IEEE Open Journal of the Communications Society

Received 04 October, 2023; revised XX Month, XXXX.

Explainable Deep-Learning Approaches for Packet-level Traffic Prediction of Collaboration and Communication Mobile Apps

Idio Guarino[1], ***Graduate Student Member, IEEE***, **Giuseppe Aceto**[1], **Domenico Ciuonzo**[1], ***Senior Member, IEEE***, **Antonio Montieri**[1], **Valerio Persico**[1], **Antonio Pescapè**[1], ***Senior Member, IEEE***

[1]Department of Electrical Engineering and Information Technologies (DIETI), University of Napoli "Federico II", Italy

CORRESPONDING AUTHOR: Idio Guarino (e-mail: idio.guarino@unina.it).

ABSTRACT

Significant transformations in lifestyle have reshaped the Internet landscape, resulting in notable shifts in both the magnitude of Internet traffic and the diversity of apps utilized. The increased adoption of communication-and-collaboration apps, also fueled by lockdowns in the COVID pandemic years, has heavily impacted the management of network infrastructures and their traffic. A notable characteristic of these apps is their multi-activity nature, e.g., they can be used for chat and (interactive) audio/video in the same usage session: predicting and managing the traffic they generate is an important but especially challenging task.

In this study, we focus on real data from four popular apps belonging to the aforementioned category: `Skype`, `Teams`, `Webex`, and `Zoom`. First, we collect traffic data from these apps, reliably label it with both the app and the specific user activity and analyze it from the perspective of traffic prediction. Second, we design data-driven models to predict this traffic at the finest granularity (i.e. at packet level) employing four advanced multitask deep learning architectures and investigating three different training strategies. The trade-off between performance and complexity is explored as well. We publish the dataset and release our code as open source to foster the replicability of our analysis. Third, we leverage the packet-level prediction approach to perform aggregate prediction at different timescales. Fourth, our study pioneers the *trustworthiness* analysis of these predictors via the application of eXplainable Artificial Intelligence to (a) interpret their forecasting results and (b) evaluate their reliability, highlighting the relative importance of different parts of observed traffic and thus offering insights for future analyses and applications. The insights gained from the analysis provided with this work have implications for various network management tasks, including monitoring, planning, resource allocation, and enforcing security policies.

INDEX TERMS communication apps; collaboration apps; COVID; deep learning; encrypted traffic; multitask approaches; traffic prediction; XAI.

I. Introduction

Network traffic and the underlying infrastructure have a symbiotic relationship, constantly evolving in tandem. Traffic adapts to leverage the enhanced capabilities offered by network technologies, such as higher bandwidth, lower latency, and increased resilience and flexibility. In turn, the infrastructure evolves to meet the demands generated by emerging applications. Recent reports from global operators indicate a significant increase in fixed and mobile broadband usage over the past two years, with growth rates ranging from 20% to 50% and 15% to 35%, respectively [1]. Moreover, such needs have dramatically shifted lately, with COVID accelerating the pace of digitization by several years and driving changes in how people across the globe utilize technology. On the one hand, the significant increase in Internet usage during the COVID pandemic ($+15$–20% in terms of volume [2] with a measurable impact on network performance—e.g., increasing variability of delay and loss rate [3]) can be attributed to the widespread adoption of remote work, remote education, online commerce, and en-

tertainment activities during lockdowns and periods of social distancing. On the other hand, with COVID becoming more endemic in many parts of the world, the continuous increase in Internet traffic is attributed to enduring habits and the widespread adoption of certain apps that people have become accustomed to. This phenomenon is not solely driven by lingering COVID-related behaviors but rather signifies the establishment of permanent habits and the global familiarity with these ubiquitous apps. Focusing on the nature of such apps, trends show that they no longer deal with just one type of traffic: it is becoming increasingly common to have video, voice, chat, and gaming content all in the same app [1]. These characteristics lead to increased complexity in network traffic management for network operators, as they encounter notable discrepancies in both inter- and intra-app behaviors [4, 5].

In this complex scenario, the quality of experience offered plays a critical role in determining customer satisfaction. Hence, network performance necessitates the implementation of automated and adaptive management of network resources [6]. For a system to be truly responsive, it needs to react promptly to observed network traffic. However, an even more effective system would be *proactive*, capable of foreseeing future traffic patterns and providing ample time to plan and execute appropriate actions.

This increasing need in recent years has driven the creation of diverse solutions for predicting network traffic. These solutions typically concentrate on *aggregated metrics*, such as total volume and average rate, spanning over extended time intervals.

Differently, in this work, we focus on **fine-grained prediction**, i.e. with outcomes at **packet level**, by designing a **novel Deep Learning (DL)** solution. Further, we inspect the **effectiveness** gains deriving by having prediction models tailored on specific apps and protocols. The scientific literature has shown the suitability of DL-powered solutions to address prediction of aggregate traffic [7, 8]. Nonetheless, the capitalization of DL for fine-grained prediction is still a challenge and performance gaps need to be overcome [9].

