Contextual Counters and Multimodal Deep Learning for Activity-Level Traffic Classification of Mobile Communication Apps during COVID-19 Pandemic

Idio Guarino[a], Giuseppe Aceto[a], Domenico Ciuonzo[a], Antonio Montieri[a], Valerio Persico[a], Antonio Pescapè[a]

[a]*University of Napoli "Federico II", Italy*

Abstract

The COVID-19 pandemic has reshaped Internet traffic due to the huge modifications imposed to lifestyle of people resorting more and more to collaboration and communication apps to accomplish daily tasks. Accordingly, these dramatic changes call for novel traffic management solutions to adequately countermeasure such unexpected and massive changes in traffic characteristics.

In this paper, we focus on communication and collaboration apps whose traffic experienced a sudden growth during the last two years. Specifically, we consider nine apps whose traffic we collect, reliably label, and publicly release as a new dataset (MIRAGE-COVID-CCMA-2022) to the scientific community. First, we investigate the capability of state-of-art single-modal and multimodal Deep Learning-based classifiers in telling the specific app, the activity performed by the user, or both. While we highlight that state-of-art solutions reports a more-than-satisfactory performance in addressing app classification (96%–98% F-measure), evident shortcomings stem out when tackling activity classification (56%–65% F-measure) when using approaches that leverage the transport-layer payload and/or per-packet information attainable from the initial part of the biflows. In line with these limitations, we design a novel set of inputs (namely Context Inputs) providing clues about the nature of a biflow by observing the biflows coexisting simultaneously.

Based on these considerations, we propose Mimetic-All a novel early traffic classification multimodal solution that leverages Context Inputs as an additional modality, achieving $\geq 82\%$ F-measure in activity classification. Also, capitalizing the multimodal nature of Mimetic-All, we evaluate different combinations of the inputs. Interestingly, experimental results witness that Mimetic-ConSeq—a variant that uses the Context Inputs but does not rely on payload information (thus gaining greater robustness to more opaque encryption sub-layers possibly going to be adopted in the future)—experiences only $\approx 1\%$ F-measure drop in performance w.r.t. Mimetic-All and results in a shorter training time.

Keywords: communication apps, collaboration apps, COVID-19, Deep Learning, encrypted traffic, multimodal techniques, contextual counters, traffic classification.

1. Introduction

The outbreak of the Covid-19 pandemic has induced governments worldwide to impose lockdown periods during last two years. These events have forced millions of citizens to stay at home, and also study, work, and socialize from there if possible. As a consequence, Internet traffic from residential users has witnessed a significant growth (e.g., +20% EU Internet traffic volume) [1], also with implications on the mobility pattern of users in cellular networks—e.g., an increase in voice traffic and uplink traffic in suburbs (+10%) [2].

Focusing on *pandemic-related change of traffic composition*, Affinito et al. [3] highlighted the changes in the usage of different categories of applications for smart working and distance learning (i.e. *Video*, *SocialMedia*, *Messaging*, and *Collaboration Tools*) based on the investigation of websites and domains during the enforcement of social distancing measures. Analogous changes in usage and volume (increased use of collaboration platforms, VPNs, and remote desktop services) have been found analyzing traffic in campus networks [4, 5, 6, 7].

The sudden change in the spatial distribution, timing, and usage mix of online presence has had a *measurable impact on network performance* in terms of increased variability of delay, loss rate, and latency [8], and degradation of over-the-top services [9], to the point that these measurements have been effectively used for global-scale inference of work-from-home and lockdown events and maps [10]. Indeed, unexpected and massive changes in traffic characteristics pose a challenge to efficient network resource management, that in turn calls for enhanced network monitoring capabilities. More specifically, the possibility to infer the application or the type of application that generated the observed traffic (viz. the process of *Traffic Classification* [11], TC in the following) becomes paramount to most management, planning, and policy enforcement actions.

While TC has been a hard problem and an active field of research for decades, its application is further challenged by specific characteristics of communication and collaboration apps:

Email addresses: idio.guarino@unina.it (Idio Guarino), giuseppe.aceto@unina.it (Giuseppe Aceto), domenico.ciuonzo@unina.it (Domenico Ciuonzo), antonio.montieri@unina.it (Antonio Montieri), valerio.persico@unina.it (Valerio Persico), pescape@unina.it (Antonio Pescapè)

consistent use of encryption; shared application-level protocols as transport sublayers (namely, TLS and HTTP); different functioning modes (*activities*) for a single application; execution from mobile devices (platforms characterized by frequent and automated software updates). These challenges are now pushing toward the adoption of advanced *Deep Learning* (DL) approaches, able to cope with frequently changing input nature, and offering promising performance when dealing with complex and hard-to-model problems [12]. Thus, on the one hand, a better understanding is required of the traffic of applications that have seen a surge in utilization after the COVID-19 pandemic. On the other hand, an assessment (and improvement) of modern TC approaches is needed, applied to this specific scenario. This is the more the case with the mentioned peculiarity of *multiple activities* that the user can access from within the same application. This specific characteristic poses several problems: as we show in our experimental evaluation, different activities in the same app present different traffic patterns, while being similar to the same activity within other apps—this can likely confuse the classifiers and lead to poor performance (as we prove experimentally); different activities have different requirements in terms of Quality of Service, policy enforcement, and monitoring—managing traffic at app level overly extends the network management actions to be performed; analyses of users behaviors, to identify needs, and plan infrastructure and service deployments, are impacted by the coarseness of surveyed information.

Following these considerations, the objective of our work is to tackle the *activity-level early classification of network traffic generated by the most popular communication and collaboration mobile apps*, whose utilization has increased with the COVID-19 pandemic—and keeps shaping the nature of Internet traffic. Specifically, we target nine communication and collaboration apps (`Discord`, `GotoMeeting`, `Meet`, `Messenger`, `Skype`, `Slack`, `Teams`, `Webex`, and `Zoom`) that have seen dramatic increase in usage in correspondence of lockdowns. We analyze their traffic and assess the performance of the state-of-art DL approaches for classifying the specific app and/or the kind of activity (*Audio-call*, *Chat*, *Video-call*). As we the find results much wanting, informed by traffic characterization of mobile apps [13] we propose and evaluate a *novel set of inputs observed from contextual traffic from the same app*, and design a novel architecture exploiting these data as well, showing *much improved activity-classification performance*. Exploiting such novel inputs, we also design a novel DL architecture *which does not rely on application-layer payload*, with minimal loss of accuracy, but better training time and increased robustness to more opaque encrypted protocols. We remark that, although the present work is focused on internet traffic data generated by some of the apps whose use has grown significantly in conjunction with the COVID-19 pandemic, the proposed methodology is general. Indeed, it can be applied in all contexts in which it is crucial to classify the app and the activity performed by the user (e.g., QoS/QoE management and user profiling).

Accordingly, the *main contributions* of this work are summarized in the following:

- We collect and publicly release a novel dataset, named MIRAGE-COVID-CCMA-2022 (where CCMA stands for *Communication-and-Collaboration Mobile Apps*), encompassing the traffic of nine communication-and-collaboration mobile apps run by users executing three different activities (Audio-call, Video-call, and Chat). The collected dataset is human-generated, recent, and reliably labeled with both the *mobile app* that generated the traffic and the *activity* that was performed by the user. The considered apps have experienced a sudden change in volume and spatial/temporal patterns during the pandemic, thus are of specific interest for network operators, network managers, academia, and society at large.
- We experimentally evaluate the capability of state-of-art DL approaches in classifying the traffic generated by communication-and-collaboration apps at different granularity levels (i.e. application and activity), highlighting the shortcomings deriving from the feature leveraged by these solutions. Specifically, we apply state-of-art (and input-size-optimized) DL architectures (an 1D-CNN [14], a hybrid 2D-CNN+LSTM [15], and the multimodal MIMETIC-ENHANCED architecture [16]) to the collected dataset, to classify the traffic at *app*, *activity*, and *app*×*activity* granularity. For each activity, we analyze the traffic in terms of (payload-carrying) packet direction, payload length, payload content, TCP window size, and inter-arrival time, for the initial part of the biflow[1] (*early behavior* analysis) and point at the *limitations* of this choice in activity recognition.
- Prompted by the limitations in classifying user activities deriving by the inputs commonly adopted in related literature [12], we investigate the traffic patterns generated by a traffic source and design a novel set of inputs (namely *Context Inputs*) able to provide hints about a biflow by observing its context (i.e. the set of co-existing biflows which run in parallel). Hence, we perform a characterization analysis suggesting that Context Inputs are able to support user-activity classification.
- Capitalizing on the appeal of Context Inputs, we design a *novel classification solution* starting from MIMETIC-ENHANCED, named MIMETIC-ALL, which leverages them as an additional modality. Our MIMETIC-ALL effectively exploits the Context Inputs obtaining *always the best performance* when compared with both ML- and DL-based TC solutions fed with the same input, while being also suitable for *early traffic classification*.
- Exploiting the multimodal nature of the architecture, we provide also a *second novel classification solution*—we name MIMETIC-CONSEQ—that trades the inputs based on payload for the new Context Inputs, paying a negligible performance cost (less than 1% F-measure drop for activity classification w.r.t. MIMETIC-ENHANCED) to gain both a smaller training time and greater robustness to future more opaque encryption sublayers (e.g., TLS with *Encrypted*

[1] A biflow (or bidirectional flow) is defined as an aggregation of packets sharing the common 5-tuple (transport-layer protocol and destination/source IP address and transport-layer port) regardless of their direction.

Server Name Indication or *Encrypted Client Hello* extensions [17]).
- In addition to TC performance, we evaluate the reliability of the proposed novel classification solutions and considered variants via a calibration analysis proving the beneficial effect of Context Inputs in reducing the expected calibration error.

We remark that the present paper constitutes the expansion and continuation of the work described in the conference paper [18], which only analyzed *existing solutions* applied to the problem of app and activity classification (also evaluated on a smaller dataset).

The remainder of the manuscript is organized as follows. Section 2 surveys related studied analyzing changes of Internet traffic during Covid-19 pandemic, ML/DL approaches for app or app-user-activity identification, positioning our work against related literature. Section 3 describes the considered experimental setup, including the collected dataset used for analysis and validation. In Sec. 4, we highlight the shortcomings of current state-of-art (multimodal) DL-based approaches used for TC. Then, in Sec. 5, we introduce the Context Inputs and describe our novel proposals based on such inputs. The experimental analysis is provided in Sec. 6. Finally, Sec. 7 provides conclusions and future prospects. In Appendix A, we report the acronyms and abbreviations used in the manuscript.

2. Related Work

In this section, we discuss the recent studies focusing on the impact of Covid-19 pandemic (Sec. 2.1) and position our study by detailing the latest advancements in TC via DL (Sec. 2.2) and user activity recognition (Sec. 2.3).

2.1. Impact of Covid-19 on the Nature of Internet Traffic

Several studies have investigated the changes that the spread of the Covid-19 pandemic on a global scale has caused to Internet traffic as a result of lockdown periods and the consequent shift in daily habits. Regarding the increasing use of tools useful for smart working and distance learning, Affinito et al. [3] provide insights on the usage of different categories of Internet applications for collaboration and entertainment, by analyzing websites and domains visited during the enforcement of the lockdown to contain the spread of the virus. As shown, during the reference period, the most used applications were `Youtube`, `Netflix`, `Facebook`, `Whatsapp`, `Skype`, and `Zoom`. Other works investigate the variation in traffic volumes and network performance [1, 2, 4, 8, 9]. Specifically, Feldmann et al. [1] analyze the effect of the lockdowns on traffic shifts during the period Mar-Jun 2020, by using network flow data from multiple vantage points (i.e. an ISP, three IXPs, and a large academic network). They point out that during the period considered, the volume of European Internet traffic increased by up to +20%, mainly due to residential traffic generated by applications for work and distance education, whose volume increased by up to 200%. Lutu et al. [2] analyze the changes in mobility and their impact on cellular network traffic, showing an overall increase in voice traffic, with a decrease in download traffic (−20%) and an increase in uplink traffic (+10%). Additional degradations of network services are highlighted by Böttger et al. [9], who analyze the traffic growth in different regions of the world and how the network responded to an increased demand from the the perspective of edge network of `Facebook` during the beginning of the Covid-19 pandemic. Authors observed a world-wide increase in traffic throughput between March and July 2020 and also a correlation between the phase of traffic growth and the spread of Covid-19 for each region. Similar network traffic shifts are also investigated by Candela et al. [8] which highlight an higher variability in latency and loss rates in Italy during the first weeks of the 2020 lockdown w.r.t. the pre-pandemic period. Finally, Favale et al. [4] analyze the impact of lockdown measures and the switch to online collaboration and e-learning solutions on a university campus during the months of March/April 2020, showing a peak of 1.5Gbit/s in the university network traffic due to the increased use of digital tools for collaboration and remote working.

2.2. Deep Learning-based (Mobile) Traffic Classification

In recent years, several works have faced TC via DL approaches. Wang [19] has first used a Stacked AutoEncoder (SAE) for unencrypted traffic identification, achieving superior performance w.r.t. standard neural networks ($\geq$ 90% precision and recall). **Encrypted TC** is targeted by Wang et al. [14], who propose a method based on 1D Convolutional Neural Network (1D-CNN)—outperforming the 2D variant—to tackle four different TC tasks: (i) VPN/nonVPN, (ii) 6 encrypted traffic classes, (iii) 6 VPN-tunneled traffic classes, and (iv) 12 encrypted applications. Similar tasks are tackled by Lotfollahi et al. [20] proposing *Deep Packet* (based on 1D-CNN and SAE), able to outperform ML-based classifiers for encrypted TC at packet granularity. This proposal outperforms ML-based classifiers in both application identification and traffic characterization. Recurrent Neural Networks have been considered by Lopez-Martin et al. [15], proposing different hybrid DL architectures that combine Long Short-Term Memory (LSTM) and 2D-convolutional layers. Zeng et al. [21] propose a framework for **encrypted TC** and **intrusion detection**—named *Deep-Full-Range*—based on three different DL architectures (i.e. CNN, LSTM and SAE) using raw traffic as input data. The framework was evaluated on both classification tasks by comparing performance with state-of-the-art methods on two public datasets obtaining better performance on both tasks.

Focusing on the classification of **mobile-app traffic**, Rezaei et al. [22] leverage a CNN fed with the header and the payload of the first six packets of a biflow. Similarly, Liu et al. [23] devise *FS-Net*, an encoder-decoder architecture based on Bidirectional Gated Recurrent Units (BiGRU) taking as input IP-packet sizes of flow sequences. In this context, in our previous work [12], we define a systematic framework to dissect the encrypted mobile TC using DL, and compare a number of the aforementioned techniques for a comprehensive evaluation. Common usage of biased inputs (e.g., local-network meta-

data [14], or source and destination ports [15]) inflating TC performance is also discussed (and discouraged).

Multimodal DL solutions have been recently proposed to face the challenges of mobile-app TC. We propose MIMETIC [24], a general framework for capitalizing the heterogeneous views associated with a traffic object, along with a novel training procedure based on pre-training and fine-tuning. Experimental results show that the MIMETIC classifier outperforms single-modal, ML-based, as well as late-combination of traffic classifiers both in terms of TC performance and training complexity. Following along the same research direction, Wang et al. [25] propose *App-Net*, consisting of two modalities: a (bidirectional) LSTM and a 1D-CNN. Experimental results show that App-Net outperforms ML-based and single-modal DL-based traffic classifiers, while performing almost on par w.r.t. MIMETIC. In the same research direction, we propose DISTILLER [26], a multimodal multitask DL approach for traffic classification to capitalize on the heterogeneity of traffic data and solving multiple traffic categorization problems simultaneously. A specific instance of the proposed framework was experimentally compared with state-of-the-art multitask DL traffic classifiers [27, 28] on the public dataset ISCX VPN-nonVPN, showing DISTILLER achieves gains over all the TC tasks considered, also exhibiting very manageable training complexity and lower computational burden than the overall-best-performing multitask baseline. Recently, Akbari et al. [29] have proposed a tripartite multimodal DL architecture based on convolutional and LSTM layers for encrypted-traffic classification at service (HTTPS traffic) and application (QUIC traffic) level. Each modality is fed with different input data, namely *(i)* raw TLS handshake bytes, *(ii)* flow time-series of IAT, size, and direction, and *(iii)* handcrafted (post-mortem) flow statistics. Unfortunately, the authors (*i*) do not compare their proposal with other multimodal solutions, (*ii*) include the adoption of both handcrafted and post-mortem flow statistics, and (*iii*) do not consider Context Inputs in the design of the proposal.

