer (batch size of 50) and early-stopping technique measured on the training accuracy.
- The *second* type of baseline corresponds to a lower-bound on achievable performance, namely we consider a Multi-Layer Perceptron (MLP) with only one hidden layer (with 100 nodes), here denoted as MLP-1, trained on the same inputs as single-modality DL architectures, so as to provide the performance achievable by "shallow" learning in the same setup. Aiming at a consistent comparison, the same number of epochs (i.e. 90), optimizer (i.e. ADAM) and batch size (i.e. 50) are used for training of MLP-1 (along with early-stopping).
- The *third* baseline is given by the flow-based RF developed in [7], taking as input 40 carefully handcrafted features, being a subset of statistics[5] computed on the sets of upstream, downstream, and complete (i.e. both of them) IP packet lengths. Such flow-based RF represents the current state-of-the-art mobile-traffic classifier, but is applicable only in the case of "post-mortem" TC (as opposed to all types of input considered in the DL/shallow baselines investigated herein).
- The *fourth* type of baseline corresponds to classifier fusion techniques [21], capitalizing the best single-modality DL architecture for each of the $P = 2$ inputs considered (namely, the architectures representing the first set of baselines) and combining them to get (hopefully-)improved classification results. The combination can be either performed with (simpler) non-trainable strategies, such as (*i*) Majority Voting (MV) and (*ii*) Soft-Output Average (SOA) [21]. Alternatively, (*iii*) "Trainable" Late Fusion (TLF) can be pursued by concatenating the softmax layers of the two single-modality DL architectures and connecting them to a "fusion" softmax layer (in the same spirit of "KL weights" in [21]). In the latter case, the same training algorithm (with the same parameters) as the other DL baselines has been adopted. These combiners are the simplest way to fuse these off-the-shelf DL traffic classifiers.

5.2. General Overview of Performance

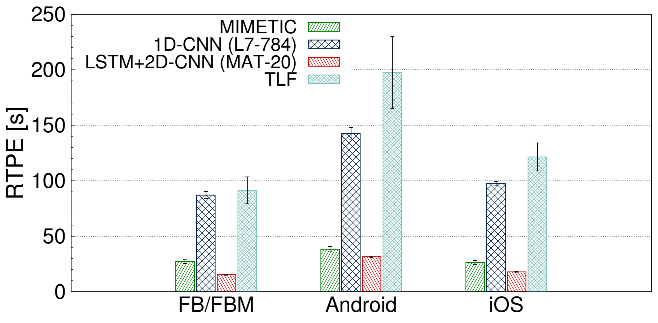
As a high-level performance comparison, in Tab. 3 we report the results (in terms of accuracy and F-measure) of the proposed

[5]The 40 best-ranked statistics (i.e. min, max, mean, standard deviation, variance, mean absolute deviation, skewness, kurtosis, and percentiles) based on the Gini impurity score [7].

MIMETIC approach, along with those of the baselines previously introduced. First, it is apparent that the *multimodal-DL architecture outperforms all the considered elements of comparison* for both metrics on all the three considered datasets, with an *improvement up to* +8.58% *and* +7.05% (i.e. F-measure on the iOS dataset) over the best classifier (MIOB-C) and fusion technique (MIOB-FT), respectively. Further, results pertaining to binary dataset FB/FBM highlight that, although single-modality DL classifiers are able to outperform the corresponding MLP-1 (shallow) counterparts, these are not able to reach performance of (off-line) RF, even employing fusion techniques of single-modality DL approaches (i.e. SOA). This may be attributed to the fact that FB and FBM rely on several shared services. Hence, their generated traffic looks very similar. Accordingly, either the whole biflow is required to reach a confident decision or more sophisticated approaches (e.g. MIMETIC) are required to infer complex traffic patterns from the first packets. On the other hand, referring to the multi-class datasets, single-modality DL approaches are not only able to provide improved performance w.r.t. shallow classifiers with analogous inputs, i.e. MLP-1 (L7-784/MAT-20), but even outperform ML-based state-of-the-art RF, thus motivating the strong appeal of DL framework and representing the best baseline in most cases. For example, in Android setup, 85.44% accuracy and 78.18% F-measure are achieved by 1D-CNN (L7-784), as opposed to 84.78% and 75.49%, respectively, obtained by the RF, and a similar reasoning applies to iOS case. An improved (although similar) trend is obtained by leveraging the SOA (the best fusion baseline observed) of the single-modality DL architectures, whose results are however worse than those obtained with MIMETIC approach. This outcome can be directly attributed to the gain, ensured by the (multimodal) MIMETIC approach, arising from sophisticated fusion of the input types considered. Indeed, the latter provides a higher discriminative power in the case of very similar apps, like FB and FBM, and also further improves the effectiveness of DL in the multi-class setups.