In line with the concerns related to the adoption of Artificial Intelligence (AI) for driving critical systems, we also deepen aspects related to the **trustworthiness** of the designed solutions, focusing on *technical robustness* and *transparency*.[1]

More specifically, the *technical contributions* provided by our work are as follows:

- We tackle the challenging task of predicting network traffic at the finest granularity level, namely the *packet level*, by leveraging Deep Neural Networks (DNNs). Our proposed models utilize a limited memory of previously observed packets to accurately predict various characteristics of the next packet, such as its direction, inter-arrival time, and transport-layer payload length. This fine-grained prediction capability empowers efficient traffic management (i) at the biflow level (thus allowing different types of aggregation, such as server-side or client-side), (ii) on flexible (including short) timescales, and (iii) at different viewpoints (e.g., packet or volume info), making valuable contributions to the advancement of next-generation networks (e.g., supporting network slicing functionalities).
- To assess the effectiveness of our proposal, we evaluate it using real traffic data from *communication-and-collaboration mobile applications (CC apps)*. We selected the ones that experienced a sudden surge in popularity (`Skype`, `Teams`, `Webex`, and `Zoom`), collected and reliably labeled their traffic at per-activity granularity, and used this ground truth to train and evaluate DL models. The dataset is publicly available, to foster reproducibility, and we published also our code as open-source, to allow replicability of the results.
- We investigate the advantage of *multitask DL models* designed for specific apps or protocols (TCP/UDP) with respect to a *single-model* trained on all the considered CC apps' traffic. In a complementary fashion, we assess how traffic prediction performance varies from multiple perspectives: (i) different *apps*, (ii) different *activities*, or (iii) different *protocols*.
- To overcome the limitation of black-box AI models, particularly DNNs, we employ *eXplainable Artificial Intelligence (XAI)* techniques. XAI allows us to gain insights into the model performance (by assessing how reliable is each prediction) and establish connections between performance and traffic characteristics (imputing importance to different packet parameters). By utilizing XAI, we strengthen our confidence in the results and identify areas for improvement. To the best of our knowledge, this study is the first to propose a DL approach evaluated using XAI to predict network traffic at the packet level.
- To demonstrate the timescale flexibility of our proposal, we capitalize on the outcomes of the fine-grained packet-level multitask predictor and define an approach that can deal with *coarser-grained traffic prediction tasks* (e.g., prediction of traffic volume and number of packets) with *arbitrary aggregation intervals*.

The present work significantly extends and improves our earlier conference paper [10] with new investigations and results, namely: (i) a *traffic characterization* of CC apps' traffic; (ii) different novel strategies for training the DL models (e.g., by separating the traffic based on the transport-layer protocol); (iii) an XAI-based evaluation that integrates interpretability, reliability, multifaceted error trend analysis, and in-depth characterization of prediction errors; (iv) the prediction of *arbitrary-granularity* traffic aggregates (e.g., number of packets and traffic volume) by capitalizing our packet-level prediction proposal.

[1] https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai

The rest of the paper is organized as follows. Section II surveys the literature predicting network traffic or analyzing the trustworthiness of DL models that focus on network traffic analysis. Section III provides the details of our methodology regarding the packet-level traffic prediction via multitask DL, the XAI techniques adopted to investigate the interpretability and reliability of DL models, and the prediction of traffic aggregates. Section IV describes the apps/activities selection rationale, the dataset collection, and the evaluation metrics. Section V presents the experimental evaluation with related take-home messages. Finally, Section VII provides conclusions and future perspectives.

II. Related Work

In this section, we provide an overview of the works that have addressed network traffic prediction (Sec. II.A) or exploited XAI techniques in the context of network traffic analysis (Sec. II.B). Finally, we provide the positioning of the present paper against the related literature for each reviewed topic (Sec. II.C).

A. Network Traffic Prediction

The scientific community has shown great interest in predicting the evolution of network traffic. Researchers have approached different prediction tasks related to a variety of distinct practical network problems. Table 1 summarizes the main aspects of each work surveyed herein. We categorize each paper based on whether (a) it tackles the prediction of traffic generated by multiple activities associated with the considered application and (b) it uses a multitask model, detailing also (c) the specific prediction techniques employed. Then, we specify (d) if such techniques are designed for fine-grained or coarse-grained traffic prediction, (e) how they are evaluated (i.e. the granularity of the prediction task), and (f) the traffic parameters/quantity predicted. Finally, the last column flags (g) the works publicly releasing the dataset used in their experimentations. The present section ends with the positioning of our work whose main points are recapped in the last row of Tab. 1.

Firstly, we can notice that none of the previous works performs a **per-activity** breakdown of the predictions associated with the considered application.

Regarding the particular **techniques** exploited for traffic prediction, Tab. 1 highlights a rising utilization of DL models, also in a **multitask** configuration [9, 11, 22]. Particularly, related works mostly employ CNN of different dimensions [9, 15], LSTM [9, 15, 17, 18, 19, 21, 23], GRU [9, 17, 23], SAE [13], GNN [24], and hybrid architectures obtained via their combinations [9, 15, 20]. Fewer works leverage Markov models (e.g., MC, HMM, and MMG) [11, 14, 16, 22], traditional ML models (e.g., LR, SVR, k-NNR, or RFR) [9, 16, 18, 22], and statistical techniques (e.g., ARIMA or FARIMA) [12, 15, 17, 18], usually as performance baselines to evaluate DL models, with the latter commonly showing better prediction performance.