Capitalizing on the latest advancements on TC solutions via DL, we investigate the performance of a state-of-art multimodal architecture and compare its performance against some other recent but simpler proposals.

2.3. User Activity Recognition

While app-level classification of mobile app traffic is just a—especially harder—case of network-traffic classification, modern apps characterized by *multiple usage modes (activities)* offer on the one hand an even harder problem, on the other hand the possibility of obtaining more actionable information w.r.t. just the sole indication of the app. Therefore, recent academic studies have investigated and demonstrated *the ability to infer user actions* performed in mobile apps by analyzing encrypted network traffic. To this aim, in our previous work [30] we have addressed the **characterization and modeling** (by means of multimodal Markov Chains) **of network traffic** (at trace, activity, and flow level) generated by apps that have experienced a traffic surge due to COVID-19 pandemic spread. Our results highlight interesting traffic peculiarities related to both the apps and the specific activities they are used for.

Conti et al. [31] propose a framework to **infer which specific actions** the users perform while running a certain **mobile app**, based on packet direction/size information. This is achieved by using both supervised (Random Forest, RF) and unsupervised learning (agglomerative clustering with a distance based on dynamic time warping) approaches for service burst classification. It is shown that *by knowing the app generating the traffic*, it is possible to identify a user action with $\geq 95\%$ accuracy for most of the actions considered within a set of 7 Android apps. Saltaformaggio et al. [32] tackle a similar task via their Netscope proposal: the evaluation is carried out considering a set of 35 popular activities (for both Android and iOS devices), based on statistics originated from IP headers. For elementary-behavior discovery, K-means clustering is used, and then a Support Vector Classifier (SVC) is trained/tested on activity-behaviors binary mapping, resulting in performance that varies depending on the device being tested, but averages 78.04% precision and 76.04% recall. Grolman et al. [33] extend the feature-extraction method proposed in [31] and employ transfer learning (based on a modified co-training method requiring only few samples in the source domain) to transfer patterns that allow identifying specific in-app activities (i.e. tweets and posts) across different configurations (i.e. device- or version-wise) to improve the process of recognizing user actions utilizing existing unlabeled encrypted data. The classification of user actions in target configuration is made by using two co-training learners based on RF and AdaBoost algorithms.

Aiolli et al. [34] focus on **identifying user activities** on smartphone-based **Bitcoin wallet apps**, leveraging the earlier methodology proposed in [35]. The fingerprints are collected by *running apps automatically* on Android and iOS devices and *simulating user actions* using scripted commands. 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 IP address/port pairs. The authors deal with the classification problem in a multi-stage hierarchical fashion to infer the app category (Bitcoin vs. other), the OS (Android vs. iOS), the app, and finally the specific action performed by the user. The results, evaluated on 29 apps (9 Bitcoin) and 7 user actions, report 95% accuracy.

Li et al. [36] address the problem of **inferring activities** from a targeted set of mobile apps via analyzing the **continuous encrypted user traffic stream**. The authors propose a DL-based framework in which, focusing on activities having duration > 15 s, they proceed by dividing each traffic stream into segments using a sliding-window approach, where each segment corresponds to an activity of a targeted app. Subsequently, the segments are normalized and represented by a time-space matrix and a traffic spectrum vector that are used to feed 2D-CNN and 1D-CNN branches of a multimodal DL architecture, respectively. The proposed solution is compared with state-of-art classifiers on a *semi-synthetic* traffic dataset and attains $> 95\%$ accuracy regarding app, activity, and both classification tasks.

Finally, Li et al. [37] propose a two step strategy method for **mobile-service TC**. In detail, in the first step, a joint DL

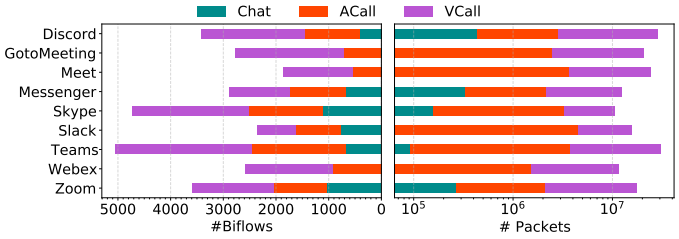


Figure 1: Communication and collaboration apps in MIRAGE-COVID-CCMA-2022 (alphabetic order). Performed activities, number of biflows (*left*-bar) and number of packets (*right*-bar) are reported for each app. Note that the log scale is used to report the packets number.

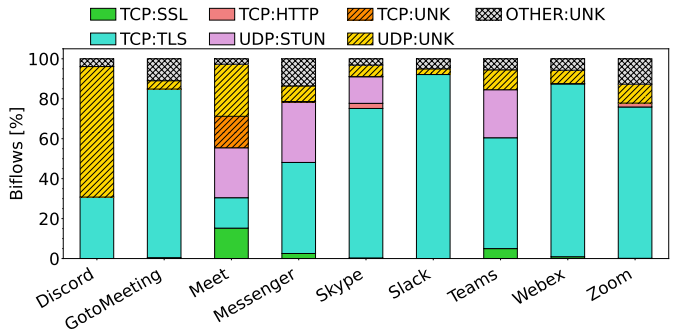


Figure 2: Protocol distribution in terms of biflows. *UNK* stands for *unknown*, SSL stands for *undetected* version of *SSL/TLS*.

model is exploited as a basic classifier to observe the mobile service traffic from multiple timescale (i.e. micro-time interval, short-time interval, and long-time interval). Specifically, based on the time scale, the basic classifier leverages a different architecture—including Logistic Regression (LR), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN)—to extract information from the features used to feed the model. In the second step, an attention mechanism is used to aggregate the basic predictions made by the first step in order to observe the mobile service traffic in an extra-long-time scale. Experimental results show that the two step strategy outperform pure DL strategies when used in the classification of 7 different services (e.g., video on demand, video call, live stream, chat), both in terms of accuracy and time delay.

Starting from the aforementioned literature on app and activity recognition, we investigate their identifiability via a real dataset targeting apps that were massively used due to the pandemic events, by avoiding handcrafted features and post-mortem identification.

3. Experimental Setup

In the following, we first provide the details about the MIRAGE-COVID-CCMA-2022 dataset we collected (Sec. 3.1), also explaining the rationale behind the selection of the apps (Sec. 3.2). Then, we introduce the considered classifiers in the experimental analysis (Sec. 3.3).

3.1. Dataset Collection and Ground Truth Generation

The MIRAGE-COVID-CCMA-2022 dataset was collected by students and researchers during Apr.–Dec. 2021 leveraging the MIRAGE architecture [38] (conveniently optimized to capture the traffic of communication and collaboration apps) in the ARCLAB laboratory at the University of Napoli "Federico II".[2] Experimenters used three *different* mobile devices (all equipped with Android 10): a Google Nexus 6 and two Samsung Galaxy A5. In each capture session the experimenters performed a *specific activity* on a *given communication and collaboration app* (details are given in the later Sec. 3.2), so as to obtain a traffic dataset that reflects the common usage of the considered apps.[3] The session duration spanned from 15 to 80 minutes based on the specific activity being carried out. Accordingly, each session resulted in a PCAP traffic trace with associated ground-truth information obtained via additional system log-files.

Based on the latter, each biflow[4] was *reliably labeled* with the corresponding *Android package-name* by considering established network-connections (via the standard Linux command `netstat`[5]). This information was further enriched with a custom label referring to the specific activity performed by the user operating the device.

To foster replicability and reproducibility we *publicly release* the MIRAGE-COVID-CCMA-2022 dataset[6].

3.2. Apps' and Activities' Selection Rationale

Communication and collaboration apps—used for business meetings, classes, and social interaction—have experienced a huge utilization increment when "stay-at-home" orders were issued worldwide and are still widely used due to the change of life- and work-style of people around the globe. The wide adoption of these apps exposing complex network behaviors has prompted recent research from academics and practitioners focusing on different facets, ranging from reaction to varying network conditions [39] to the different usages of RTP/RTCP protocols [40] for the multimedia transfers.

Based on both popularity and utilization boost, herein we focus on *nine* communication and collaboration apps: `Discord`, `GotoMeeting`, `Meet`, `Messenger`, `Skype`, `Slack`, `Teams`, `Webex`, and `Zoom`. Indeed, `Zoom` has obtained the steepest increment with its traffic scaling by orders of magnitude, followed by `Webex`, `GotoMeeting`, `Teams`, `BlueJeans` (whose traffic we are currently collecting), and `Skype` [41]. Also, during 15th–21st March 2020, `Zoom` was downloaded 14×, 20×, and 55×

[2] We highlight that the captures were carried out by adhering to the distancing/mask-wearing rules prescribed by regional/national decrees in force at the moment of the collection.

[3] Each traffic capture session has been performed with the up-to-date version of the app. Also, to limit background traffic, network access has been disabled for all the apps but the one under test.

[4] A bidirectional flow (biflow) encompasses all the packets sharing the same 5-tuple (i.e. source and destination IP address, source and destination port, and transport-level protocol) in both upstream and downstream directions [12].

[5] `https://linux.die.net/man/8/netstat`

[6] `http://traffic.comics.unina.it/mirage/mirage-covid-2022`. In detail, we release the dataset in two formats, making available both the raw traffic data captured (in *JSON* format) and a pre-processed version providing the set of inputs leveraged in this specific work (in *pickle* format).

more than the weekly average during Q4 2019 in the US, UK, and Italy, respectively [42]. Similarly, Teams also experienced significant growth in Italy (resp. France) with 30× (resp. 16×) more downloads. The considered apps have been extensively exploited for remote (and blended) teaching in Italian[7] and European[8] institutions and universities. As also highlighted by Sandvine [43], in 2021 video traffic has proven to be even more significant compared to the previous year, both as a standalone and as an embedded component of app mashups. Indeed, social and communication applications have gained further popularity, with WhatsApp, Zoom, Teams, and Messenger being the most used for *messaging*, and Zoom, Webex, and Teams for *enterprise conferencing*.

Figure 1 depicts the mobile apps collected in MIRAGE-COVID-CCMA-2022 and used in this study, highlighting also the activities carried out with each, and the amount of traffic collected in terms of number of biflows and packets. Specifically, according to the observed app usage, the experimentation covered the following *activities* (all related to live events): *Chat ("Chat")*—involves just two participants exchanging textual messages and/or multimedia content (e.g., images or GIFs); *Audio-call ("ACall")*—involves just two participants transmitting only audio; *Video-call ("VCall")*—involves many attendees which can transmit both video and audio (e.g., live events such as video calls between two or more attendees or webinars).

Furthermore, Fig. 2 reports the characterization of MIRAGE-COVID-CCMA-2022 traffic in terms of the adopted protocols for each app. Specifically, for all the apps a significant percentage of SSL/TLS biflows is observed (ranging from 30% for Discord to 90% for Slack). Moreover, all apps generate a relevant portion of UDP biflows. Remarkably, in the case of Discord (unlike the other apps) the UDP traffic is predominant compared to TCP counterpart (i.e. 65% vs. 30%). Finally, for Meet, Messenger, Skype, and Teams there is also a significant presence of STUN[9] biflows (between 10% and 30%), commonly used for multimedia communications in the prevalent case of presence of a Network Address Translator (NAT). These findings are consistent with the outcomes of traffic analysis performed in other studies [40, 39].

3.3. Baselines Considered and Learning Setup

In the following, we consider state-of-the-art classifiers selected among the best single-modal and multimodal alternatives—based on extensive performance evaluation carried out in our previous works [12, 24, 16]—in terms of both DL architecture and *unbiased* input data.

Specifically, we consider an 1D-CNN fed with the first N_b bytes of transport-layer payload (PAY) of each biflow [14]. Also, we evaluate a hybrid composition of *2D-CNN+LSTM* (named Hybrid hereinafter) as proposed in [15], having as input the following informative fields (SEQ) of the first N_p packets of each biflow: (*i*) the number of bytes in transport-layer payload (PL), (*ii*) TCP window size (TCPWIN, set to zero for UDP packets), (*iii*) inter-arrival time (IAT), and (*iv*) packet direction (DIR) $\in \{-1, 1\}$.

Finally, we also consider the *multimodal* Mimetic-Enhanced classifier [16], being an *enhanced* version of the generic Mimetic framework originally proposed in [24]. Mimetic-Enhanced consists of two modalities fed each with one of the two input types (namely PAY and SEQ) used for the baseline single-modal classifiers described above. First, to augment the information carried by input data, Mimetic-Enhanced improves the original Mimetic by introducing a *trainable* embedding layer for both modalities. From the architectural viewpoint, the modality fed with the PAY input type consists of two 1D convolutional layers—each followed by a 1D max-pooling layer—and a final dense layer. On the other hand, the modality fed with the SEQ input type consists of a BiGRU and one dense layer. Finally, the intermediate features extracted by each modality are concatenated and fed to a shared dense layer and then to the last softmax classifier. Mimetic-Enhanced is trained via a two-phase procedure consisting of the pre-training of individual modalities plus the fine-tuning of the whole architecture, and adopts an adaptive learning-rate scheduler. We refer to [16] for further details on the Mimetic-Enhanced architecture and related hyperparameters.

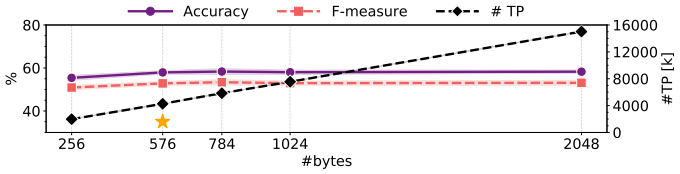
In all the following analyses, the performance evaluation is based on a *stratified ten-fold cross-validation*. Indeed, the latter represents a solid assessment setup since it keeps the sample ratio among classes for each fold. Also, to foster a fair comparison, all the classifiers (both novel proposals and baselines, cf. Sec. 5.3 for details on our proposals) are trained for a total of 100 epochs (for Mimetic-Enhanced and the novel multimodal proposals, we consider 30 epochs for pre-training of each modality and 40 epochs for fine-tuning) for minimizing the categorical cross-entropy loss, and exploit the Adam optimizer (with a batch size of 50) and a validation-based early-stopping technique to prevent overfitting. In detail, for each fold, we consider 80% of the whole training data as the actual training set, while the remaining and 20% is assigned to the validation set.

Finally, hereinafter we take advantage of the ground-truth information associated to each biflow (labeled with both the app generating the traffic and the activity performed) to *instruct and evaluate different supervised strategies* corresponding to *three* TC tasks: *(i)* classifying the app (App-TC), *(ii)* classifying the activity (Act-TC), and *(iii)* classifying both the app and the activity (Joint-TC). Accordingly, we *train* the above models with: *(a)* app-related ground truth only (App), *(b)* activity-related ground truth only (Act), and *(c)* the joint app-and-activity ground truth (App×Act). Note that App and Act produce classifiers able to address only the specific task they are trained for (i.e., classifying either apps or activities), whereas when training the DL architectures based on App×Act, *all* three TC tasks above-described can be addressed.