5.3. Training Complexity of DL Architectures

A useful analysis towards real-world implementations, complementing the overview of performance, is the investigation of the training complexity of the proposed multimodal-DL classifier. With this aim, Fig. 3a and Fig. 3b show, respectively, the RTPE and the overall number of trainable parameters of the considered instance of MIMETIC framework (considering both phases of the proposed training procedure). MIMETIC complexity is then compared against that of single-modality DL classifiers when they are fed with the inputs extracted from the three datasets. For completeness, also the complexity of TLF baseline is considered. Differently, the complexity of the other two classifier fusion baselines (namely, SOA and MV) is not reported, since it is strongly linked to the training requirements of the single-modality DL classifiers being combined. Precisely, it corresponds to the more complex single-modality baseline (resp. the sum of baseline complexities) in the case of a parallel (resp. sequential) implementation. We point out that a similar reasoning applies to the pre-training phase of MIMETIC, for which the most penalizing sequential implementation of per-modality pre-training is assumed in this comparison. We highlight that the times refer to the same hardware architecture (8 × Intel(R) Core(TM) i7-4710MQ CPU @ 2.50GHz with Ubuntu 16.04 (64 bit)) in the same load conditions (i.e. being the DL classifier the sole CPU-intensive process running on it). As expected, a decreasing RTPE is obtained when the size of the classification problem is reduced (i.e. moving from the Android dataset, to the iOS and FB/FBM datasets) with a stronger trend for 1D-CNN (L7-784) and TLF, being the more complex architectures. Interestingly, *multimodal-DL classifier* not only reaches the highest classification performance, but it also *shows an RTPE* > 3.5× *lower* than its "main competitor" 1D-CNN (L7-784) (i.e. 38.34 (±0.82) s vs. 142.72 (±1.74) s) in the hardest classification (i.e. Android) setup. This is due mainly to shorter inputs and simpler (viz. computationally-lighter) layers involved in the MIMETIC instance (cf. Sec. 3.3), allowed by an improved capitalization of the inputs available. It is worth noting that the same reasoning equally applies with respect to MV and SOA baselines, whose complexities are dominated (in the best case) by the more complex single-modality DL architecture. Moreover, MIMETIC shows also the lowest complexity increase when passing to a harder classification problem (i.e. it exhibits a higher scalability), being highly desirable in mobile contexts. Indeed, a +41% increment in RTPE (against +64% for 1D-CNN, +105% for LSTM+2D-CNN, and +116% for TLF) is observed when moving from the FB/FBM to the Android dataset. Finally, the inspection of Fig. 3b highlights that the RTPE is strongly related to the number of parameters to be trained, with the MIMETIC approach (in its complete architectural configuration) having ≈ 3.6× and ≈ 4.1× fewer trainable parameters than 1D-CNN and TLF, respectively.



(a) RTPE. Results are in the format *avg. (± std.)* obtained over 10-folds.