Concerning the **design** of traffic predictors, several works propose solutions tailored for forecasting the evolution of traffic aggregates (**CG**). On the other hand, fewer proposals are designed to forecast traffic characteristics at a finer level (**FG**).

Regardless of the specific design choice, we also highlight the granularity of the prediction task faced to evaluate the proposed solutions.

The works performing coarse-grained evaluation (**CG-eval**) forecast various traffic aggregates, such as bit rates [23], packet distributions [17], and traffic volumes [12, 13, 17, 18, 19, 20, 24] at different time resolutions, ranging from less than one second to few seconds, minutes, hours, and even days. Among the latter, some works take into account also the geographical or topological distribution of sources and destinations, rather than leveraging the (sole) temporal information, via traffic matrices [18] or considering the geographic distribution of data volumes (e.g., aggregated data calls) as observed at base stations [15, 20, 24].

Differently, the proposals performing fine-grained evaluation (**FG-eval**) can capitalize on different sources of information. Some works tackle packet-level traffic prediction relying entirely on network-layer features available also in case of encryption (●), such as packet sizes, directions, and inter-arrival times [9, 11, 22]. Other prediction approaches consider video traffic and forecast video frame properties exploiting application-layer information which is not always available when encryption strategies are enforced [12, 14, 16]. We underline this shortcoming by partially flagging (◐) the FG-eval column in Tab. 1.

B. AI Trustworthiness in Traffic Analysis

Recently, researchers have applied XAI techniques to improve the performance, robustness, reliability, and feasibility of AI models that tackle networking-related tasks [36]. Table 2 reports the works tackling various networking problems by means of different XAI methods, focusing on the aim of the trustworthiness analysis conducted. Specifically, we flag the works that aim to (partially) interpret their forecasting results and/or measure to which extent the confidence associated with the latter can be deemed reliable (i.e. high/low confidence leads to high/low accuracy in prediction). The last row of Tab. 2 summarizes the present work, whose positioning w.r.t. related work is discussed at the end of this section.

Referring to the **networking problems** addressed in the light of AI trustworthiness, several papers face anomaly detection [25] or traffic classification [28, 29, 30, 31, 32, 33, 34, 35]. Networking-related prediction tasks with different facets are also tackled: video bit-rate adaptation based on reinforcement learning [26], video quality prediction via clustering [27], and packet-level traffic prediction (i.e. the same problem tackled in the present work) via Markovian and DL approaches [9, 22].

Table 1: Related work tackling various network traffic prediction tasks using different approaches. Reported papers are listed in chronological order. The last row summarizes the current proposal. *The solution is not designed for a specific prediction task and is evaluated on video-traffic datasets with different aggregations (from single frame to seconds).
MT: Multitask. **FG**: Fine-Grained. **CG**: Coarse-Grained. **Data**: Publicly Available Dataset. ● present, ◐ partially present, ○ lacking.

Reference	Per-activity	MT	Techniques	Design	CG-eval	FG-eval	Prediction	Data
Dainotti et al. [11]	○	●	HMM	FG	○	●	PS, IAT	●
Katris and Daskalaki [12]	○	○	FARIMA, RBF, MLP	FG *	●	◐	Frame Size, Traffic Volume	●
Oliveira et al. [13]	○	○	MLP, JNN, SAE	CG	●	○	Traffic Volume	●
Tanwir et al. [14]	○	○	HMM, MMG	FG	○	◐	Frame Size	●
Huang et al. [15]	○	○	3D-CNN, 3D-CNN+LSTM, ARIMA, LSTM, MLP	CG	●	○	Data CDR Count	●
Kalampogia and Koutsakis [16]	○	○	LR, MC	FG	○	◐	B-Frame Size	●
Ramakrishnan and Soni [17]	○	○	ARIMA, CV, GRU, LSTM, MA, RNN	CG	●	○	Packet Distribution Traffic Volume	◐
Hua et al. [18]	○	○	ARIMA, LSTM, MLP RCLSTM, SVR	CG	●	○	Traffic Volume	●
Huo et al. [19]	○	○	STL+Seq2Seq-LSTM	CG	●	○	Traffic Volume	●
Zhang et al. [20]	○	○	Seq2Seq+ConvLSTM	CG	●	○	Traffic Volume	●
He et al. [21]	○	○	AVP, DQN, LSTM PMF, ZP	CG	●	○	Dw Bytes	○
Aceto et al. [22]	○	●	HMM, MC, LR, k-NNR, RFR	FG	○	●	PL, IAT, DIR	●
Montieri et al. [9]	○	●	CNN, LSTM, GRU, SeriesNet, DSANet, MC, LR, k-NNR, RFR	FG	○	●	PL, IAT, DIR	●
Saha et al. [23]	○	○	RNN, LSTM, GRU	CG	●	○	Bit-rate	○
Fang et al. [24]	○	○	GNN	CG	●	○	Traffic Volume	○
This Work	●	●	CNN, LSTM, GRU, SeriesNet	FG	●	●	PL, IAT, DIR, Number of Packets, Traffic Volume	●