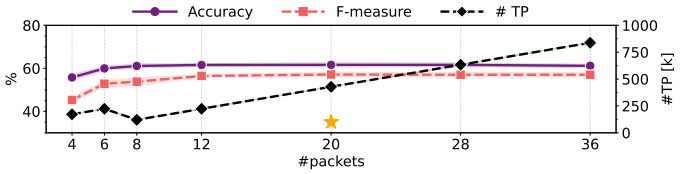
[7] Fondazione CRUI – COVID-19 | Strumenti per la didattica digitale.

[8] European University Institute – Software available at the EUI.

[9] Session Traversal Utilities for NAT (STUN) protocol allows an end-point to determine the IP address and port assigned to it by a NAT [44]



(a) 1D-CNN.



(b) HYBRID.

Figure 3: Accuracy [%], F-measure [%], and number of trainable parameters of 1D-CNN (a) when varying the input dimensions N_b and HYBRID (b) when varying the input dimensions N_p. Results refer to the Joint-TC task. The best trade-off value is highlighted via a ⋆ marker.

4. Empirical Evaluation of State-of-the-Art Shortcomings in Activity Classification

After tuning the size of their inputs (Sec. 4.1), we assess the capability of state-of-art solutions in tackling classification at both app and activity granularity (Sec. 4.2). Then, we look at traffic patterns to understand the causes of the poor performance attained for activity TC (Sec. 4.3) and investigate alternative paths viable to highlight peculiarities in the traffic generated by different activities, considering its temporal evolution (Sec. 4.4), and aggregate behavior (Sec. 4.5).

4.1. Sensitivity Analysis

First, we perform a sensitivity analysis to tune the dimension of the above types of input employed, i.e. the number of bytes N_b and number of packets N_p. For brevity, we refer to the Joint-TC task, i.e. the hardest in its nature.

Figure 3 shows the *accuracy* and *F-measure*[10] attained by the 1D-CNN and HYBRID architectures, when varying $N_b \in [256, 2048]$ B and $N_p \in [4, 36]$ packets, respectively.[11] We also report how the number of trainable parameters (TP) varies with the size of the considered input data to highlight the (necessary) input size-complexity trade-off. As reported in Fig. 3a, although the best performance in terms of accuracy and F-measure is obtained with $N_b = 784$ B, when passing from $N_b = 576$ B to $N_b = 784$ B, both the accuracy and F-measure remain stable despite the larger input size. Additionally, the same variation of N_b causes a non-negligible increase in the number of trainable parameters (+1.5M), resulting in a much more complex architecture with a limited improvement (i.e. $\leq 0.5\%$ in terms of F-measure).

On the other hand, regarding Fig. 3b, we can notice that HYBRID reaches the best performance in terms of both accuracy and F-measure by using $N_p = 20$ packets. In this specific case, considering 20 packets instead of 12 still provides a small improvement of performance in terms of F-measure, at the cost of a slight increase (and thus manageable) of complexity (+200k in terms of number of trainable parameters).[12] Regarding the latter, it is useful to underline that the number of trainable parameters corresponding to $N_p \in [4, 6]$ is comparable to that obtained by using $N_p = 12$ packets. The reason for this can be traced to the fact that such a small input implies the use of different padding, which causes additional complexity to implement the HYBRID architecture. *Hence, in the following we employ $N_b = 576$ B and $N_p = 20$ packets in order to keep the trade-off between classification performance (in terms of F-measure) and complexity of the obtained classifiers. Additionally, it is apparent that there is an upper bound on the achievable TC performance even optimizing the input size, whose causes are deepened in the next.*

4.2. App and Activity Classification via State-of-the-art Approaches

Hereinafter, we evaluate the performance achieved by state-of-the-art classifiers (i.e., 1D-CNN, HYBRID, and MIMETIC-ENHANCED), when adopting different training strategies (i.e., APP, ACT, and APP×ACT). We recall that APP (resp. ACT) is able to solve only the App-TC (resp. Act-TC) task, whereas APP×ACT allows to solve both each separate task and the Joint-TC problem.

As a result, Tab. 1 reports the performance of the classifiers in terms of accuracy and F-measure attained for each TC task (column-wise). The table is also complemented by a computational complexity assessment (via the "TP" and the training "Time" columns).[13] By comparing the *general performance* achieved on the three TC problems, Joint-TC clearly confirms to be the *most difficult* task to tackle (with 58% – 67% accuracy and 53% – 62% F-measure ranges) due to the greater number of classes (i.e. the 24 combinations of apps and activities). In contrast, the (easier) App-TC task (consisting of 9 classes) can be effectively solved by all the architectures (94% – 99% accuracy and 95% – 99% F-measure). Finally, the performance obtained on the Act-TC task highlights the *actual difficulty* in activity recognition. Indeed, despite the smallest number of classes (i.e. the 3 activities) considered in our study, very low performance (60% – 68% accuracy and 56% – 65% F-measure) are attained with the all state-of-art approaches.

[10] accuracy is the share of correctly-classified samples, while F-measure is the harmonic mean of precision (the proportion of classifier decisions for a given class which are actually correct) and recall (the per-class accuracy).

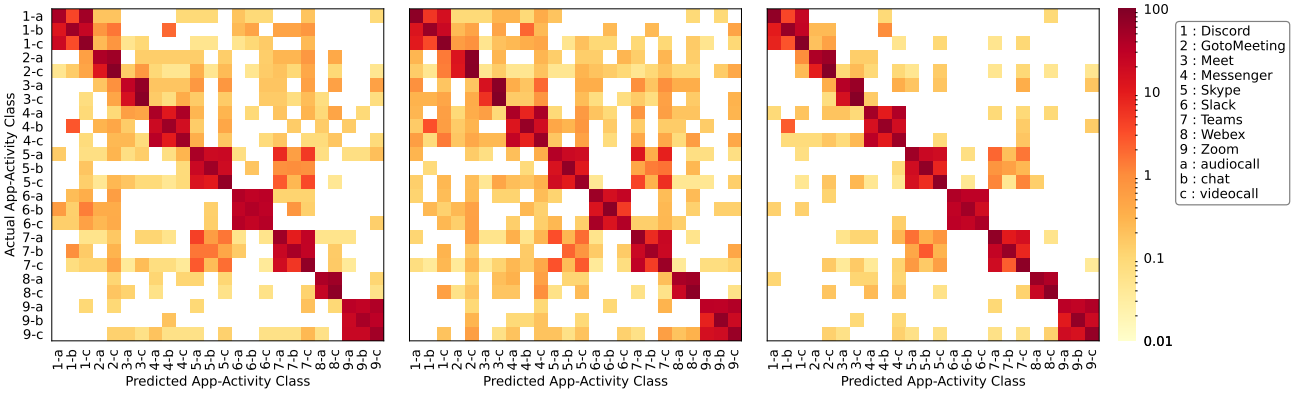
[11] We did not perform the same analysis also for the multimodal MIMETIC-ENHANCED due to the combinatorial complexity resulting from considering all the possible combinations of N_b and N_p and the time needed to train/test each resulting configuration.

[12] Moreover, we underline that the number of trainable parameters of HYBRID is dominated by that of 1D-CNN, being the former more than one order of magnitude smaller than the latter. This consideration also applies to all architectures and single-modality branches fed with SEQ compared to those fed with PAY.

[13] In the case of MIMETIC-ENHANCED, since the training methodology adopted allows to parallelize the pre-training of the two modalities, we have calculated the resulting training time by adding the time needed to perform the pre-training of the "slower" modality and the time needed to perform the successive fine-tuning of the whole architecture.

Table 1: Comparison of accuracy, F-measure, number of Trainable Parameters (#TP), and Training Time (Time) for the three state-of-art DL architectures (1D-CNN, HYBRID, and MIMETIC-ENHANCED) when trained on different class-labels (i.e. related to APP×ACT, APP, and ACT) for different classification tasks. #TP slightly varies with the classification task (with variations being smaller than reported precision). Results are in the format *avg*.(±*std*.) obtained over 10-folds. Training time was computed by pre-training the individual modalities in parallel. The best result per metric (column) is highlighted in boldface. **MM** denotes a multi-modal architecture.

Classifier	MM	Training Strategy	Joint-TC		App-TC		Act-TC		#TP [k]	Time [*min*]
			Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]		
1D-CNN	○	APP×ACT	57.94(±0.82)	52.84(±0.86)	96.12(±0.30)	96.66(±0.25)	59.95(±0.77)	56.12(±0.80)	4272	47(±5)
		APP	-	-	97.48(±0.21)	98.05(±0.23)	-	-	4256	45(±4)
		ACT	-	-	-	-	60.18(±1.04)	56.45(±1.17)	4250	47(±3)
HYBRID	○	APP×ACT	61.62(±0.93)	57.12(±1.04)	94.36(±0.41)	95.11(±0.36)	64.77(±0.96)	63.13(±0.86)	428	**12**(±**4**)
		APP	-	-	94.85(±0.51)	95.62(±0.42)	-	-	**426**	15(±4)
		ACT	-	-	-	-	63.99(±0.90)	62.06(±1.05)	**426**	**12**(±**4**)
MIMETIC-ENHANCED	●	APP×ACT	**67.12**(±**1.14**)	**62.29**(±**1.21**)	98.54(±0.21)	98.75(±0.18)	**67.94**(±**1.13**)	**65.33**(±**1.15**)	1235	57(±6)
		APP	-	-	**98.73**(±**0.18**)	**98.95**(±**0.16**)	-	-	1225	42(±3)
		ACT	-	-	-	-	65.37(±0.74)	62.60(±1.05)	1221	49(±4)



(a) 1D-CNN. (b) HYBRID. (c) MIMETIC-ENHANCED.

Figure 4: Confusion matrices of 1D-CNN (a), HYBRID (b), and MIMETIC-ENHANCED (c) considering the APP×ACT and related to the Joint-TC. Note that the log-scale is used to evidence small errors.

By looking at classifier standpoint, MIMETIC-ENHANCED *outperforms* the two competing approaches on all the three considered TC tasks. For instance, MIMETIC-ENHANCED roughly achieves +5% (resp. +10%) F-measure than HYBRID (resp. 1D-CNN) on Joint-TC. Interestingly, Tab. 1 also highlights that the training strategy adopted may impact the performance achieved by the models. Indeed, while for App-TC all the architectures achieve better performance when relying on APP training strategy, for Act-TC the training APP×ACT results in better performance for MIMETIC-ENHANCED and HYBRID. Hence, this highlights that app classification may be conducive to activity recognition, while the opposite does not necessarily hold.

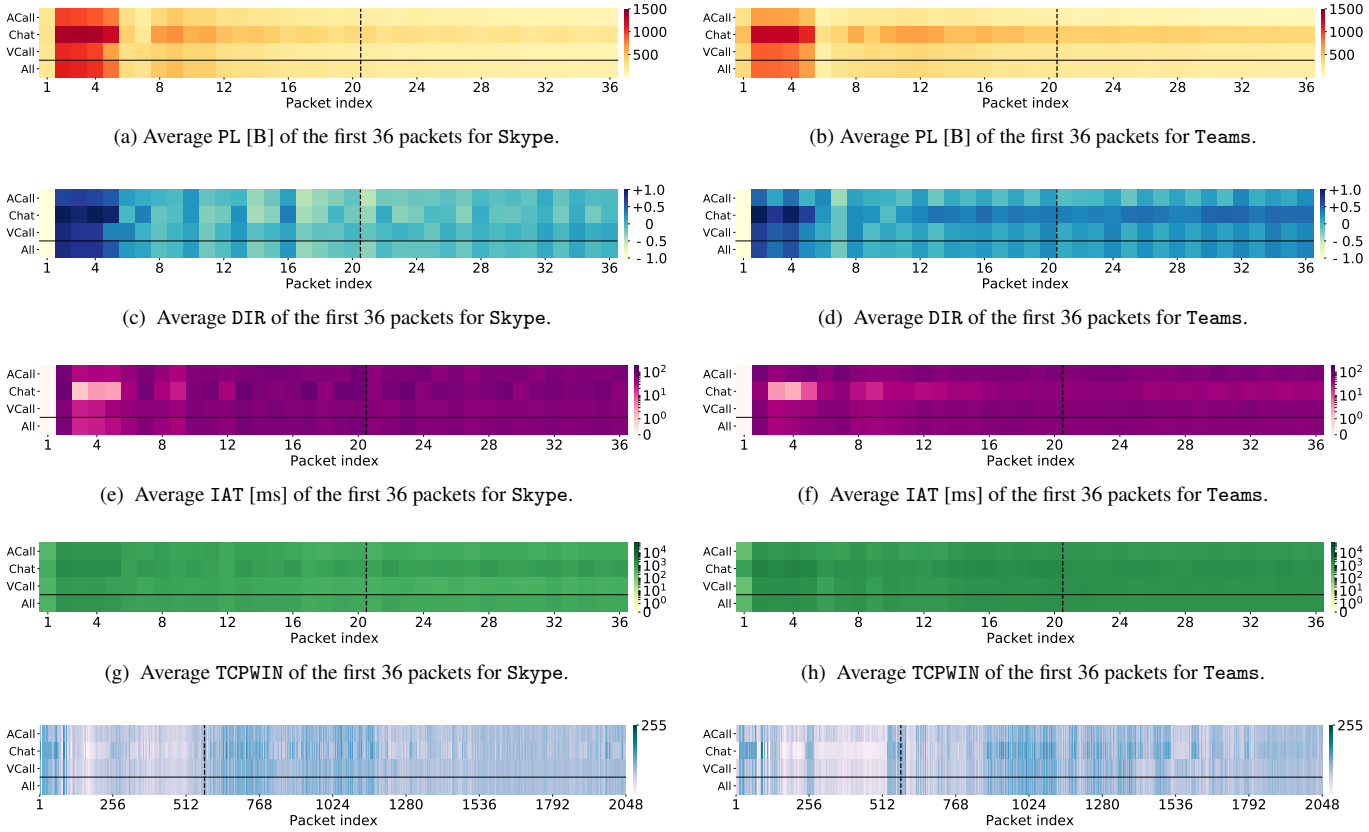
Furthermore, looking at the *complexity of the considered architectures*, the analysis provides other interesting pieces of evidence (last two columns of Tab. 1). In fact, the complexity highly varies with considered architecture: 1D-CNN and MIMETIC-ENHANCED are ≈ 10× and ≈ 4× more complex than HYBRID, respectively, in terms of trainable parameters (which are roughly proportional to both the training time and the memory occupation). *Hence,* MIMETIC-ENHANCED *provides the best trade-off between TC performance and complexity.*

To provide details on the performance at a finer grain, Fig. 4 reports the *confusion matrix* of each architecture on the Joint-TC task. Delving into these matrices, we can notice that MIMETIC-ENHANCED can substantially reduce the misclassification patterns w.r.t. both 1D-CNN and HYBRID, confining the errors within the activities of the same app, illustrated by the more evident block diagonal pattern in Fig. 4c. Still, the *block-diagonal* pattern of all the confusion matrices confirms *the general difficulty in discriminating adequately among the activities with the given set of inputs*.[14]

4.3. *Understanding the Causes: Biflow-level Characterization of Early Behavior*

In the previous analysis, we have shown that state-of-art classifiers achieve *unsatisfactory performance* when used to solve the activity classification problem. Consequently, in this section we analyze the typical behavior of biflows in the initial part of the communication, in order to collect clues about whether activities related to specific apps can be discriminated by observ-

[14]This is also confirmed by the confusion matrices on the Act-TC task, not shown for brevity.



(a) Average PL [B] of the first 36 packets for Skype.

(b) Average PL [B] of the first 36 packets for Teams.