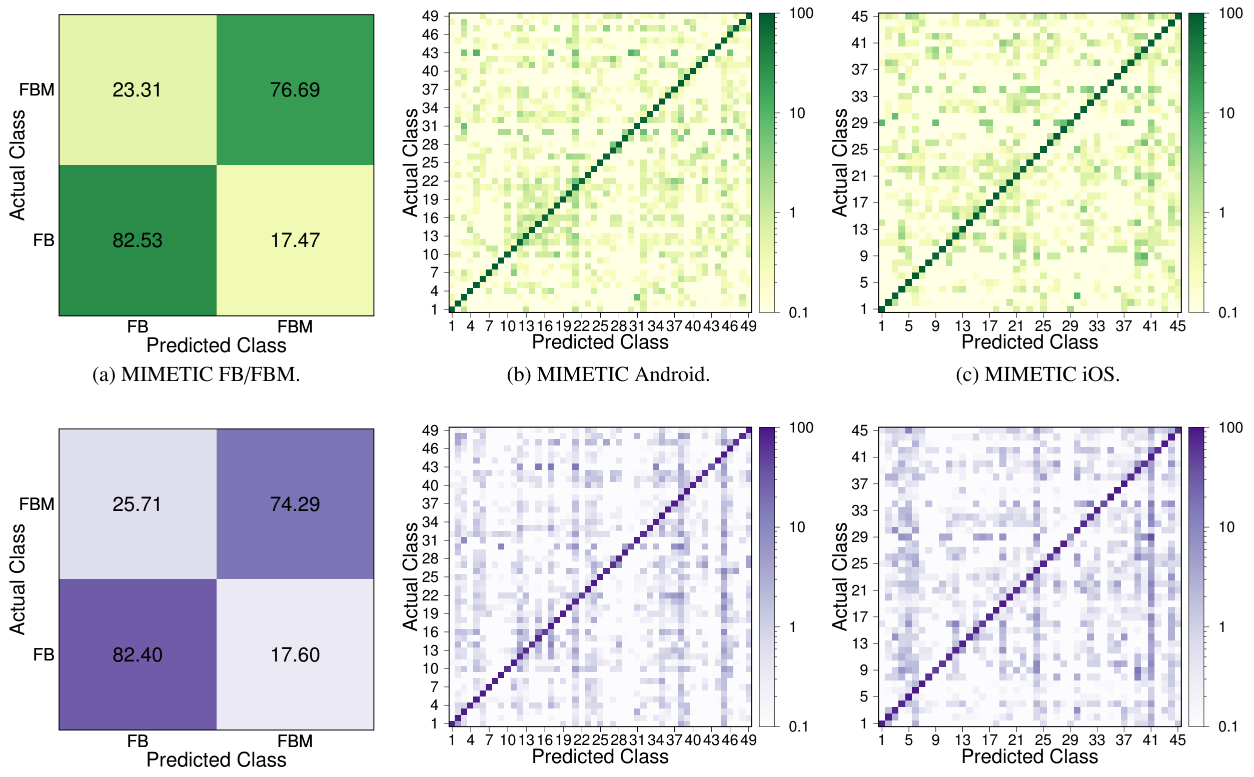
	FB/FBM	Android	iOS
MIMETIC⋆	0.9346	1.6202	1.6176
1D-CNN [11] (L7-784)	5.8223	5.8705	5.8664
LSTM + 2D-CNN [13] (MAT-20)	0.4260	0.7380	0.7376
TLF	6.2484	6.6133	6.6081

(b) Number of trainable parameters (in millions). ⋆ numbers refer to the whole MIMETIC framework.

Figure 3: Complexity analysis. Run-Time Per Epoch (RTPE) (a) and number of trainable parameters (b) of MIMETIC framework and DL-based baselines.

Table 4: Top-K accuracy [%] comparison of MIMETIC with the four types of baselines. Results refer to multi-class datasets and are in the format *avg. (± std.)* obtained over 10-folds. Top-K accuracy of MV is not reported due to unavailability of soft-outputs. Highlighted values: **overall best classifier**, best baseline classifier (♦), and best baseline fusion technique (‡) for both multi-class datasets and each K considered.