We can notice that most of the works applies **interpretability** techniques to provide explanations of the decisions taken. More specifically, commonly used **XAI methods** provide various forms of *post-hoc explanations* [33]: *(a) layer-wise relevance propagation* [25] which supplies explanations in an iterative fashion exploiting the layered structure of the neural network; *(b) interpretable local surrogates* via *LIME* [26, 27] which replaces the decision function with a self-explanatory local surrogate model; *(c)* different types of *perturbation analyses*, such as occlusion analysis [28, 32] or universal perturbation attacks [34]; *(d)* importance attribution based on *Shapley values*, either local [37] or global [33]. Other XAI methods based on *visual representations* (e.g., t-SNE and Feature Maps) [29] inspect the activation of intermediate neurons to highlight the most important features that led to the decision. Moreover, *Markovian Distillation* is applied to interpret traffic-prediction results by comparing Markov Chains' transition probabilities and DL predictions [9, 22]. Going further, solutions aiming at *explainability-by-design* compare parts of input data with per-class prototypes [35].

Finally, the **reliability** of DL models is investigated via a *calibration analysis* [31, 33] of their probabilistic outputs that aims to determine whether the confidence associated with the final decision reflects its reliability and possibly improve it.

C. Positioning w.r.t. Related Literature

This work aims to predict the traffic generated by some of the most popular and used CC apps at the packet level (i.e., our proposal is designed to fulfill fine-grained traffic prediction), specifically focusing on the lack of trustworthiness possibly characterizing DL solutions for network traffic analysis.

Compared to other works addressing **network traffic prediction** (ref. Tab. 1), we exploit fine-grained predictions to forecast also traffic aggregates in terms of the number of packets and volume. Thus, *we perform both fine-grained and coarse-grained traffic prediction* exploiting only network-layer features differently than all the reported works. Indeed, all of them exclusively focus on one of these tasks, namely either CG-eval or FG-eval. The sole exception is the work of [12], which is, however, specifically tailored for the prediction of video-frame sizes and leverages application-layer information unavailable in case of encryption.

Considering the related works exploiting multitask models, in [11] the focus is on the preliminary aspects of

Table 2: Related work employing XAI when facing various networking problems. Reported papers are listed in chronological order. The last row summarizes the current proposal.

Reference	Networking Problem	Interpretability	Reliability	XAI Method
Amarasinghe et al. [25]	Anomaly Detection	●	○	LRP
Dethise et al. [26]	Video bit-rate Prediction	●	○	LIME
Morichetta et al. [27]	Video Quality Prediction	●	○	LIME
Rezaei et al. [28]	Traffic Classification	●	○	Occlusion Analysis
Beliard et al. [29]	Traffic Classification	●	○	t-SNE Feature Map Visualization
Wang et al. [30]	Traffic Classification	●	○	DEEP SHAP
Aceto et al. [31]	Traffic Classification	○	●	Calibration Analysis
Aceto et al. [22]	Packet-level Traffic Prediction	◐	○	Markovian-Distillation
Akbari et al. [32]	Traffic Classification	●	○	Occlusion Analysis
Montieri et al. [9]	Packet-level Traffic Prediction	◐	○	Markovian-Distillation
Nascita et al. [33]	Traffic Classification	●	●	DEEP SHAP, Calibration Analysis
Sadeghzadeh et al. [34]	Traffic Classification	●	○	Perturbation Analysis
Fauvel et al. [35]	Traffic Classification	●	○	Explainable-by-Design DL Architecture
This Work	Packet-level Traffic Prediction	●	●	DEEP SHAP, Calibration Analysis

● present, ◐ partially present, ○ lacking.

traffic modeling of unidirectional flows—thus neglecting the advantage of considering request-response interaction—of non-mobile-app traffic by means of HMM. On the other hand, HMM and MC models are investigated to characterize and predict network traffic generated by mobile apps in [22] without exploiting the advantages of multitask DL models.

The closest work to ours is [9], since it addresses the same prediction problem (i.e., network-traffic prediction at packet level via DL approaches) investigated herein. However, compared to the aforementioned reference, this paper provides the following major contributions: *(a)* We focus on the traffic generated by *CC apps*. First, this choice reflects a practical interest of network operators, as these apps have become extremely popular with the COVID pandemic. In addition, *CC apps* are peculiar due to their multi-activity nature. In fact, it is increasingly common to have video, voice, chat, and game content within the same app rather than dealing with only one type of traffic. This multi-activity nature makes network management more complex as different types of traffic require different network management techniques due to their unique characteristics and requirements [4]. Therefore, we investigate the suitability of DL solutions to predict their fine-grained characteristics. *(b)* According to such goals, we rely on a more detailed ground truth, which includes the specific activity performed by the users (beyond the app generating the traffic), to investigate the impact of *multiple activities* on the capability of the DL model to predict traffic characteristics. *(c)* We assess the suitability of considering a single model for all apps or a specific model for each transport-level protocol to predict CC-app traffic against per-app models.