(c) Average DIR of the first 36 packets for Skype.

(d) Average DIR of the first 36 packets for Teams.

(e) Average IAT [ms] of the first 36 packets for Skype.

(f) Average IAT [ms] of the first 36 packets for Teams.

(g) Average TCPWIN of the first 36 packets for Skype.

(h) Average TCPWIN of the first 36 packets for Teams.

(i) Average PAY of the first 2048 Bytes for Skype.

(j) Average PAY of the first 2048 Bytes for Teams.

Figure 5: Properties of biflows' time-series with respect to PL, DIR, IAT, TCPWIN, and PAY for Skype (a, c, e, g, i), Teams (b, d, f, h, j), and Webex (k, l, m, n, o) based on the *activity*-type and in summary (*All*) form. The downstream and upstream DIR is mapped on +1 and −1, respectively. For each input type, the vertical dashed line marks the size of the input fed to classifiers, based on the sensitivity analysis in Sec. 4.1.

ing the inputs used to feed the considered DL architectures. Accordingly, we analyze the PL/DIR/IAT/TCPWIN sequences of the first 36 packets[15] (i.e. the fields considered in the SEQ input) and the first 2048 B (i.e. the PAY input). For each packet/byte index, we report the average value across all biflows for a given app. The analysis presented in Fig. 5 focuses on Skype[16], Teams[17], and Webex[18]. For these apps, the above information is broken down into the considered activities (*ACall*, *Chat*, and *VCall*) and also reported in summary form (*All*).

Considering the generic behavior of the specific app (*All*), in all cases there are significant differences between the initial part—which extends at most up to the 15^{th} packet—and the remaining part of the sequence, when considering PL/DIR/IAT. This does not occur in the case of TCPWIN, for which the behavior is slightly less evident. Specifically, in the cases of Skype and Teams, the main patterns are located on the first 5 packets, while for Webex the trend extends up to the 15^{th} packet.

Moreover, when focusing on the specific app, similarities can be seen between the patterns associated with the different activities. For Skype and Teams we observe characteristic patterns associated with *ACall* and *VCall* on the first 5 packets while for *Chat* we observe a trend that extends over all 36 packets. Specifically, focusing on *VCall* and *ACall*, if we consider PL and DIR, in the case of Skype we observe identical behavior between the two activities (excluding the 5^{th} packet) while in the case of Teams the pattern associated with *VCall* is slightly more pronounced (i.e. packets with higher payload in the downstream direction). On the contrary, this does not hold when looking at the IAT for which we observe longer times in the case of *ACall*. It is also interesting to note that in the case of Webex, *ACall* and *VCall* present an identical behavior and this is probably due to the fact that Webex[19], unlike the other two shown, is an app designed to allow users to make video calls and this function has been adapted to make simple audio calls, since it does not have a native function to perform this type of activity. Finally, referring to PAY, similar observations can be made about the indistinguishability between the activities associated with Skype, Teams, and Webex, especially regarding the

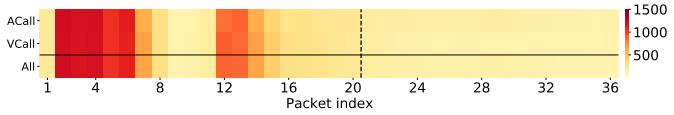
[15] Packets with no payload are discarded since they reflect transport-layer signaling neither depending on the nature of the app nor the performed activity.

[16] A similar behavior as Skype has been observed for Messenger, Slack and Zoom.

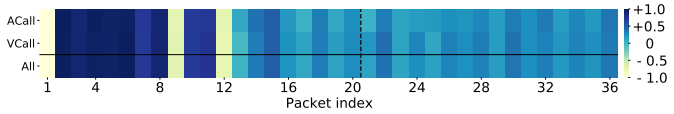
[17] A similar behavior as Teams has been observed for Discord.

[18] A similar behavior as Webex has been observed for GotoMeeting and Meet.

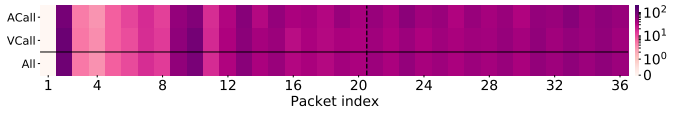
[19] The same reasoning applies to Meet and GotoMeeting.



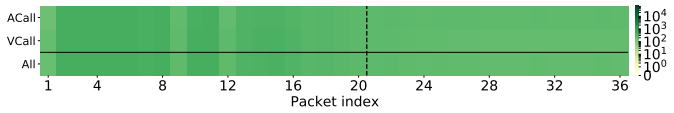
(k) Average PL [B] of the first 36 packets for Webex.



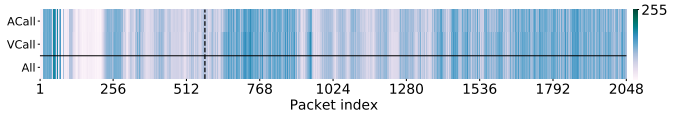
(l) Average DIR of the first 36 packets for Webex.



(m) Average IAT [ms] of the first 36 packets for Webex.



(n) Average TCPWIN of the first 36 packets for Webex.



(o) Average PAY of the first 2048 Bytes for Webex.

Figure 5: Properties of biflows' time-series with respect to PL, DIR, IAT, TCPWIN, and PAY for Skype (a, c, e, g, i), Teams (b, d, f, h, j), and Webex (k, l, m, n, o) based on the *activity*-type and in summary (*All*) form. The downstream and upstream DIR is mapped on +1 and −1, respectively. For each input type, the vertical dashed line marks the size of the input fed to classifiers, based on the sensitivity analysis in Sec. 4.1.

ACall and *VCall*.

Different activities performed by the user using a specific app exhibit very similar behaviors when considering the features that are typically used as input to feed the classifiers.

4.4. Highlighting Activity Peculiarities via an Alternative View: Time Evolution of App Connections

The previous analysis suggests that the shortcomings encountered in TC performance are related to the nature of the considered inputs, which are not able to capture the peculiarities of the activities performed. Hence, here we investigate whether alternate paths are viable based on the observation of "simultaneous" (viz. concurrent) biflows.

Indeed, the behavior of an app on the network is the result of the mixture of multiple concurrent traffic biflows between the client—running a certain (communication and collaboration) app—and a variety of servers [13], whose number and characteristics may depend both on the app and the activity. To validate this hypothesis, we analyze the *overall traffic* generated or received by the app during a traffic capture in order to quantify the number of concurrent biflows over time, by considering (non-overlapping) windows.

Formally, for a capture starting at time t_0 and having duration D, the i^{th} aggregation interval gathers all biflows that transmit at least one packet within $[t_0 + (i-1)\Delta, t_0 + i\Delta)$, with $i \in \{1, 2, \cdots \lceil \frac{D}{\Delta} \rceil\}$, where Δ is the duration of the samplig window. Considering three exemplifying apps (i.e. Skype, Teams, and Webex), Fig. 6 shows the amount of concurrent biflows considering a (non-overlapping) window size $\Delta = 5\,\text{s}$. We can observe a limited time frame at the very beginning of the communication during which the client generates several biflows whose number depends on the type of action conducted by the user (this is particularly evident for instance in Fig. 6b). These concurrent interactions might be connected to authorization/accounting, telemetry, and advertising services, each expected to be located on dedicated servers.[20]

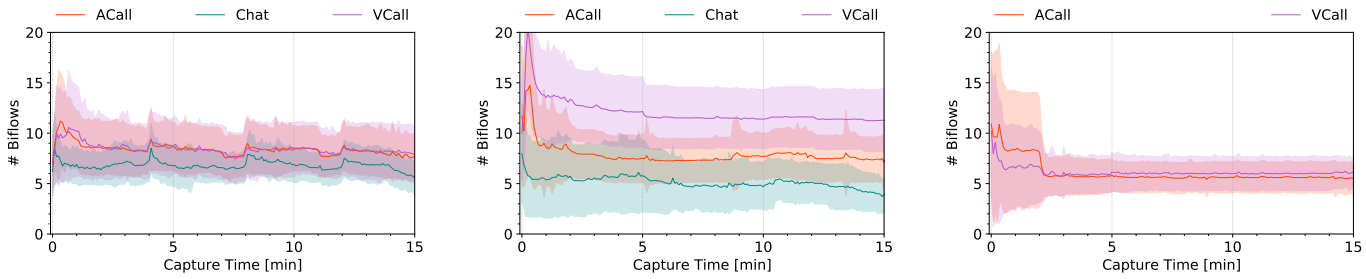
The number of generated biflows then settles to a lower value (which varies with both the app and the activity, and ranges from 5 to 13 biflows). Interestingly, for some apps such as Teams, this plateau value appears to be more stable for both *Acall* and *Vcall*, whereas the *Chat* exhibits more pronounced variations across multiple captures. Generally, if we compare the number of biflows associated to *ACall* and *VCall* before reaching the plateau, we can observe that in the case of Teams, the *VCall* is characterized by a higher number of biflows than *ACall*, while the opposite scenario occurs in the case of Webex.

Observing the steady-state behavior, *VCall* tends to maintain a higher number of active biflows than *ACall* in the case of Teams, while the two activities tend to maintain the same number of active biflows for Webex. Interestingly, the behavior of the two activities is very different for Teams—with *VCall* always being associated with a higher number of active biflows. This does not hold for Skype where the behavior is very similar in both phases. Finally, for both Skype and Teams, the traffic related to *Chat* is always carried by a lower number of biflows compared to the other two activities.

We also evaluated the number of different server sockets contacted by the client, which characterize the packets belonging to the same *service burst* [45]. The definition of service burst starts from that of *burst* [46], defined as a sequence of packets (regardless of the biflow they belong to) having an inter-packet time smaller than a given threshold. Hence, a service burst is defined as the subset of packets of a burst belonging to biflows that share the same *transport protocol, and destination IP:port pair* The service burst ideally groups packets related to the same server application and strictly-related information (due to the timing), and has been used previously for the purpose of TC [31, 47, 45], implying also that the client device is set (as in our analysis). At any time, a service burst can refer to a single server socket, not including traffic related to other simultaneous communications from the same client (see Fig. 8).

Performing an analysis of concurrent server sockets (strictly analogous to what we have done with biflows) we can observe trends similar to those shown in Fig. 6 for biflows but with a

[20] An experimental analysis of 500 popular mobile apps conducted in [13] has found an average of 4.5 different cloud services accessed per mobile app.



(a) Skype. (b) Teams. (c) Webex.

Figure 6: Amount of concurrent biflows (*mean* ± *std* across different captures) in each 5-second slot for Skype (a) and Teams (b), and Webex (c) across the different activities performed. Values are calculated considering the first 15 min of each capture.

notable offset. Specifically, we found that the number of server sockets are at most 50% of the overall number of active biflows. This clearly implies that multiple biflows concurrently reach the same service. *Hence, the presence of a significant number of concurrent biflows and multiple concurrent service bursts imply that a single service burst (not to say a single biflow) cannot account for most traffic exchanged by a given app at a give time interval. Therefore, contextual information is missed when using biflow-based or service-burst-based inputs.* This (expected) experimental finding will inform the definition of novel inputs in the following Sec. 5, and motivates the aggregated characterization of app traffic in the following.

4.5. Highlighting Activity Peculiarities via an Alternative View: Characterization of Simultaneous Biflows

Informed by the analysis on time evolution of app communications, in this section we characterize the traffic carried by concurrent biflows in terms of aggregate metrics, namely *bit-* and *packet-rates*, computed considering a (non-overlapping) window of size $\Delta = 5\,\text{s}$ according to the aforementioned methodology. Results are shown in Figs. 7a and 7b, depicting the downstream and upstream bit-rate, respectively. Similarly, Figs. 7c and 7d report the downstream and upstream packet-rate, respectively. In addition, for each app, the distribution is broken down across the specific activities highlighted with different colors.

Observing the **downstream bit- and packet-rates** (in Figs. 7a and 7c), we can notice that the different activities are characterized by significantly-different patterns in terms of median rate which, however, do not seem strictly related to a specific app. In fact, for all apps *VCall* always corresponds to the highest bit-rate (resp. packet-rate) which varies in the range $[651, 821]$ Kbps (resp. $[115, 182]$ pps). This is expected given the *simultaneous transmission of both audio and video traffic*. On the contrary, in the case of *ACall* and *Chat*, the bit-rate (resp. packet-rate) varies in the ranges $[46, 61]$ Kbps (resp. $[33, 55]$ pps) and $[2, 3]$ Kbps (resp. $[0.6, 0.8]$ pps), respectively.

In contrast, when **upstream bit- and packet-rates** (in Figs. 7b and 7d) are considered, there is no clear separation as in the downstream case. In fact, the different activities tend to have *closer median values*, varying in the range $[3, 68]$ Kbps (resp. $[1, 54]$ pps) for the bit-rate (resp. packet-rate) and depending more on the type of app. Indeed, for both Skype and Teams we observe *differences* between *ACall* and *VCall*. On the contrary, focusing on Webex, the same two activities are characterized by *very similar bit- and packet-rates*, i.e. 26 vs. 29 Kbps and 20 vs. 21 pps, respectively. Moreover, referring to *Chat*, the activity pattern seems quite distinguishable from *ACall* and *VCall* for *both* rate metrics (cf. Figs. 7b and 7d) in the case of Teams. Conversely, this does not hold for Skype, for which the *Chat* bit-rate is much more similar to that of *VCall*.

5. Context-Based Multimodal Deep Learning Traffic Classification

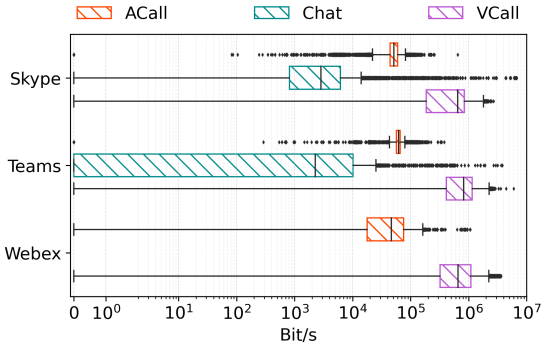
Hereinafter, we introduce the definition of context behavior and related Context Inputs in Sec. 5.1; the traffic characterization of Context Inputs is then provided in Sec. 5.2. Finally, Sec. 5.3 describes the novel Mimetic-All architecture proposed herein.

5.1. Definition of Context Behavior

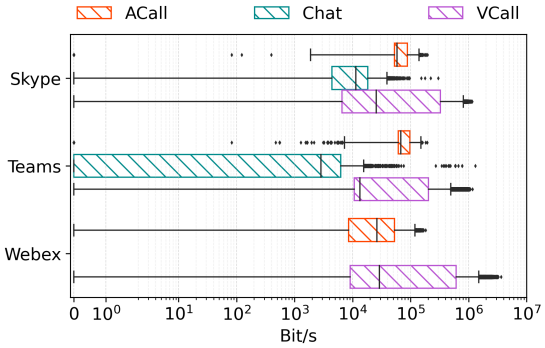
Based on above considerations, in addressing the TC problem at activity level we intend to exploit *both* the (*a*) intrinsic characteristics of the biflow to be classified and (*b*) the information related to its context, namely taking into account the traffic related to the biflows that are contextually created by the same app when the user performs a specific activity. To be able to do this in a practically meaningful and useful manner, we need to clarify the concepts of *classification object* and contextual information.

A classification object represents the (aggregated) unit of traffic that will receive the classification label—and thus will be subject to the management actions that motivated the classification, e.g., it will be throttled, or blocked, or given higher priority, or will trigger a special logging rule, etc. Typically, the classification object is also the source of input data for the classifier. This is the case for the biflow, whose information is used to extract classification features, and is the typical unit of traffic for network management.