	Architecture	*Android*			*iOS*		
		$K = 1$	$K = 3$	$K = 5$	$K = 1$	$K = 3$	$K = 5$
	MIMETIC	**89.49 (± 0.32)**	**94.29 (± 0.28)**	**95.82 (± 0.28)**	**89.14 (± 0.82)**	**95.17 (± 0.37)**	**96.74 (± 0.32)**
I	1D-CNN [11] (L7-784)	85.44 (± 0.45) ♦	91.48 (± 0.24)	93.48 (± 0.22)	83.29 (± 0.62) ♦	91.04 (± 0.36) ♦	93.42 (± 0.22)
	LSTM + 2D-CNN [13] (MAT-20)	75.24 (± 0.58)	85.60 (± 0.47)	89.80 (± 0.34)	70.80 (± 1.06)	83.34 (± 0.69)	87.82 (± 0.48)
II	MLP-1 (L7-784)	78.71 (± 0.65)	86.93 (± 0.40)	89.88 (± 0.37)	77.16 (± 0.63)	86.96 (± 0.50)	90.40 (± 0.51)
	MLP-1 (MAT-20)	69.94 (± 0.47)	79.22 (± 0.51)	84.94 (± 0.34)	54.42 (± 0.63)	72.47 (± 0.59)	80.03 (± 0.56)
III	RF (flow-based)	84.78 (± 0.30)	91.69 (± 0.31) ♦	93.89 (± 0.24) ♦	80.77 (± 0.84)	90.70 (± 0.61)	93.58 (± 0.52) ♦
IV	SOA	87.08 (± 0.29) ‡	92.83 (± 0.31) ‡	94.66 (± 0.26) ‡	84.68 (± 0.55) ‡	92.36 (± 0.28) ‡	94.65 (± 0.23) ‡
	TLF	68.87 (± 1.05)	79.35 (± 0.92)	83.41 (± 0.86)	62.01 (± 0.97)	75.03 (± 0.64)	80.40 (± 0.55)



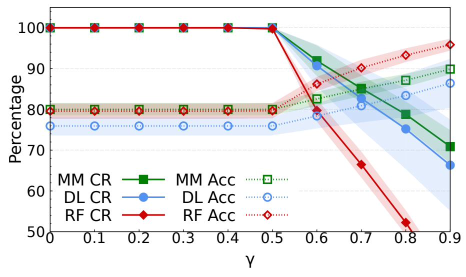
(a) MIMETIC FB/FBM. (b) MIMETIC Android. (c) MIMETIC iOS.

(d) SOA FB/FBM. (e) SOA Android. (f) SOA iOS.

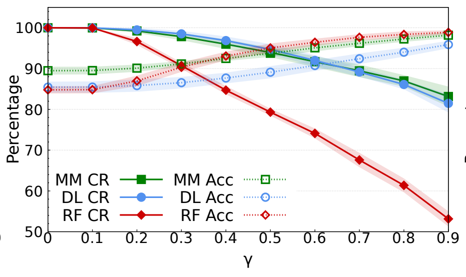
Figure 4: Confusion matrices of architecture based on MIMETIC framework (top) and Soft-Output Average (SOA) of best single-modality DL classifiers (bottom) for the FB/FBM (a & d), Android (b & e), and iOS (c & f) datasets. Note that the log scale is used to evidence small errors (except for FB/FBM). Categorical class-labels for multi-class datasets are the same as in [21].

5.4. *Fine-Grained Performance*

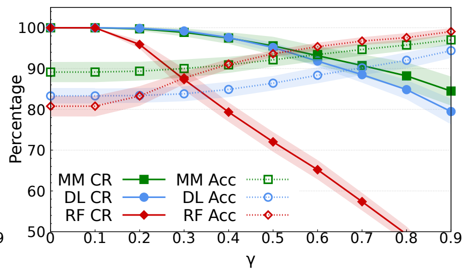
We now deepen the (classification) performance investigation of the MIMETIC framework (along with the baselines), initially by reporting their Top-K accuracy ($K \in \{1, 3, 5\}$) on the multi-class datasets in Tab. 4. We highlight that the above table does not include MV Top-K accuracy score, as the latter is based on classifiers' hard outputs and therefore there is no (natural) definition for the confidence vector associated to its decision. Clearly, from the inspection of these fine-grained results it is apparent that all the considered classifiers are able to improve their accuracy when a larger pool of predicted apps may be taken into consideration, (shallow) MLP-1 classifiers included. However, the latter classifiers approach neither the accuracy of the MIMETIC architecture nor of their single-modality DL counterparts. This is due to their inability of inferring deeply-structured traffic patterns as a whole. Indeed, they are not able to predict the true label even when considering the Top-K classes ranked by their confidence. On the other hand, *MIMETIC* reports also the *highest global accuracy* (soft-output) behavior in both the multi-class datasets. Indeed,



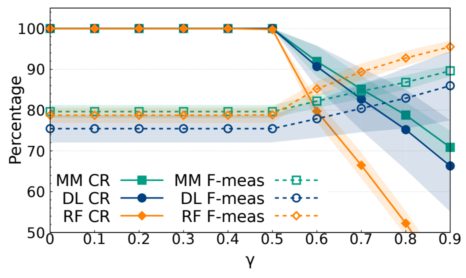
(a) FB/FBM.