Focusing on **AI trustworthiness in traffic analysis**, *we address the inherent lack of trustworthiness of DL models in packet-level traffic prediction* (ref. Tab. 2). Firstly, we offer *interpretability* through XAI techniques. Our approach involves conducting a comprehensive analysis of DL models using DEEP SHAP to provide interpretable results. To the best of our knowledge, no other research has utilized these XAI techniques to investigate the interpretability of DL models for network traffic prediction, irrespective of the granularity of prediction or the specific DL model employed. Indeed, our methodology overcomes the limitations of preliminary interpretability solutions exploited in previous works facing packet-level traffic prediction [9, 22] based on Markovian distillation, which is constrained by memory limitations. Moreover, we investigate the *reliability* of DL models via a calibration analysis. To the best of our knowledge, no other work dealing with network traffic prediction has investigated such an aspect.

III. Methodology

In this section, we present an exhaustive account of the methodology devised for predicting network traffic at the packet level using multitask DL and explaining/assessing the forecasting outcomes. The workflow we employ to address this prediction problem is depicted in Fig. 1.

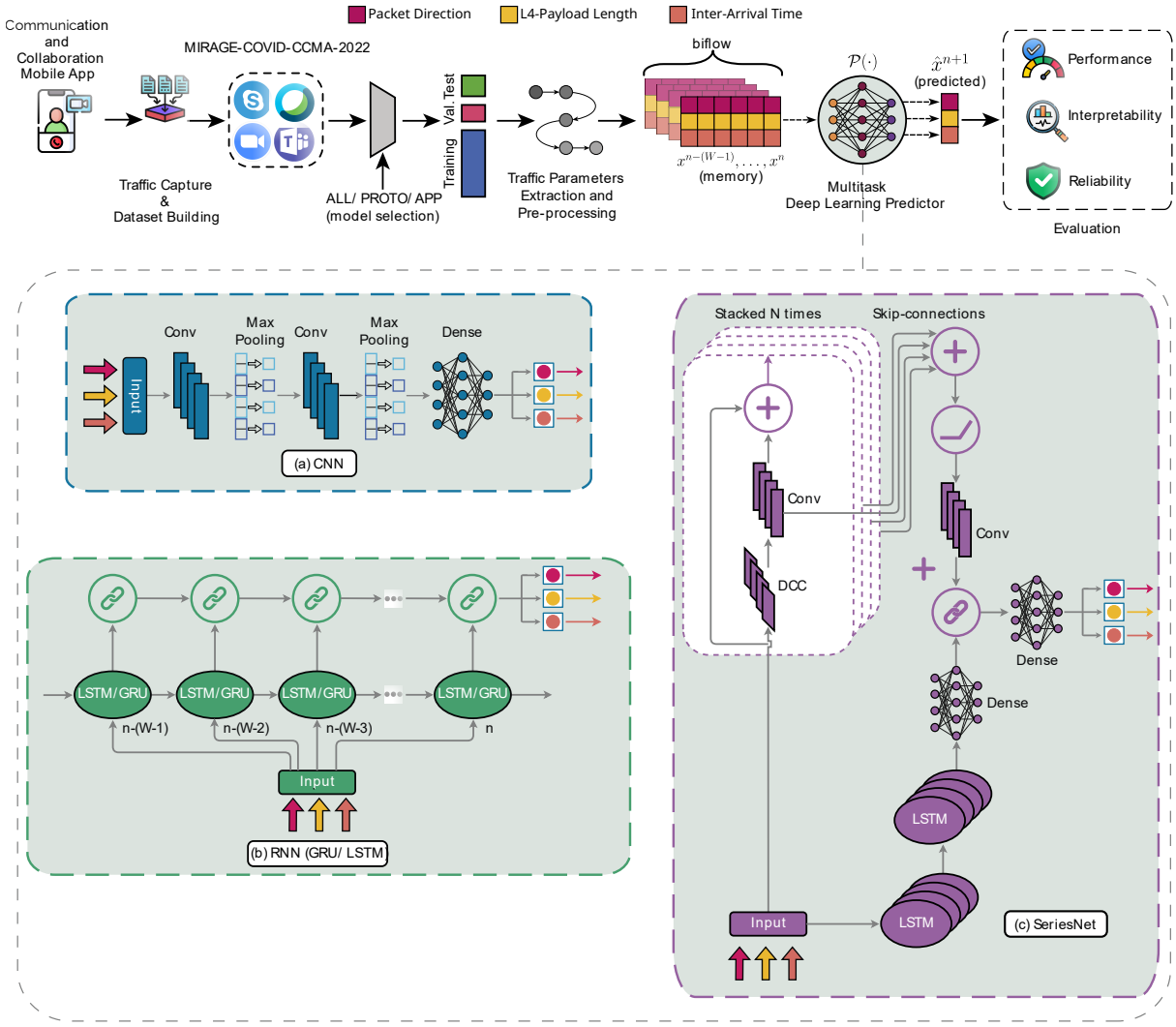


Figure 1: Workflow of the methodology defined for predicting the traffic of *communication-and-collaboration mobile apps* via multitask DL approaches and explaining/assessing the forecasting outcomes via XAI techniques.