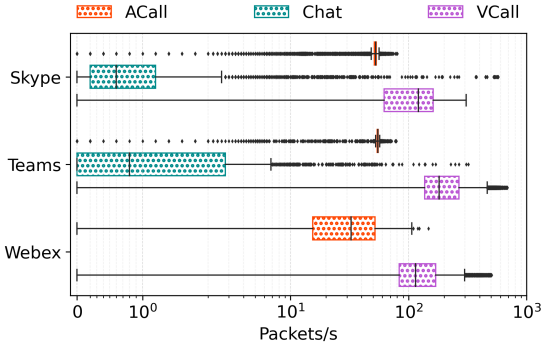
Considering the *service burst*, defined in Sec. 4.4, and recently used in the context of TC as *classification object* [47, 31, 45], we notice that this aggregate of traffic presents however



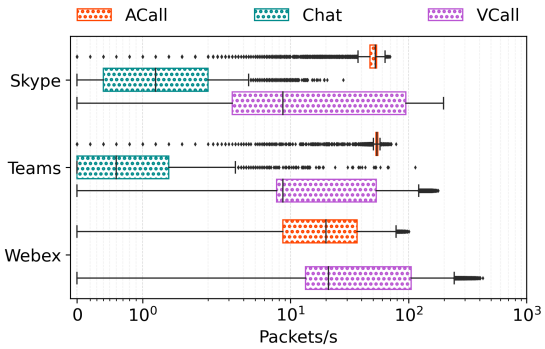
(a) Downstream bitrate.



(b) Upstream bitrate.



(c) Downstream packet-rate.



(d) Upstream packet-rate.

Figure 7: Downstream bitrate (a), upstream bitrate (b), downstream packet-rate (c), and upstream packet-rate (d). Values are evaluated over time intervals of $\Delta = 5$ s. Boxes report the 1^{st} and 3^{rd} quartiles (1Q and 3Q, respectively), while whiskers mark 1Q−1.5 IQR and 3Q+1.5 IQR, where IQR=3Q−1Q. Black diamonds highlight outliers.

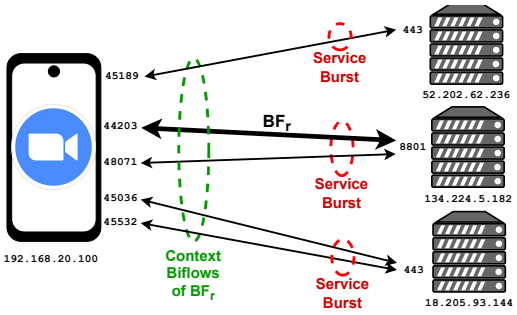


Figure 8: Comparison between *context biflows of a reference biflow* BF_r and *service bursts*. All packets are generated or received by the same mobile app.

some significant *limitations*, discussed hereafter. First, the (service) burst definition is highly-sensitive w.r.t. the choice of the inter-packet-time threshold: this value can critically depend on the network conditions, and on the presence (or lack thereof) of multiple clients accessing the same service. Secondly, using a service burst as the TC object poses a practical problem on the usage of the classification result: from both network management and network security standpoints the biflow is a well-known and widely-used granularity to apply relevant actions such as filtering, throttling, priority queuing, accounting. Conversely, it is not straightforward (or necessarily meaningful) to apply the same actions on varying sets of packets separated by the aforementioned transmission gaps. Finally, service bursts are unlikely to reflect (and allow to exploit) the heterogeneous nature of modern-application traffic. Indeed, a single application to perform its functions can *at the same time* communicate with different network services [13], and a mix of services is likely to be more indicative of a specific *activity* of the same application. In Secs. 4.4 and 4.5 we experimentally validated the presence of multiple simultaneous biflows, and several service bursts, that differently characterize the time evolution and volume of traffic of each app and activity.

Following these considerations, **we keep the biflow as the TC object**, and we will use the information from simultaneous same-app communications as *contextual information* that will help discerning the activities of a given application. Specifically, in what follows we refer to the TC object as the *reference biflow* or BF_r. We consider the traffic generated by the same device, identified by its IP address[21] hereafter referred to as *device IP*, and the same app generating BF_r: a filter on the *device* IP address and a pre-classification stage detecting the app[22] allow to select all biflows potentially related to BF_r.

[21] If NAT is performed, the original IP is easily available if the traffic capture happens on the gateway performing NAT, or if the log of NAT mapping is provided: both cases are common in enterprise scenarios. If this information is not explicitly available, different approaches to cluster traffic originating from a single device can be applied to passively monitored NATted traffic [48].

[22] For app-level classification, an F-measure greater than 92% can be achieved also just using the single-modality bSEQ classifier fed with informative fields of the first 20 packets (see Tab. 3). On the basis of the results depicted in Fig. 3b, this can be considered a design choice depending on deployment constraints.

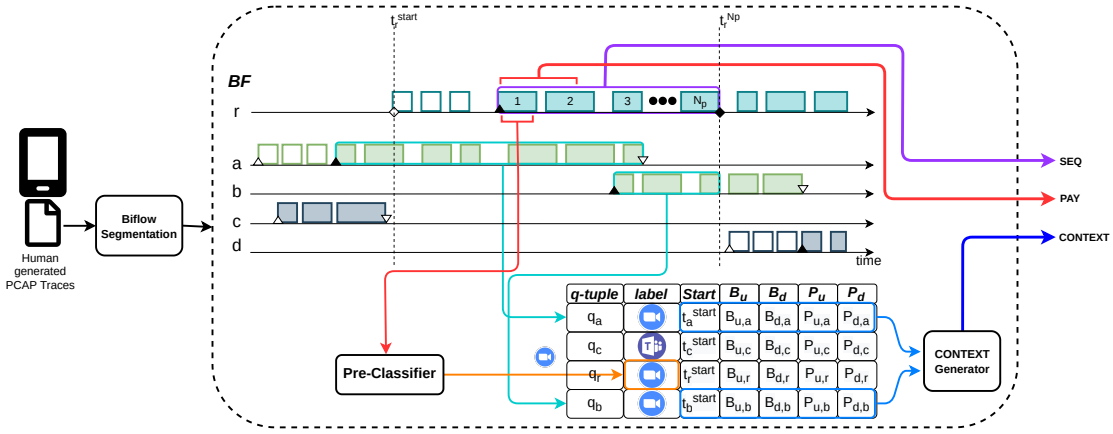
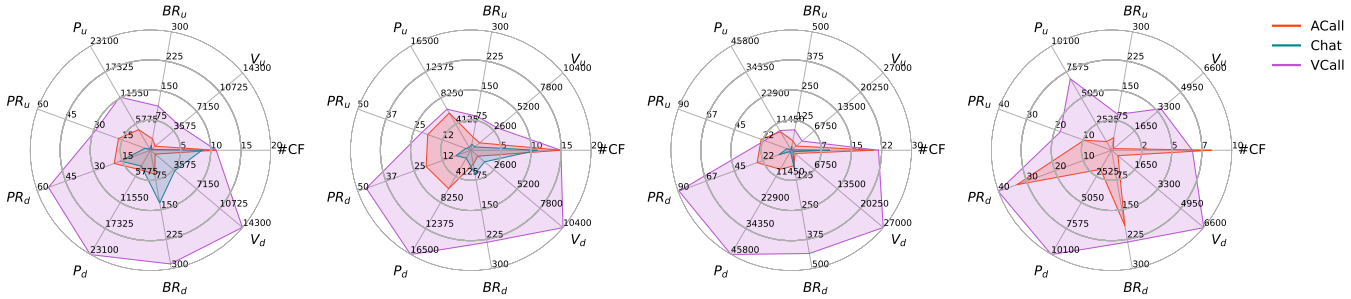


Figure 9: For each biflow are highlighted: the arrival time of the first packet (△) (i.e. the SYN packet for TCP biflows), the arrival time of the last packet (▽), and the arrival time of first packet with *non-zero* payload (▲). Additionally, for the current biflow $\mathtt{BF}_r$, in addition to the arrival time of the first packet (◊), the arrival time of the $\mathtt{P}^{\mathtt{th}}$ packet with *non-zero* payload is also highlighted (⧫). **PAY** denotes the first $\mathtt{N}_b$ byte of transport-level payload. **SEQ** denotes header fields extracted from the sequence of the first $\mathtt{N}_p$ packets. **CONTEXT** denotes Context Inputs computed at the arrival of the $\mathtt{N}_p$-th packet of the reference biflow $\mathbf{BF}_r$ (i.e., the biflow to be classified). $\mathbf{t}_i^{\mathbf{start}}$ refers to the starting time of the generic biflow $\mathbf{B}_i$ identified by the quintuple $\mathbf{q}_i$. $\mathbf{B_{u},}_i/\mathbf{B_{d},}_i$ and $\mathbf{P_{u},}_i/\mathbf{P_{d},}_i$ denote the total amount of byte and packets, respectively, transmitted/received by the biflow $\mathbf{B}_i$, up to time $\mathbf{t}_r^{\mathbf{N}_p}$.



(a) All. (b) Skype. (c) Teams. (d) Webex.

Figure 10: Properties of biflows' with respect to the *nine* Context Inputs (i.e., $\mathtt{V_u}, \mathtt{V_d}, \mathtt{BR_u}, \mathtt{BR_d}, \mathtt{P_u}, \mathtt{P_d}, \mathtt{PR_u}, \mathtt{PR_d}, \mathtt{\#CF}$, arranged symmetrically in the upper half for upstream, lower half for downstream) for All apps (a), Skype (b), Teams (c), and Webex (d) based on the activity-type (*ACall*, *Chat*, and *VCall*). Values are calculated at the arrival of the 20^{th} packet for each current biflow and averaged over all biflows. $\mathtt{V_u}$ and $\mathtt{V_d}$ are reported in KB. $\mathtt{BR_u}$ and $\mathtt{BR_d}$ are reported in Kbps. $\mathtt{PR_u}$ and $\mathtt{PR_d}$ are reported in Pps. $\mathtt{P_u}$, $\mathtt{P_d}$, and #CF are reported in unit.

To restrict the contextual information to only communications simultaneous to BF_r, and to impose *causality* (thus enabling online classification), we further restrict considered packets to biflows that were open during the transmission of the first N_p payload-carrying packets of BF_r. In summary, naming t_r^{start} the arrival time of the SYN packet of BF_r, and $t_r^{N_p}$ the arrival time of its $\mathtt{N_p}$-th payload-carrying packet, the set of packets defining *contextual biflows* satisfy *all* the following *four* conditions:

- same *device* IP of BF_r;
- same app label of BF_r;
- biflow did not end before t_r^{start};
- arrival time of the packet precedes $t_r^{N_p}$.

The overall mechanism is illustrated in Fig. 9 and described hereafter. For each biflow, the starting time and the current number of bytes/packets transmitted (in both the upstream and downstream directions) are saved. Then, for each reference biflow BF_r, at time $t_r^{N_p}$ (arrival of its $\mathtt{N_p}$-th payload-carrying packet), the packets belonging to its contextual biflows are used to compute *nine* aggregate metrics to be used as *Context Inputs* (Context in short). Specifically, these metrics correspond to: (a) the number of contextual biflows (#CF), (b) the amount of transmitted byte/packets ($\mathtt{V}_*/\mathtt{P}_*$) and (c) the bit-/packet- rate[23] ($\mathtt{BR}_*/\mathtt{PR}_*$) in both directions (we use $* = \mathtt{u}$ and $* = \mathtt{d}$ to denote the *upstream* and *downstream* directions, respectively).

Regarding the practical feasibility of the definition above, we highlight that the Context Inputs are time-sliced analogous to flow counters kept by routing devices and traffic monitoring middleboxes (*NetFlow* and *IPFIX* standard [49]), and can be directly derived from their values sampled at two instants determined by each reference biflow (the start and the arrival of N_p-th packet).

[23] Rates are computed by considering the time interval between the starting of the oldest contextual flow and $t_r^{N_p}$.

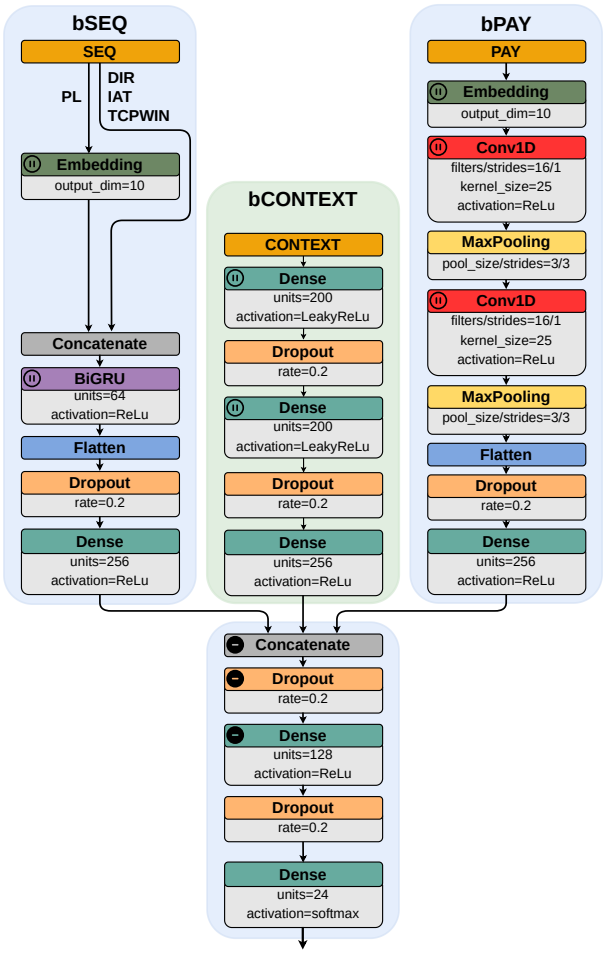


Figure 11: Proposed MIMETIC-ALL classifier. The macro-blocks shared with the original MIMETIC-ENHANCED architecture are highlighted in blue. The green macro-block denotes the new branch (i.e., bCONTEXT) dedicated to the novel set of *Context Inputs*. For each single block the color filling indicates the type of layer. The ⛔ symbol characterizes the layers that are not part of the architecture when considering the single-modality (bPAY, bSEQ, or bCONTEXT) alone. The ⏸ symbol marks layers frozen during the fine-tuning phase.

5.2. *Analysis of the Context Inputs*

In this section, we use the Context Inputs previously defined above to characterize the traffic generated when the user performs one of the considered activities (i.e. *ACall*, *Chat*, and *VCall*), taking into account also the specific app. To this end, in Fig. 10, for each activity, we report the average value of each feature calculated in correspondence of the 20^{th} packet of the *reference biflow* as a radar chart. In detail, Fig. 10a focuses on activities without taking into account the generating app whereas Figs. 10b- 10d provide a drill-down for `Skype`, `Teams`, and `Webex`, respectively.

As shown in Fig. 10a, despite the (average) number of contextual biflows is similar (`#CF`), the different activities show very different behaviors w.r.t. the other Context Inputs. In fact, as expected, *VCall* represents the activity originating *the highest contextual traffic for both directions*, in terms of packets and bytes. This is mainly due to the simultaneous transmission of audio and video traffic streams. Also, comparing *Chat* and *ACall*, we notice that the former presents a predominance of downstream traffic (in terms of both volume and byte-rate) whereas the latter presents a more balanced traffic that stands out especially for the quantity and rate of packets transmitted in the upstream direction.

Focusing on the downstream direction, we notice that for both `Skype` and `Teams` (cf. Figs. 10b and 10c) *VCall* still continues to have the highest amount of traffic both in terms of bytes and packets. Furthermore, for `Webex` (cf. Fig. 10d), we notice a similar behavior between *VCall* and *ACall* in terms of packet- and byte-rate, whereas this does not occur when considering the number of packets and byte volume received, which are higher in the case of *VCall*. Moreover, when comparing *ACall* and *Chat* in the case of `Skype` and `Teams`, the two activities differ mostly in terms of packets, whereas if we consider the byte volume, the difference becomes less evident, especially for `Teams`.