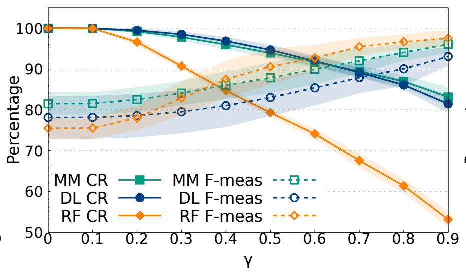
(b) Android.



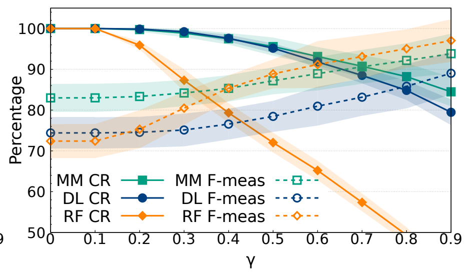
(c) iOS.



(d) FB/FBM.



(e) Android.



(f) iOS.

Figure 5: Accuracy (a-c), F-measure (d-f), and ratio of classified samples (CR) [%] vs. censoring threshold γ of MIMETIC (MM) framework vs. best single-modality DL classifier (DL) and state-of-the-art ML-based mobile-traffic classifier (RF). Average on 10-folds and corresponding $\pm 3\sigma$ confidence interval are shown. To ease direct comparison, the y-axes are limited to percentages over 50%: only CR of RF continues dropping below that threshold, to values of little practical interest.

although the (off-line) RF classifier provides a slightly better global behavior than the best single-modality DL classifier (for $K \in \{3, 5\}$ on Android dataset and $K = 5$ on iOS dataset), namely 1D-CNN (L7-784), it is *able to outperform neither the proposed MIMETIC approach nor the simpler SOA combination of single-modality DL classifiers*. This result suggests that the capitalization of multiple modalities improves the global discrimination capabilities of DL-based classifiers. In detail, in Android (resp. iOS) scenario, the MIMETIC architecture is able to reach 94.29% and 95.82% (resp. 95.17% and 96.74%) accuracy when the Top-3 and Top-5 predicted apps are considered, respectively.

Then, to assess possible relevant misclassification-patterns and their mitigation through intermediate-fusion, Fig. 4 shows the confusion matrices of the MIMETIC approach in the three datasets, and compares them with those obtained via SOA. From direct inspection, MIMETIC clearly achieves *less-structured and milder error patterns* in the three cases considered, as opposed to SOA. This confirms the flexibility of the information fusion provided by our framework. Equally important, the confusion matrices highlight the *appeal of cost-sensitive learning* within the MIMETIC formulation (see the definition of loss functions in Eqs. (1) and (2)), able to deal with imbalanced TC problems (which are common in the mobile context) by preventing a classification imbalance toward the most-represented classes.

5.5. Performance vs. Reject Option

Finally, as a complementary analysis oriented to finer performance control, Fig. 5 shows the accuracy and F-measure (first and second row of plots, respectively) of (*a*) the proposed MIMETIC approach, (*b*) the best single-modality DL approach (1D-CNN (L7-784)), and (*c*) the flow-based RF developed in [35] (i.e. the current ML-based state-of-the-art traffic classifier) vs. the censoring threshold γ on each of the three datasets. We exclude herein the SOA approach, due to lower performance (while having a common "fusion" rationale) with respect to MIMETIC. We notice that a threshold value implying varying performance w.r.t unclassified samples, can be observed only if $\gamma \geq 1/L$ (L corresponds to the number of classes).[6] In all the plots, for the sake of a thorough comparison, we also report the ratio of classified samples (CR) vs. γ (for $\gamma \geq 50\%$, being smaller values of little practical interest). We highlight that we opted to keep the visualization of accuracy/F-measure and CR *separate* so as to provide a finer understanding of the corresponding interplay among the two conflicting measures. By looking at the qualitative profiles of CR and both performance measures, the following considerations can be drawn. First, results highlight that all the methods enjoy improved classification performance when increasing γ, at the price of a decreasing CR. Secondly, the RF approach has the sharpest improvement of accuracy & F-measure with γ which is paid, unfortunately, with a quick CR decrease. This outcome may be explained with