First, in Sec. III.A, we provide a formal statement of the prediction task we have tackled and the corresponding solution we have devised. Hence, the section includes: (a) the description of the traffic parameters of interest (Sec. III.A.1); (b) the specification of the multitask DL architectures used to predict them (Sec. III.A.2); (c) the details about the adopted procedures for training these architectures (Sec. III.A.3).

Then, Sec. III.B details the XAI techniques used for inspecting the prediction outcomes. Accordingly, the section describes: (a) how interpret these models using post-hoc techniques (Sec. III.B.1); (b) how to analyze their reliability through calibration analysis (Sec. III.B.2).

We conclude in Sec. III.C with a description of the devised procedure to forecast traffic at a coarser (but arbitrary) granularity by exploiting packet-level predictions.

A. Multitask Deep Learning for Packet-Level Traffic Prediction

Our goal is to utilize DL approaches to predict network traffic generated by CC-apps at the *finest granularity*, i.e. at the packet level. To achieve this, we employ the widely-used bidirectional flow (biflow) as *traffic object* for our prediction task. A biflow embodies all the packets that share the same 5-tuple (`IP src, IP dst, port src, port dst, protocol`) in both upstream and downstream directions [9].

Specifically, given a biflow up to its n^{th} packet, we aim to predict P *traffic parameters* associated to the $(n+1)^{th}$ packet. In this case, the desired *output* of our DL architecture is represented by the vector $\boldsymbol{x}^{n+1}$. These predictions are based on the previous values of the same traffic parameters, which are stored in a *memory window* of size W. Then, the observations $\boldsymbol{x}^{n}, \ldots, \boldsymbol{x}^{n-(W-1)}$ are used as *input* to the DL architecture.

It is important to note that we construct the input using an *incremental windowing* approach that utilizes a sliding memory window with a unit stride. This allows us to incrementally add samples to the window until it reaches the maximum size of W. In cases where the prediction task has accumulated memory that is less than or equal to W, we apply left zero-padding to reach the desired window size. This enables predictions to be made on the initial part of the biflow (see later Sec. 1) and/or on biflows that are shorter than W.

Furthermore, we leverage *multitask architectures* that simultaneously address multiple prediction tasks, with each task focusing on one of the P parameters under consideration[2]. Therefore, we are targeting the design of a *single* DL model in the form:

$$\hat{\boldsymbol{x}}^{n+1} = \mathcal{P}(\boldsymbol{x}^{n}, \boldsymbol{x}^{n-1}, \ldots, \boldsymbol{x}^{n-(W-1)}) \tag{1}$$

where $\hat{\boldsymbol{x}}^{n+1}$ denotes the prediction vector associated to $\boldsymbol{x}^{n+1}$.

1) Traffic Parameters

Our objective is to predict three traffic parameters ($P = 3$), for the packets within a biflow. These parameters are: (i) the *direction* (`DIR`), which is a binary value indicating whether the packet is in the downstream or upstream direction; (ii) the *payload length* (`PL`), representing the size of the transport-layer payload measured in bytes; (iii) the *inter-arrival time* (`IAT`), which refers to the time interval between the arrival of two consecutive packets.

In this paper, we focus on predicting such $P = 3$ parameters since, generally, most network performance problems (e.g., loss, delay, jitter) occur at the packet level. Therefore, predicting the size, direction, and arrival time of the next packet can improve resource allocation within buffers and bandwidth [11]. These aspects are crucial for ensuring the quality of service and a seamless communication experience, particularly when dealing with CC apps [38].

2) DL Architectures

The DL architectures employed are depicted in Fig. 1. We underline that all DL architectures end with a number of dense layers (aka *heads*) equal to the number of parameters to be predicted (i.e., P, with $P = 3$ in Fig. 1), each using a Sigmoid activation function.

More in detail, for the convolutional model, we adopt a *1D-CNN* architecture (see Fig. 1a). It consists of two sequential convolutional layers with 32 and 64 filters, respectively, and a kernel size of 5. Each convolutional layer is followed by a max-pooling layer with a pool size of 3. The architecture further includes a dense layer with 128 neurons. All these layers employ the Rectified Linear Unit (ReLU) activation function.

On the other hand, we also consider two recurrent architectures: a Gated Recurrent Unit (*GRU*) and a Long Short-Term Memory (*LSTM*) network (see Fig. 1b). Both recurrent models are unidirectional and consist of 200 units, where the activation function employed is the Sigmoid.