On the other hand, by considering the upstream direction, we notice a clear pattern for *Chat* that results in a smaller amount of traffic w.r.t. all the Context Inputs. Conversely, this does not apply to *ACall* and *VCall*. In fact, for `Skype` and `Teams` (cf. Figs. 10b and 10c), there is a similarity between the two activities in terms of number of packets transmitted and packet-rate. However, this is not true when looking at the byte volume and the byte-rate. Finally, in the case of `Webex` (cf. Fig. 10d), a clear distinction between *ACall* and *VCall* w.r.t. all the Context Inputs is evident.

5.3. *Leveraging Context Inputs: the* MIMETIC-ALL *Architecture*

In Sec. 4.2, we have shown that the state-of-art MIMETIC-ENHANCED [16] is able to outperform the considered single-modality classifiers, especially when addressing classification tasks that take into account the activities performed by the users. In this section, taking advantage of the modularity offered by the general MIMETIC framework [24], we describe the design of MIMETIC-ALL to exploit the Context Inputs discussed in Sec. 5.1.

As shown in Fig. 11, the proposed MIMETIC-ALL architecture consists in *three per-modality* (viz. input-specific) branches, henceforth named simply bPAY, bSEQ, and bCONTEXT (where the initial "b" stands for "branch"). As in the original proposal [16], bPAY and bSEQ branches take as input the first N_b bytes of the transport layer payload (`PAY`) and the informative fields extracted from the sequence of the first N_p packets (`SEQ`), respectively (cf. Sec. 3.3 for details on such input types). Additionally, both branches exploit a *trainable embedding layer* to embed each input element into a vector of dimension $e = 10$, resulting in an overall embedding matrix $E \in \mathbb{R}^{N \times e}$—with N denoting the input dimensionality (i.e. N_b and N_p for the `PAY`- and `SEQ`-modality, respectively). The motivation for applying an embedding to these inputs is due to the categorical nature of `PAY` and `SEQ` and for providing a better representation (i.e. more informative) of the inputs.

Specifically, in the bPAY branch, the corresponding embedding matrix is fed to a sequence of single-modality layers, consisting of two 1D convolutional layers—with a kernel size of

25, unit stride, and 16 and 32 filters, respectively—each followed by a 1D max-pooling layer—with spatial extent of 3 and unit stride—and finally, one dense layer with 256 neurons. On the other hand, the layers of the bSEQ branch are a bidirectional GRU (BiGRU)—with 64 units and return-sequences behavior—and one dense layer with 256 neurons. All the layers are set with the *Rectified Linear Unit (ReLU)* activation function. Besides the above branches, the newly-added bCONTEXT branch takes as input the contextual (aggregated) inputs associated with the N_p-th packet of each reference biflow B_r (`Context`). Indeed, as opposed to `PAY` and `SEQ`, these aggregated inputs do not have a natural ordering or sequentiality. In detail, inspired by the proposal of Akbari et al. [29], we have used a Multi-Layer Perceptron (MLP) network to ingest (viz. distill information from) them. This choice is driven by the fact that, unlike `PAY` and `SEQ`, the Context Inputs are calculated in an aggregate form and, having no natural ordering or sequentiality, there is no spatial/time evolution dependence that could be exploited by convolutional/recurrent layers. Specifically, bCONTEXT consists of two dense layers—characterized by 200 neurons and a *LeakyReLU* activation function. Similar to the other two branches, we also use a final dense layer with 256 units and a *ReLU* activation function. Finally, the features extracted by the single-modality branches are joined via a concatenation layer and fed to a (dense) *shared representation layer*—with 256 neurons and *ReLU* activation—before performing the classification through a softmax.

As in the original proposal, MIMETIC-ALL is trained via a two-phase procedure consisting of an independent *pre-training* of each single-modality branch followed by a *fine-tuning* of the entire architecture after freezing the lower single-modality layers (i.e., the dense, 1D convolutional, and BiGRU layers, as highlighted in Fig. 11 via the ⏻ symbol). In more detail, we performed the pre-training of each branch and the fine-tuning for 30 and 40 epochs, respectively, to minimize respective categorical cross-entropy loss functions via the ADAM optimizer (with a batch size of 50).[24] Additionally, as opposed to the MIMETIC-ENHANCED proposal [16], preliminary investigation showed that using a fixed learning-rate (i.e. equal to 0.001) instead of an adaptive one results in a performance improvement for Joint-TC.[25] Finally, to improve regularization and mitigate the possible overfitting, we added a dropout of 0.2 at the end of each single-modality branch and after every dense layer, and adopted an early-stopping technique measured on the validation accuracy with the same setup described in Sec. 3.3 (i.e. 20% of training data are used for validation).

Remarks on hyperparameter choice: during the design process of the bCONTEXT branch, we evaluated several combinations regarding the tuning of the activation functions and the number of neurons of the dense layers. Specifically, regarding the former, we obtained slightly better performance by using a *LeakyReLU* on the first two layers and a *ReLU* on the last one. Specifically, the *LeakyReLU* allows for a small, non-zero gradient when the unit is saturated and not active compared with the *ReLU*, alleviating potential problems caused by the hard activation of the latter [50]. Conversely, trying different configurations[26] we obtained the best trade-off—between performance and complexity—by setting 200 neurons on the first two dense layers. Moreover, we noticed better performance when considering the same number of neurons (i.e., 256) in the final dense layers of the single modalities before concatenation.

Table 2: Variants of classifiers used in this work with respective input data used to feed them.

Variant	`PAY`	`SEQ`	`Context`
bPAY	✓	—	—
bSEQ	—	✓	—
bCONTEXT	—	—	✓
MIMETIC-ENHANCED	✓	✓	—
MIMETIC-CONPAY	✓	—	✓
MIMETIC-CONSEQ	—	✓	✓
MIMETIC-ALL	✓	✓	✓

`PAY`: First N_b bytes of transport-layer payload;
`SEQ`: Informative header fields extracted from the sequence of the first N_p packets;
`Context`: Context Inputs computed at the arrival of the N_p-th packet.

6. Experimental Evaluation

As shown in Sec. 4.2, the sole combination of biflow transport-layer payload (`PAY`) and packet-level (`SEQ`) inputs within MIMETIC-ENHANCED classifier *is not sufficient to guarantee adequate performance* when dealing with Joint-TC and Act-TC tasks. Thus, in Sec. 5 we identified a new set of inputs—which we named *Context Inputs*—suitable to distinguish traffic associated with different activities performed with the same app and proposed the MIMETIC-ALL classifier leveraging them (Sec. 5.3).

In this section, we evaluate the impact of this context-based modality on TC performance when adopting the APP×ACT training strategy. The latter strategy indeed allows tackling *all the considered classification tasks* (i.e. App-TC, Act-TC, and Joint-TC).

For the sake of a complete evaluation and aiming at performing a *reasoned ablation study*, other than MIMETIC-ENHANCED and MIMETIC-ALL, in the following analysis we investigate the performance of classifiers obtained by selecting a subset of the three modalities available. For readers' convenience, in Tab. 2 we report the different instances drawn from the MIMETIC framework according to the considered type of input (i.e. `PAY`, `SEQ`, and `Context`).

[24] The single-modality branches, when considered alone (i.e. not in a multi-modal configuration), are trained for 100 epochs without dividing the training process into *pre-training* and *fine-tuning*.

[25] In addition, we have also tried a hybrid configuration—i.e. consisting of an adaptive learning rate during the pre-training of `PAY` and `SEQ` and the fine-tuning, and a fixed on bCONTEXT—obtaining a lower performance compared to the fixed setting.

[26] We tested 32, 64, 128, 200, and 256 neurons.

Table 3: Accuracy, F-measure, number of Trainable Parameters (#TP), and Training Time (Time) comparison of DL-based traffic classifiers when combining different mixes of modalities. Models are trained using App×Act class labels. Results are in the format *avg.* (±*std.*) obtained over 10-folds. The best result per metric (column) is highlighted in boldface. Training time is calculated by pre-training the individual modalities in parallel. The input types fed to each classifier are shown in Tab. 2.

Classifier	Joint-TC		App-TC		Activity-TC		#TP [k]	Time [*min*]
	Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]		
bPAY	42.34 (±1.12)	35.80 (±1.25)	76.27 (±0.84)	77.22 (±0.82)	53.57 (±0.70)	46.83 (±1.46)	460	22 (±3)
bSEQ	57.56 (±0.80)	51.37 (±1.12)	91.03 (±0.59)	92.36 (±0.53)	62.13 (±0.96)	59.12 (±0.91)	706	26 (±7)
bContext	66.44 (±1.58)	64.68 (±2.05)	81.12 (±1.53)	80.64 (±1.67)	78.91 (±0.91)	77.75 (±0.97)	**99**	**2** (±**0**)
Mimetic-Enhanced	67.12 (±1.14)	62.29 (±1.21)	98.54 (±0.21)	98.75 (±0.18)	67.94 (±1.13)	65.33 (±1.15)	1235	57 (±6)
Mimetic-ConPay	78.09 (±0.99)	76.94 (±0.98)	95.15 (±0.48)	95.01 (±0.52)	81.17 (±0.97)	80.36 (±0.92)	628	30 (±4)
Mimetic-ConSeq	81.15 (±0.84)	80.32 (±0.85)	97.39 (±0.42)	97.61 (±0.35)	83.02 (±0.74)	82.39 (±0.73)	875	33 (±4)
Mimetic-All	**82.53** (±**0.90**)	**81.04** (±**1.02**)	**99.05** (±**0.26**)	**99.17** (±**0.22**)	**83.13** (±**0.85**)	**82.51** (±**0.81**)	1368	56 (±7)

Table 4: ECE comparison of DL-based traffic classifiers when combining different mixes of modalities for Joint-TC. The input types fed to each classifier are shown in Tab. 2.

Classifier	ECE [%]
bPAY	9.43 (±1.18)
bSEQ	3.33 (±0.96)
bContext	3.68 (±0.76)
Mimetic-Enhanced	13.59 (±2.94)
Mimetic-ConPay	4.29 (±0.68)
Mimetic-ConSeq	**2.40** (±**0.52**)
Mimetic-All	9.20 (±1.67)

6.1. Overall Performance Comparison

In Tab. 3, we report the performance of DL-based traffic classifiers combining different mixes of modalities, namely when considering both single-modality branches alone—each fed with one of the three different input types `PAY`, `SEQ`, and `Context`—and their combinations. The following results are obtained considering the best-performing $N_b = 576$ B and $N_p = 20$ packets. As shown, performance varies depending on the considered type of input and TC task.

In detail, the suitability of `Context` inputs discussed in Sec. 5.2, is demonstrated by the fact that considering only the bContext branch results in a significant performance improvement on the Act-TC task compared to Mimetic-Enhanced (+12% F-measure). With the ability of Mimetic-Enhanced to discriminate apps and the modularity the multimodal architecture being given, these results suggest that the addition of `Context` inputs can lead to a classifier that is able to simultaneously achieve good performance for all the considered tasks.

Therefore, the impact on performance when combining `Context` with `PAY` and `SEQ` is evaluated. Considering the classification performance of Mimetic-Enhanced as a reference, combining `Context` and `PAY` (Mimetic-ConPay) results in an increase of +15% in terms of F-measure for both Joint- and Act-TC tasks (+11% and +13% of accuracy, respectively). Unfortunately, the same combination results in a non-negligible decrease of −4% of F-measure (and −3% of accuracy) for the App-TC.

In contrast, combining `SEQ` and `Context` (Mimetic-ConSeq) we obtain a performance increase for both Joint- and Act-TC—i.e. +18% (resp. +14%) and +17% (resp. +15%) in terms of F-measure (resp. accuracy)—w.r.t. Mimetic-Enhanced. It is noteworthy that, by using the latter input combination, it is possible to ignore the payload content (i.e. `PAY`): this peculiarity makes this solution suitable also for future scenarios where more opaque encryption sublayers are deployed (e.g., extensions for *Encrypted Server Name Indication* and *Encrypted Client Hello* in TLS 1.3 [17]), likely hindering classification solutions leveraging payload (see [16] for an analysis of DL usage of payload-based inputs). Finally, combining the *three input types* (Mimetic-All), this model is able to outperform the other configurations for all the considered TC tasks.

Overall, in agreement with the results already discussed in Sec. 4.2, leveraging the bPAY and bSEQ branches individually leads to a performance degradation for all TC tasks, compared to Mimetic-Enhanced. In fact, when considering the `PAY` input only (i.e. the bPAY branch), we incur the worst performance, which corresponds to a loss between −18% (resp. −14%) and −26% (resp. −25%) in terms of F-measure (resp. accuracy). On the contrary, when we consider only the `SEQ` input (i.e. the bSEQ branch) we obtain a significant loss regarding the Joint-TC task (−11% F-measure) while for App- and Act-TC tasks the loss is more contained (−6% F-measure).

More in detail, Fig. 12a depicts the Act-TC performance of Mimetic-Enhanced (i.e. the state-of-art baseline), conditioned on each application. To further investigate the results obtained by combining `Context` with the other inputs, in Figs. 12b–12d we report the *gain/loss w.r.t.* Mimetic-Enhanced at the activity granularity (conditioned on each app).

As depicted in Fig. 12a, for most applications (6 out of 9), *VCall* shows to the best classification performance (≈ 80%). This does not hold for `Messenger`, `Slack`, and `Teams`, which result in the range 40%–60%. On the contrary, in the case of *Chat* ≈ 65% F-measure is achieved, regardless of the considered app (except for `Slack`). Finally, for *ACall* the performance varies strongly depending on the app between 38% (for `Zoom`) and 77% (for `Discord`). Interestingly, for `Slack`, the performance (always in the range of 60–65% F-measure) does not significantly vary with the activity.

The addition of `Context` results in a gain in terms of F-measure that depends on the specific combination of inputs used and varies with the app and the activity (Figs. 12b–12d).

Table 5: Accuracy and F-measure comparison of DL-based Mimetic-All against ML-based traffic classifiers when using all input types (i.e. `PAY`, `SEQ`, and `Context`). Models are trained using App×Act class labels. Results are in the format *avg.* (±*std.*) obtained over 10-folds. The best result per metric (column) is highlighted in boldface.

Classifier	DL	Joint-TC		App-TC		Act-TC	
		Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]	Accuracy [%]	F-measure [%]
DT	○	67.96 (±1.15)	66.05 (±1.36)	95.91 (±0.62)	96.35 (±0.56)	70.12 (±1.15)	68.89 (±1.34)
RF	○	80.57 (±0.74)	77.94 (±0.86)	98.92 (±0.14)	99.14 (±0.12)	81.33 (±0.79)	80.41 (±0.82)
Mimetic-All	●	**82.53** (±**0.90**)	**81.04** (±**1.02**)	**99.05** (±**0.26**)	**99.17** (±**0.22**)	**83.13** (±**0.85**)	**82.51** (±**0.81**)

Moreover, while Mimetic-ConSeq and Mimetic-All exhibit approximately the same average performance, Figs. 12c and 12d witness that the gains obtained strongly depend on the specific app. Indeed, Mimetic-All results in higher gains than Mimetic-ConSeq on the activities of `Skype`, `Teams`, and `Webex`, but lower performance improvements are observed for the remaining apps (i.e. `Discord`, `GotoMeeting`, `Meet`, `Messenger`, `Slack`, and `Zoom`). Also, for these apps we see that the main differences are related to *ACall* for which the former obtains a lower gain of about 3%–4% compared to the latter. Finally, the worst performance of Mimetic-ConPay compared to Mimetic-ConSeq and Mimetic-All is mainly due to a general lower gain for the activities of all the apps, especially regarding *VCall* for `Teams` and *ACall* for `Meet`, `Teams`, and `Webex`. In this regard, it is worth noting that for `Teams`, Mimetic-ConPay brings no improvement w.r.t. Mimetic-Enhanced for the *ACall* activity.