[6]Specifically, this value corresponds to 0.5 in the case of the FB/FBM dataset, whereas it equals ≈ 0.02 for Android and iOS datasets.

a high number of biflows classified with low confidence by RF. As an example, by looking at Android dataset and having a 90% F-measure as a target, the *RF* approach *rejects double of the tested biflows w.r.t. the proposed MIMETIC approach* (20% vs. 10%, respectively). On the other hand, comparing MIMETIC with the (best) single-modality DL architecture, a similar behavior of the CR with γ is observed for the two approaches, whereas MIMETIC provides an almost-constant performance improvement over all the range. This again confirms the global performance gain originating from the adoption of multimodal-DL in our proposal. Specifically, by *rejecting* the classification of only **10**% *of instances*, on the Android dataset, the proposed *MIMETIC approach* is able to achieve *accuracy and F-measure such that* $\geq$ 95% *and* $\geq$ 90%, respectively. Similarly, for the iOS dataset, the proposed approach is able to achieve (roughly) the same targeted performance. Unfortunately, in the FB/FBM scenario, achieving $\geq$ 90% target performance on both measures would require $\approx$ 30% biflows to be censored. This result reflects the difficulty in solving an "overlapped-apps" classification task, sharing many third-party (common) services in their execution.

6. Conclusions and Future Directions

In this work we tackled classification of mobile (encrypted) traffic via a *multimodal-DL approach*, named *MIMETIC*, proposing a *general TC framework* able to capitalize heterogeneous input data (capturing intra- and inter-modal dependences) and implemented a specific instance tested on *three real users' datasets of mobile traffic*. The latter implementation has been show to *outperform both ML- and DL-based baselines* (with up to +8.58% improvement over the best baseline, i.e. 82.64% on iOS dataset), while having a RTPE > 3.5$\times$ lower than its "main single-mode DL competitor". A comparison of fine-grained performance also showed the *superiority of the MIMETIC approach* in a highly multi-class TC task, reaching 96.74% score when considering Top-5 accuracy on iOS dataset and a uniform misclassification pattern (as a result of cost-sensitive learning), as underlined by confusion matrices. Finally, enriching MIMETIC *with a reject option* allowed to *report a* $\geq$ 90% *F-measure* on both multi-class datasets (resp. binary dataset), by rejecting 10% (resp. 30%) of the examined biflows.

The proposed MIMETIC framework suggests the following *research directions*. First, the generality of the multimodal-DL TC framework proposed allows the adoption and the use of *more sophisticated* DL layers, such as inception and residual connections. Secondly the training procedure considered, including a pre-training phase, can be used in conjunction with the exploitation of massive unsupervised data for improved transfer learning. Thirdly, it is of clear interest the prototyping of multimodal-DL architectures able to cope with more challenging TC objects (e.g. service burst [7]), especially in the definition of the corresponding multiple modalities associated to different input types. Finally, a real-world implementation of multimodal-DL architectures in open-source tools (e.g. TIE [36]) is of relevant interest.

References

[1] A. Dainotti, A. Pescapè, and K. C. Claffy. Issues and future directions in traffic classification. *IEEE Network*, 26(1), 2012.

[2] F. Jejdling et al. Ericsson mobility report. *Ericsson AB, Business Area Networks, Stockholm, Sweden, Tech. Rep. EAB-18*, 4510, 2018.

[3] H. Shi and Y. Li. Discovering periodic patterns for large scale mobile traffic data: Method and applications. *IEEE Transactions on Mobile Computing*, 2018.