In addition to the individual convolutional and recurrent architectures, we also employ a composite architecture, namely an extended version of the *SeriesNet* (see Fig. 1c). This DL model is based on Dilated Causal Convolution (DCC). More in detail, the overall architecture comprises 7 DCC layers. Each DCC has 32 filters, a dilation size of 2, and applies the Scaled Exponential Linear Unit (SELU) activation function. Furthermore, each DCC includes a residual connection that connects the input to the output. The last 2 DCC layers incorporate a dropout rate of 0.8. The sum of the parameterized skip connections from each DCC is passed through a ReLU activation function and used as input to a 1×1 convolutional layer. Finally, the output of this layer is concatenated with the stacking of two LSTM layers, each containing 200 units, and a dense layer with 128 neurons.

Although we restrict our analysis to the aforementioned DL models, we remark that *the methodology devised in this paper (i.e. multitask packet-level prediction, interpretability and trustworthiness of predictors, and tunable coarser-grained traffic prediction) is quite general and can be straightforwardly extended to other more sophisticated prediction models*.

3) Loss Specification and Training Details

Our main objective is to predict P traffic parameters for the next packet, which are stored in the vector $\boldsymbol{x}^{n+1}$. Consequently, the DL architecture is trained to minimize a *weighted* sum of the losses associated with the P prediction tasks considered, namely:

$$\mathcal{L}\left(\boldsymbol{\theta}_{\text{shared}}, \{\boldsymbol{\theta}_p\}_{p=1}^{P}\right) \triangleq \sum_{p=1}^{P} \lambda_p \, \mathcal{L}_p\left(\boldsymbol{\theta}_{\text{shared}}, \boldsymbol{\theta}_p\right) \tag{2}$$

where the weight λ_p indicates the importance level of the p^{th} task in the overall multitask objective function[3]. The shared parameters $\boldsymbol{\theta}_{\text{shared}}$ are associated with the layers

[2] In a previous study [9], we conducted experiments that demonstrated the advantages of multitask architectures in terms of prediction performance and complexity, as compared to single-task solutions. The results showed that multitask architectures outperformed single-task solutions in both aspects. Specifically, the multitask architectures exhibited better prediction performance, yielding more accurate results, while also reducing the overall training time. This finding suggests that employing a single deep learning architecture to predict each traffic parameter individually is less effective and more computationally expensive compared to utilizing multitask architectures.

[3] On the basis of preliminary results, not reported for brevity, herein, we set $\lambda_1 = \lambda_2 = 0.45$ and $\lambda_3 = 0.10$, where $p = 1$, $p = 2$, and $p = 3$ are associated to the `DIR`, `PL`, and `IAT` prediction task, respectively.

that are common to all tasks, while the parameters $\boldsymbol{\theta}_p$ are *specific* to the p^{th} task. Moreover, due to the multitask nature of these architectures, the specific loss function $\mathcal{L}_p(\cdot)$ to optimize depends on the prediction task being addressed. Specifically, we train the DL architectures to minimize the *binary cross-entropy* loss function for the prediction of the binary `DIR`. For the prediction of non-binary parameters, such as `PL` and `IAT`, we minimize the *Mean Squared Error (MSE)* between the predicted values and the actual traffic parameters associated with the $(n+1)^{th}$ packet of a given biflow.

In this study, we employ different training strategies that vary based on how traffic information is grouped. These strategies allow the models to capture traffic characteristics beyond those specific to an individual app, enabling a more comprehensive understanding of the traffic patterns. Notably, such strategies directly affect the number and complexity of DL models a network operator needs to develop, train, and deploy in the network. As a consequence, we examine the following *three training strategies* corresponding to different aggregation levels:

- **per-app models (`APP`)**: a separate model is created for each app. This means that there is a specific predictor $\mathcal{P}(\cdot)$ associated with each individual app. The models are trained using traffic data labeled with the corresponding app information (e.g., the Android package name).
- **per-proto models (`PROTO`)**: the models are designed to consider traffic based on the protocols used. However, the models do not differentiate between individual apps within each protocol.
- **overall-model (`ALL`)**: a single model is created to encompass all the apps. In other words, a single predictor $\mathcal{P}(\cdot)$ is utilized for all the apps. The model is trained using the entire traffic dataset, without taking into account the specific information about the individual app (or protocol) that generated the traffic.

We remark that the training strategies described do not pose any constraint on the model used for traffic prediction. Hence, each of them can be applied to all the DL traffic predictors considered in this work. However, note that to be practically deployed, per-app models require the presence of an upstream traffic classifier to properly route network traffic to the proper prediction model, namely a more complex network setup compared to the other training strategies.

B. XAI-tools for Traffic Prediction

The black-box nature of DL models makes it difficult for network operators to trust them, especially when the results of their decisions could have a significant impact on the network. In this scenario, XAI can assist network operators in making improved decisions regarding the integration of AI into their networks by offering transparency, building trust, identifying problems, and ensuring compliance with regulations [36].

1) Interpretability of `DIR` Prediction via DEEP SHAP

In this section, we present the methodology employed to investigate the *interpretability* of the results obtained from a probabilistic binary classification task, specifically the prediction of `DIR` values, using DL architectures. We recall that such an interpretability analysis provides a means to *explain* the model, determining whether the predictions are more influenced by specific parts of the input traffic, also uncovering potential biases. It can also enable the improvement of model performance and the assessment of robustness and vulnerabilities (e.g., how much the model is susceptible to adversarial attacks). Additionally, from a *transparency* viewpoint, interpretability techniques can facilitate the validation of the prediction outcome by providing insights into the internal mechanisms of DL architectures, thus making the resulting decisions more trustworthy.