The novel set of Context Inputs is thus useful when dealing with the classification of both the app and the specific activity performed by the user. Also, results prove that the combination with the traffic inputs commonly used in the literature at biflow level (e.g., the first N_b byte of the transport level payload and/or features extracted from the first N_p packets) is fruitful.

6.2. Calibration Analysis

In this section, after demonstrating the effectiveness of Mimetic-Enhanced when tackling the Joint-TC task through the inclusion of *Context Inputs*, we evaluate its reliability [51] by means of a metric to quantify its calibration. To this end, we consider both the whole confidence vector—i.e. $\boldsymbol{p}(\boldsymbol{x}) = p_1(\boldsymbol{x}), \cdots, p_L(\boldsymbol{x})$—and the confidence associated to the inferred class—i.e. $\hat{p}(\boldsymbol{x}) = \max_{i=1,\cdots,L} p_i(\boldsymbol{x})$.

In general, a calibrated classifier is such that for each sample, the confidence of the predicted class $\hat{p}$ equals $P\{\hat{\ell} = \ell \mid \hat{p}\}$, where ℓ and $\hat{\ell}$ are the true and predicted labels, respectively. To quantify the degree of this (mis)alignment, we partition the predictions into M equally-spaced bins and compute both the (expected) accuracy and confidence for each of these.

Specifically, let B_m be the set of tested TC objects for which the confidence associated with the predicted label falls within $I_m \triangleq (\frac{m-1}{M}; \frac{m}{M})$, the accuracy and confidence associated with the bin are:

$$\mathrm{acc}(B_m) = |B_m|^{-1} \sum_{n \in B_m} 1(\hat{\ell}_{(n)} = \ell_{(n)}) \tag{1}$$

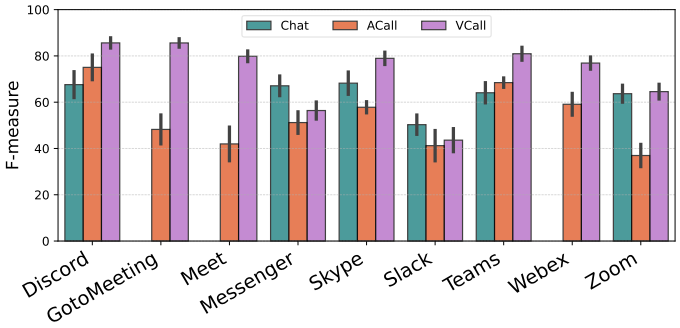
$$\mathrm{conf}(B_m) = |B_m|^{-1} \sum_{n \in B_m} \hat{p}(n) \tag{2}$$

where $\ell_{(n)}$, $\hat{\ell}_{(n)} \triangleq \arg\max_{1,\cdots,L} p_i(n)$, and $\hat{p}(n) \triangleq \max_{1,\cdots,L} p_i(n)$ are the true label, the predicted label, and the predicted confidence of the n^{th} sample, respectively. Since the confidence values vary in $[1/L, 1]$, we consider the starting point of the confidence interval to be $1/L$. Accordingly, to obtain a summary metric of the deviation from the perfect calibration, we consider the *Expected Calibration Error (ECE)*, defined as $E_{\hat{p}}\{|P\{\hat{\ell} = \ell \mid \hat{p}\} - \hat{p}|\}$, which represents the expected absolute deviation between the confidence and the confidence-conditional accuracy. The above metric can be approximated as:

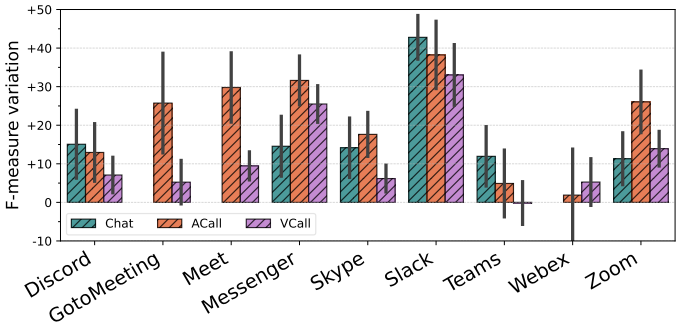
$$\mathrm{ECE} \approx \sum_{m=1}^{M} \frac{|B_m|}{N} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \tag{3}$$

where N is the overall number of tested samples. In Tab. 4, we report the ECE values (in percentage form) obtained considering each single-modality branch (i.e. bPAY, bSEQ, and bContext) and their multimodal combinations (i.e. Mimetic-Enhanced, Mimetic-ConPay, Mimetic-ConSeq, and Mimetic-All). First of all, the calibration of each classifier varies greatly depending on the considered branches (viz. input types). In fact, when considering the `SEQ` and `Context` inputs individually or their combination, we obtain a well-calibrated model—with an ECE $\in [2.4, 3.68]$%—as compared to the other setups. In contrast, considering (a) the sole `PAY` input or (b) combined with `SEQ`, results in a model with an ECE $\in [9.43, 13.59]$%, representing classifiers with lower calibration (viz. reliability). Finally, it is interesting to note the benefit on model calibration due to the use of *Context Inputs*. In fact, in all the cases these are included, we obtain a calibration gain corresponding to a reduced ECE of up to −5%.

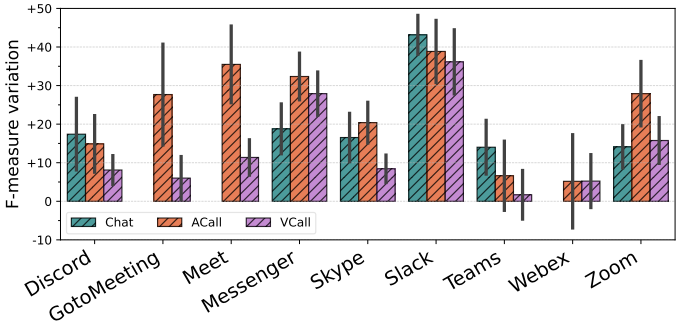
Then, to visualize in detail how $P\{\hat{\ell} = \ell \mid \hat{p}\}$ varies with $\hat{p}$, in Fig. 13 we show the accuracy as a function of the confidence by means of the so-called *reliability diagrams*. Therein, each diagram is compared with the ideal $P\{\hat{\ell} = \ell \mid \hat{p}\} = \hat{p}$ identity line: a perfectly-calibrated classifier entails a reliability diagram corresponding to the identity function. From inspection of the results, when considering the `PAY` and `SEQ` individually or in combination (see Figs. 13b–13d), the samples tend to be more evenly distributed within the bins. In contrast, when these inputs are combined with Context Inputs (see Figs. 13e–13g), we see that the samples tend to converge in the last bin (i.e. $[90\%, 100\%]$). This in turn means that the confidence of these classifiers *increases systematically*. Nonetheless, since the addition of Context Inputs also implies a *lower* ECE *for all the classifiers*, such increasingly-optimistic behavior is not paid at



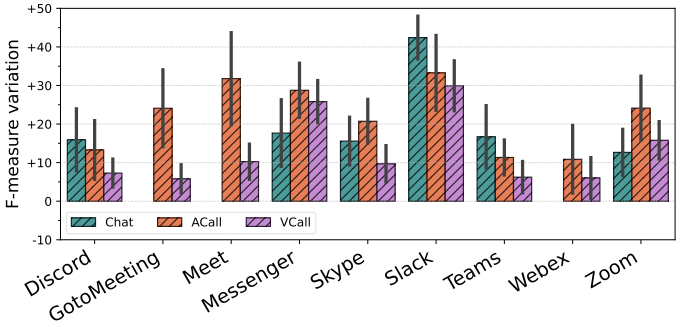
(a) MIMETIC-ENHANCED.



(b) MIMETIC-CONPAY.



(c) MIMETIC-CONSEQ.



(d) MIMETIC-ALL.

Figure 12: Per-activity performance for each app achieved by MIMETIC-ENHANCED (i.e., the basic configuration) (a). Gain/loss w.r.t. MIMETIC-ENHANCED obtained by considering the *Context Inputs* and related to MIMETIC-CONPAY (b), MIMETIC-CONSEQ (c), and MIMETIC-ALL (d).

the expenses of a decreased calibration. This clearly highlights a structural improvement of the classifiers' performance due to information originating from context biflows.

6.3. *Comparison of* MIMETIC-ALL *with Traffic Classifiers based on Machine Learning*

In this section, to show the effectiveness of our proposal, we compare MIMETIC-ALL with two ML algorithms commonly used in the state of the art (see Sec. 2), namely the Decision Tree (DT) and the Random Forest (RF). Specifically, ML algorithms have been trained and evaluated using `scikit-learn` Python Framework, adopting predefined hyper-parameters and using 100 estimators for the RF classifier. The aim of the analysis is also to assess separately the benefit of (*a*) the Context Inputs and (*b*) the proposed multimodal architecture (being able to process them wisely and thus achieve satisfactory TC performance). Indeed, unlike MIMETIC-ALL, for ML-classifiers the absence of multi-modality implies that the different inputs cannot be handled separately. Therefore, the three inputs have been treated as a single (monolithic) input obtained from their concatenation.

Table 5 reports the overall performance of models, trained using APP×ACT class labels and *using all input types* (i.e. `PAY`, `SEQ`, and `Context`). As shown, DT results in the worst performance w.r.t. all classification tasks. Specifically, the highest gap against MIMETIC-ALL concerns Joint- and Act-TC tasks, which corresponds to a worsening of approximately −15% (resp. −15%) and −14% (resp. −13%) in terms of F-measure (resp. Accuracy), respectively. Furthermore, the gap is reduced to roughly −3% in terms of both F-measure and Accuracy w.r.t. App-TC task. Finally, comparing RF with MIMETIC-ALL, while the performance related to App-TC is quite similar, there is a loss of up to 3% regarding Joint-TC and Act-TC in terms of both F-measure and Accuracy.

Hence, MIMETIC-ALL *is able to outperform both ML and DL algorithms when addressing the problem of joint traffic classification of the app and activity performed by the user.*

7. Conclusions and Future Directions

The COVID-19 pandemic has caused a sudden (while long-lasting) surge of the usage of Communication and Collaboration Apps. The latter have impacted the nature of Internet traffic, calling for novel improved tools for network monitoring and management. In this work, we focused on fine-grained TC of the most popular *Communication and Collaboration apps* and *corresponding activities* via DL approaches. Specifically, we considered three TC tasks (Joint-TC, App-TC, and Act-TC) and different training strategies based on the ground truth. MIMETIC-ENHANCED (a state-of-art multimodal TC approach) has been compared against recent DL single-modal solutions (1D-CNN and HYBRID), showing *the limitations of current architectures in recognizing activities*, due to inputs based only on the traffic object (biflow) under consideration. Indeed, while all the considered architectures achieve good performance (96–98% F-measure) when tackling App-TC, for Joint-TC and Act-TC these incur severe TC performance limitations (62% and

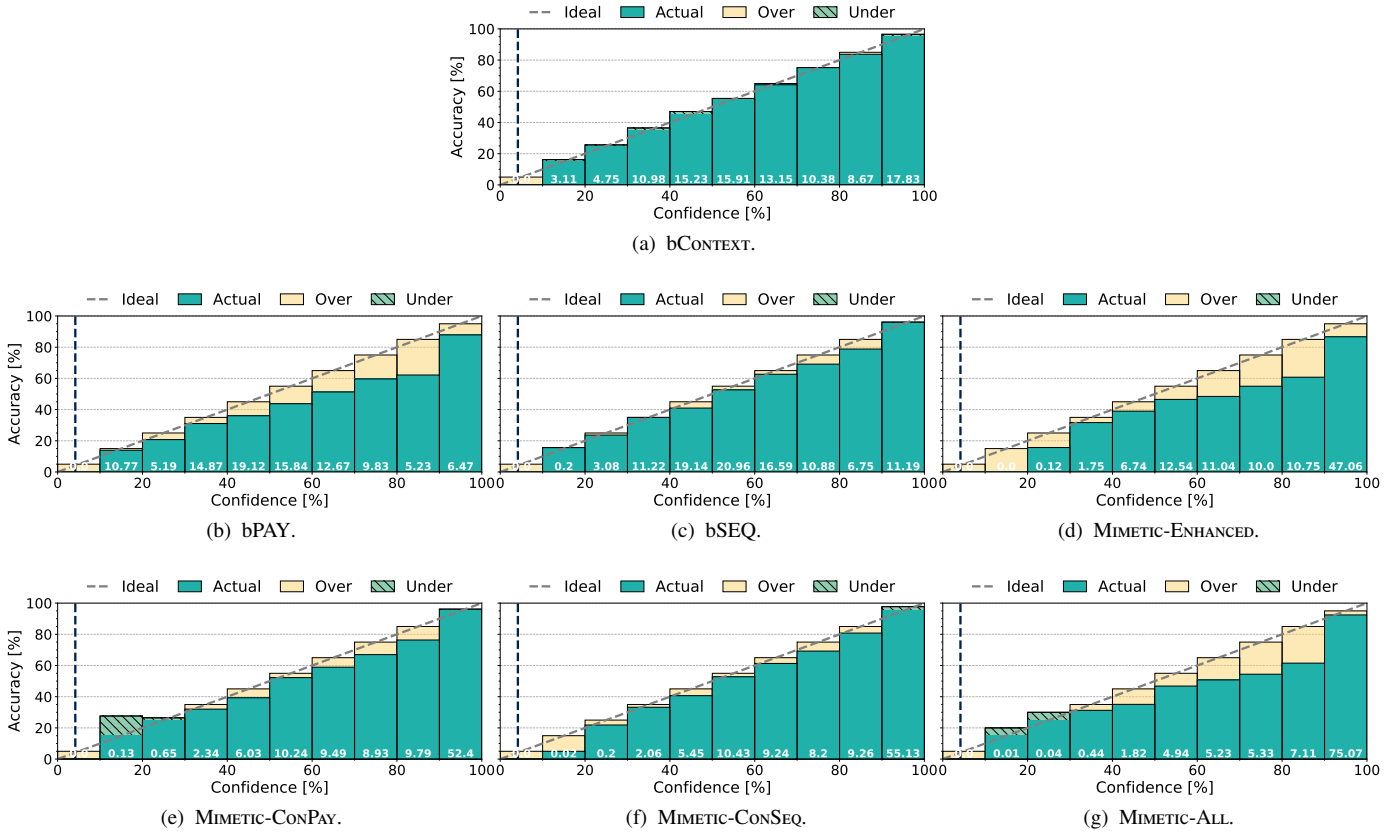


Figure 13: Impact of Context Inputs on calibration: reliability diagrams for APP×ACT-TC for classifiers characterized by different combinations of modalities (the input types fed to each classifier are shown in Tab. 2). Graphs in column allow to compare calibration without (upper row) and with (lower row) the Context Inputs. Confidence is divided in 10 bins, and it is $\geq \frac{1}{L}$ (vertical dashed line), with L being the number of classes. Over and under gap represent an over-confident (optimistic) and under-confident (pessimistic) miscalibration pattern, respectively. The number at the bottom of each bar reports the percentage of samples within the corresponding bin.