[4] H. Yao, G. Ranjan, A. Tongaonkar, Y. Liao, and Z. M. Mao. SAMPLES: Self adaptive mining of persistent lexical snippets for classifying mobile application traffic. In *ACM 21st International Conference on Mobile Computing and Networking (MobiCom)*, 2015.

[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 *USENIX Workshop on Offensive Technologies (WOOT)*, 2016.

[6] Y. Fu, H. Xiong, X. Lu, J. Yang, and C. Chen. Service usage classification with encrypted internet traffic in mobile messaging apps. *IEEE Transactions on Mobile Computing*, 15(11), 2016.

[7] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensic and Security*, 13(1), 2018.

[8] A. Dainotti, F. Gargiulo, L. I. Kuncheva, A. Pescapé, and C. Sansone. Identification of traffic flows hiding behind tcp port 80. In *2010 IEEE International Conference on Communications*, May 2010.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[10] Z. Wang. The applications of Deep Learning on traffic identification. *BlackHat USA*, 2015.

[11] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2017.

[12] M. Lotfollahi, R. Shirali, M. J. Siavoshani, and M. Saberian. Deep packet: A novel approach for encrypted traffic classification using deep learning. *preprint arXiv:1709.02656*, 2017.

[13] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access*, 5, 2017.

[14] R. Li, X. Xiao, S. Ni, H. Zheng, and S. Xia. Byte segment neural network for network traffic classification. In *IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2018.

[15] V. Tong, H. A. Tran, S. Souihi, and A. Mellouk. A novel QUIC traffic classifier based on convolutional neural networks. In *IEEE International Conference on Global Communications (GlobeCom)*, 2018.

[16] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. Mobile encrypted traffic classification using deep learning. In *IEEE/ACM Network Traffic Measurement and Analysis Conference (TMA)*, 2018.

[17] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *28th International Conference on Machine Learning (ICML)*, 2011.

[18] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust RGB-D object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[19] D. Ramachandram and G. W. Taylor. Deep multimodal learning: A survey on recent advances and trends. *IEEE Signal Processing Magazine*, 34 (6), 2017.

[20] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé. Mobile encrypted traffic classification using Deep Learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management*, 2019.

[21] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè. Multi-classification approaches for classifying mobile app traffic. *Journal of Network and Computer Applications*, 103, 2018.

[22] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic. Who do you sync you are? smartphone fingerprinting via application behaviour. In *6th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, 2013.

[23] 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 Conference on Communications and Network Security (CNS)*, 2015.

[24] H. F. Alan and J. Kaur. Can Android applications be identified using only TCP/IP headers of their launch time traffic? In *9th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, 2016.

[25] W. M. Shbair, T. Cholez, J. François, and I. Chrisment. A multi-level framework to identify HTTPS services. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2016.

[26] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani. Characterization of encrypted and VPN traffic using time-related features. In *2nd Int. Conference on Information Systems Security and Privacy (ICISSP)*, 2016.

[27] P. Wang, F. Ye, X. Chen, and Y. Qian. Datanet: Deep learning based encrypted network traffic classification in SDN home gateway. *IEEE Access*, 2018.

[28] Y.-C. Chen, Y.-J. Li, A. Tseng, and T. Lin. Deep learning for malicious flow detection. In *IEEE 28th International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017.

[29] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang. Automatic multi-task learning system for abnormal network traffic detection. *Int. Journal of Emerging Technologies in Learning*, 13(4), 2018.

[30] Z. Chen, K. He, J. Li, and Y. Geng. Seq2Img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks. In *IEEE International Conference on Big Data (Big Data)*, 2017.

[31] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng. Malware traffic classification using convolutional neural network for representation learning. In *IEEE International Conference on Information Networking (ICOIN)*, 2017.

[32] F. Chollet et al. Keras. `https://keras.io`, 2015.

[33] M. Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[34] L. Bernaille, R. Teixeira, and K. Salamatian. Early application identification. In *ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2006.

[35] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2016.

[36] W. De Donato, A. Pescapè, and A. Dainotti. Traffic identification engine: an open platform for traffic classification. *IEEE Network*, 28(2), 2014.