65% F-measure, respectively). To this end, we devised a novel set of *Context Inputs* that are conducive to effective classification of activities in an *early fashion*. We show that these inputs can exploit the side-information arising from concurrent biflows opened by the same app when performing a given activity, so we capitalize them to design a novel multimodal DL-based TC architecture, named MIMETIC-ALL. The latter classifier outperformed previous (multimodal) proposals and state-of-art ML classifiers (DT and RF) fed with the novel set of inputs proposed in this paper, reaching 81% and 82% F-measure for Joint-TC and Act-TC.

Still, when considering the complexity aspects, MIMETIC-ALL proves to be the best trade-off, exposing a complexity $\approx 4\times$ lower than 1D-CNN. The additional use of Context Inputs was also shown to be beneficial to improve the general calibration of the proposal (as well as the considered variants), thus providing a structural improvement of the classification model. Finally, *Context Inputs were also proven as excellent substitutes for payload-based inputs* in former state-of-the-art MIMETIC-ENHANCED, trading a small loss (−1% F-measure) in App-TC for the substantially improved performance in tasks involving activities (+18% and +17% for Joint- and Act-TC, respectively) and also the significant reduction of memory footprint and training time (i.e., −29% and −42%, respectively), besides the intrinsic greater robustness to future more opaque encryption sublayers. The experimental results were based on a newly collected dataset covering nine Android mobile apps (`Discord`, `GotoMeeting`, `Meet`, `Messenger`, `Skype`, `Slack`, `Teams`, `Webex`, and `Zoom`) and three user activities (*Chat*, *ACall*, and *VCall*), which is publicly released to the community.

Future directions of research will account for (*i*) validation of the proposed methodology—based on Context Inputs—on other benchmarking datasets; (*ii*) use of advanced learning strategies encompassing multi-task and hierarchical traffic classifiers (e.g. coupling a App-TC classifier with Joint-TC second stage, completely avoiding payload-based inputs); (*iii*) investigation of complementary traffic analysis tasks as fine-grain modeling and prediction; (*iv*) the use of XAI techniques to interpret and improve the behavior of MIMETIC-ALL, including deployment on resource-constrained devices.

Acknowledgements

This work was carried out by Dr. Idio Guarino under the grant "O. Carlini" funded by Consortium GARR, the Italian national network of University and Research. This work

Table A.6: List of acronyms and abbreviations (reported in alphabetical order).

Acronym	Definition
ACall	Audio-Call activity
ACT	Activity-related ground truth
Act-TC	Activity Traffic Classification task
APP	App-related ground truth
APPxACT	Joint App-and-Activity ground truth
App-TC	App Traffic Classification task
bCONTEXT	Branch of MIMETIC-ALL taking as input CONTEXT
bPAY	Branch of MIMETIC-ALL taking as input PAY
bSEQ	Branch of MIMETIC-ALL taking as input SEQ
BiGRU	Bidirectional Gated Recurrent Unit
Chat	Chat activity
CNN	Convolutional Neural Network
CONTEXT	Context Inputs computed at the arrival of the N_p-th packet
DIR	Packet Direction
DL	Deep Learning
DT	Decision Tree
ECE	Expected Calibration Error
GRU	Gated Recurrent Unit
IAT	Inter-Arrival Time
Joint-TC	Joint App-and-Activity Traffic Classification task
LR	Logistic Regressor
LSTM	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer Perceptron
MM	Multi-Modal architecture
PAY	First N_b bytes of transport-layer payload
PL	Transport-layer Payload Length
RF	Random Forest
RNN	Recurrent Neural Network
SAE	Stacked AutoEncoder
SEQ	Informative fields extracted from the sequence of the first N_p packets
SVM	Support Vector Classifier
TC	Traffic Classification
TCPWIN	TCP Window Size
VCall	Video-Call activity
XAI	eXplainable Artificial Intelligence
#CF	Number of Contextual Biflows
#TP	Number of Trainable Parameters

is partially funded by "Centro Nazionale HPC, Big Data e Quantum Computing – Italian Center for Super Computing (ICSC)" – Codice progetto MUR: CN_00000013 – CUP Unina: E63C22000980007, and by the Italian Research Program "PON Ricerca e Innovazione 2014-2020 (PON R&I) – Asse IV: Istruzione e ricerca per il recupero – REACT-EU – Azione IV.4: Dottorati e contratti di ricerca su tematiche dell'innovazione" – CUP Unina: E65F21002920003.

Appendix A. Acronyms and Abbreviations

The present Appendix provides the list of acronyms and abbreviations used in the manuscript and their definitions in Tab. A.6.

References

[1] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, G. Smaragdakis, The lockdown effect: Implications of the COVID-19 pandemic on Internet traffic, in: ACM Internet Measurement Conference (IMC), 2020, p. 1–18. doi:10.1145/3419394.3423658.

[2] A. Lutu, D. Perino, M. Bagnulo, E. Frias-Martinez, J. Khangosstar, A characterization of the COVID-19 pandemic impact on a mobile network operator traffic, in: ACM Internet Measurement Conference (IMC), 2020, p. 19–33. doi:10.1145/3419394.3423655.

[3] A. Affinito, A. Botta, G. Ventre, The impact of COVID on network utilization: an analysis on domain popularity, in: IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2020, pp. 1–6. doi:10.1109/CAMAD50429.2020.9209302.

[4] T. Favale, F. Soro, M. Trevisan, I. Drago, M. Mellia, Campus traffic and e-Learning during COVID-19 pandemic, Computer Networks 176 (2020) 107290. doi:10.1016/j.comnet.2020.107290.

[5] A. Ukani, A. Mirian, A. C. Snoeren, Locked-in during lock-down, in: 21st ACM Internet Measurement Conference (IMC), 2021, pp. 480–486. doi:10.1145/3487552.3487828.

[6] M. Karamollahi, C. Williamson, M. Arlitt, Zoomiversity: A case study of pandemic effects on post-secondary teaching and learning, in: O. Hohlfeld, G. Moura, C. Pelsser (Eds.), Passive and Active Measurement, Springer International Publishing, Cham, 2022, pp. 573–599. doi:10.1007/978-3-030-98785-5_26.

[7] A. Choi, M. Karamollahi, C. Williamson, M. Arlitt, Zoom session quality: A network-level view, in: O. Hohlfeld, G. Moura, C. Pelsser (Eds.), Passive and Active Measurement, Springer International Publishing, Cham, 2022, pp. 555–572. doi:10.1007/978-3-030-98785-5_25.

[8] M. Candela, V. Luconi, A. Vecchio, Impact of the COVID-19 pandemic on the Internet latency: A large-scale study, Computer Networks 182 (2020) 107495. doi:10.1016/j.comnet.2020.107495.

[9] T. Böttger, G. Ibrahim, B. Vallis, How the Internet reacted to COVID-19: A perspective from Facebook's edge network, in: ACM Internet Measurement Conference (IMC), 2020, p. 34–41. doi:10.1145/3419394.3423621.

[10] E. Stutz, Y. Pradkin, X. Song, J. Heidemann, Visualizing Internet measurements of Covid-19 work-from-home, in: IEEE International Conference on Big Data (Big Data), 2021, pp. 5633–5638. doi:10.1109/BigData52589.2021.9671311.

[11] A. Dainotti, A. Pescapè, K. C. Claffy, Issues and future directions in traffic classification, IEEE Network 26 (2012) 35–40.

[12] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Mobile encrypted traffic classification using Deep Learning: Experimental evaluation, lessons learned, and challenges, IEEE Trans. Netw. Service Manag. 16 (2019) 445–458. doi:10.1109/TNSM.2019.2899085.

[13] M. Henze, J. Pennekamp, D. Hellmanns, E. Mühmer, J. H. Ziegeldorf, A. Drichel, K. Wehrle, Cloudanalyzer: Uncovering the cloud usage of mobile apps, in: Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2017, pp. 262–271. doi:10.1145/3144457.3144471.

[14] W. Wang, M. Zhu, J. Wang, X. Zeng, 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, pp. 43–48. doi:10.1109/ISI.2017.8004872.

[15] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Network traffic classifier with convolutional and recurrent neural networks for Internet of Things, IEEE Access 5 (2017) 18042–18050. doi:10.1109/ACCESS.2017.2747560.

[16] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, A. Pescapé, XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures, IEEE Transactions on Network and Service Management 18 (2021) 4225–4246. doi:10.1109/TNSM.2021.3098157.

[17] E. Rescorla, K. Oku, N. Sullivan, C. A. Wood, TLS Encrypted Client Hello, Internet-Draft draft-ietf-tls-esni-14, Internet Engineering Task Force, 2022. URL: https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-14, work in Progress.

[18] I. Guarino, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, A. Pescapé, Classification of communication and collaboration apps via advanced deep-learning approaches, in: IEEE 26th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2021, pp. 1–6. doi:10.1109/CAMAD52502.2021.9617789.

[19] Z. Wang, The Applications of Deep Learning on Traffic Identification.,

2015.

[20] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, M. Saberian, Deep packet: A novel approach for encrypted traffic classification using deep learning, Soft Computing 24 (2020) 1999–2012. doi:10.1007/s00500-019-04030-2.

[21] Y. Zeng, H. Gu, W. Wei, Y. Guo, *Deep-Full-Range*: A deep learning based network encrypted traffic classification and intrusion detection framework, IEEE Access 7 (2019) 45182–45190. doi:10.1109/ACCESS.2019.2908225.

[22] S. Rezaei, B. Kroencke, X. Liu, Large-scale mobile app identification using deep learning, IEEE Access 8 (2020) 348–362. doi:10.1109/ACCESS.2019.2962018.

[23] C. Liu, L. He, G. Xiong, Z. Cao, Z. Li, FS-Net: A flow sequence network for encrypted traffic classification, in: IEEE Conference on Computer Communications (INFOCOM), 2019, pp. 1171–1179. doi:10.1109/INFOCOM.2019.8737507.

[24] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapè, MIMETIC: mobile encrypted traffic classification using multimodal deep learning, Elsevier Computer Networks 165 (2019) 106944. doi:10.1016/j.comnet.2019.106944.

[25] X. Wang, S. Chen, J. Su, Automatic mobile app identification from encrypted traffic with hybrid neural networks, IEEE Access 8 (2020) 182065–182077. doi:10.1109/ACCESS.2020.3029190.

[26] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, DISTILLER: Encrypted traffic classification via multimodal multitask deep learning, Journal of Network and Computer Applications (2021) 102985. doi:10.1016/j.jnca.2021.102985.

[27] H. Huang, H. Deng, J. Chen, L. Han, W. Wang, Automatic multi-task learning system for abnormal network traffic detection, Int. Journal of Emerging Technologies in Learning 13 (2018) 4–20. doi:10.3991/ijet.v13i04.8466.

[28] Y. Zhao, J. Chen, D. Wu, J. Teng, S. Yu, Multi-task network anomaly detection using federated learning, in: ACM 10th International Symposium on Information and Communication Technology (SoICT), 2019, pp. 273–279. doi:10.1145/3368926.3369705.

[29] I. Akbari, M. A. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, S. Tuffin, A look behind the curtain: Traffic classification in an increasingly encrypted web, Proc. ACM Meas. Anal. Comput. Syst. 5 (2021). doi:10.1145/3447382.

[30] I. Guarino, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, A. Pescapè, Characterizing and modeling traffic of communication and collaboration apps bloomed with COVID-19 outbreak, in: IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI), 2021, pp. 400–405. doi:10.1109/RTSI50628.2021.9597263.

[31] M. Conti, L. V. Mancini, R. Spolaor, N. V. Verde, Analyzing android encrypted network traffic to identify user actions, IEEE Trans. Inf. Forensics Security 11 (2016) 114–125. doi:10.1109/TIFS.2015.2478741.

[32] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, J. Qian, Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic, in: USENIX Workshop on Offensive Technologies (WOOT), 2016, pp. 69–78.

[33] E. Grolman, A. Finkelshtein, R. Puzis, A. Shabtai, G. Celniker, Z. Katzir, L. Rosenfeld, Transfer learning for user action identication in mobile apps via encrypted traffic analysis, IEEE Intelligent Systems 33 (2018) 40–53. doi:10.1109/MIS.2018.111145120.

[34] F. Aiolli, M. Conti, A. Gangwal, M. Polato, Mind your wallet's privacy: identifying bitcoin wallet apps and user's actions through network traffic analysis, in: 34th ACM/SIGAPP Symposium on Applied Computing (SAC), 2019, pp. 1484–1491. doi:10.1145/3297280.3297430.

[35] V. F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Robust smartphone app identification via encrypted network traffic analysis, IEEE Transactions on Information Forensics and Security 13 (2018) 63–78. doi:10.1109/TIFS.2017.2737970.

[36] D. Li, W. Li, X. Wang, C.-T. Nguyen, S. Lu, App trajectory recognition over encrypted internet traffic based on deep neural network, Computer Networks 179 (2020) 107372. doi:10.1016/j.comnet.2020.107372.

[37] C. Li, C. Dong, K. Niu, Z. Zhang, Mobile service traffic classification based on joint deep learning with attention mechanism, IEEE Access 9 (2021) 74729–74738. doi:10.1109/ACCESS.2021.3081504.

[38] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, A. Pescapé, MIRAGE: Mobile-app traffic capture and ground-truth creation, in: 4th International Conference on Computing, Communications and Security (ICCCS), 2019, pp. 1–8. doi:10.1109/CCCS.2019.8888137.

[39] K. MacMillan, T. Mangla, J. Saxon, N. Feamster, Measuring the performance and network utilization of popular video conferencing applications, in: 21st ACM Internet Measurement Conference (IMC), New York, NY, USA, 2021, p. 229–244. doi:10.1145/3487552.3487842.

[40] A. Nistico, D. Markudova, M. Trevisan, M. Meo, G. Carofiglio, A comparative study of RTC applications, in: 2020 IEEE International Symposium on Multimedia (ISM), IEEE, Naples, Italy, 2020, pp. 1–8. URL: https://ieeexplore.ieee.org/document/9327919/. doi:10.1109/ISM.2020.00007, zSCC: 0000009.

[41] Sandvine, The Global Internet Phenomena Report COVID-19 Spotlight., 2020.

[42] Lexi Sydow - App Annie, Video Conferencing Apps Surge from Coronavirus Impact., 2020.

[43] Sandvine, The Global Internet Phenomena Report., 2022.

[44] M. Petit-Huguenin, G. Salgueiro, J. Rosenberg, D. Wing, R. Mahy, P. Matthews, Session Traversal Utilities for NAT (STUN), RFC 8489, 2020. URL: https://www.rfc-editor.org/info/rfc8489. doi:10.17487/RFC8489.

[45] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Multi-classification approaches for classifying mobile app traffic, Journal of Network and Computer Applications 103 (2018) 131–145. doi:10.1016/j.jnca.2017.11.007.

[46] T. Stöber, M. Frank, J. Schmitt, I. Martinovic, Who do you sync you are? smartphone fingerprinting via application behaviour, in: ACM 6th conference on Security and privacy in wireless and mobile networks (WISEC), 2013, pp. 7–12. doi:10.1145/2462096.2462099.

[47] V. F. Taylor, R. Spolaor, M. Conti, I. Martinovic, Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic, in: IEEE European Symposium on Security and Privacy (EuroS&P), 2016, pp. 439–454. doi:10.1109/EuroSP.2016.40.

[48] E. Vanrykel, G. Acar, M. Herrmann, C. Diaz, Leaky birds: Exploiting mobile application traffic for surveillance, in: International Conference on Financial Cryptography and Data Security, Springer, 2016, pp. 367–384. doi:10.1007/978-3-662-54970-4_22.

[49] B. Claise, B. Trammell, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information, RFC 7011, 2013. URL: https://rfc-editor.org/rfc/rfc7011.txt. doi:10.17487/rfc7011.

[50] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al., Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, volume 30, Citeseer, 2013, p. 3.

[51] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberger, On calibration of modern neural networks, in: 34th International Conference on Machine Learning (ICML), 2017, p. 1321–1330.