To start interpreting DL architectures, we adopt a simpler explanation model, denoted as $g(\cdot)$, which is designed to closely approximate the original model $f(\cdot)$. To explain the predictive behavior of a DL-based traffic predictor, we use the model $f(\cdot)$ as the soft output associated with the generic direction (i.e. p_{up}/p_{dw}). This allows us to determine which inputs contribute the most to the confidence probability value that is associated with a given direction. In the following, our focus lies on *local methods*, which explain the model $f(\boldsymbol{x})$ within the neighborhood of a specific instance $\boldsymbol{x}$, referred to as an explanation of the input packet sequence in this case. These local explanations utilize *simplified inputs* $\boldsymbol{x}'$, which are mapped to the original inputs $\boldsymbol{x}$ through the mapping $\boldsymbol{x} = \boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{x}')$.

Herein, we adopt the *Additive Feature Attribution (AFA)* functional form as the explanation model $g(\cdot)$ for our analysis. The AFA model is defined as:

$$g(\boldsymbol{z}^{'}) = \phi_0 + \sum_{m=1}^{M} \phi_m \, z_m^{'} \tag{3}$$

where $\boldsymbol{z}^{'} \in \{0,1\}^M$, M represents the number of simplified inputs, and $\phi_m \in \mathbb{R}$. This specific class of explanation models assigns an "effect" ϕ_m to each input, indicating its contribution. Consequently, the output of the original model $f(\boldsymbol{x})$ can be approximated by summing the effects of all input attributions.

A widely used method for computing AFA solutions is through the application of *Shapley values*, which have their origins in cooperative game theory. These values quantify the contribution of a particular player, denoted as m, to the overall payoff achieved by the entire coalition $\mathcal{C}$, denoted as $v(\mathcal{C})$. Specifically, the overall payoff is obtained by first evaluating the payoff of all possible subsets $\mathcal{S} \subset \mathcal{C}$ of cooperating players which include m. Secondly, the effect of excluding player m from each $\mathcal{S}$ on the payoff is evaluated, namely $v(\mathcal{S}) - v(\mathcal{S} \setminus m)$. The final m^{th} Shapley value is then obtained by taking the (weighted) average of such differences over all the $\mathcal{S}$. When applying this method to explain a DL-based model, the input data is mapped to the players of the

cooperative game, while the output of the DL architecture, represented by $f(\boldsymbol{x})$, corresponds to the payoff function.

However, the exact computation of Shapley values becomes exponentially complex as the size of the input, denoted as M, increases. To address this challenge, we employ an approximation method known as *SHapley Additive exPlanation (SHAP)*, which efficiently calculates the Shapley values. This approximation eliminates the need to retrain the models by approximating the Shapley values through the conditional expectation $f(\boldsymbol{h}_{\boldsymbol{x}}(\boldsymbol{z}^{'})) \approx \mathbb{E}\left\{f(\boldsymbol{z})|\boldsymbol{z}_{\mathcal{S}}\right\}$, where $\mathcal{S}$ represents the set of non-zero indices within $\boldsymbol{z}^{'}$.

Specifically, in our study, we utilize DEEP SHAP [39], an adaptation of the *DeepLIFT* algorithm designed for evaluating SHAP values on neural architectures. DeepLIFT employs a compositional approximation of SHAP values by using the output expectation as the reference value. It also utilizes explicit Shapley equations for consistent linearization. The reference value is a user-defined parameter that is chosen to be an uninformative background value for the m^{th} input.

To be more specific, we use DEEP SHAP to explain the soft-output associated with the predicted DIR, denoted as $\hat{p}(\boldsymbol{x})$, for each input sequence represented by $\boldsymbol{x} = (\boldsymbol{x}^{n}, \boldsymbol{x}^{n-1}, \ldots, \boldsymbol{x}^{n-(W-1)})$. For a given input sequence $\boldsymbol{x}$, we interpret the SHAP value ϕ_m as the importance value of the m^{th} traffic parameter composing $\boldsymbol{x}$ in forming the confidence p_i associated with the i^{th} direction for the next packet. It is worth noting that since $\phi_m \in \mathbb{R}$, which means that values can be negative, we should interpret the importance values as follows: positive values increase the confidence in the i^{th} direction compared to its average value, while negative values decrease it. Additionally, the sum of the SHAP values equals the soft-output value ($p_i(\boldsymbol{x})$) minus the *base output*. The *base output* represents the average of the same soft-output value obtained from the samples associated with the background set, i.e. $\mathbb{E}\{p_i\}$.

Finally, because the absolute importance range of the m^{th} traffic parameter may significantly fluctuate over different input packet sequences due to the variability of soft outputs, we use *global explanations* obtained by combining *per-packet-sequence* and *normalized* explanations obtained from DEEP SHAP. Specifically, our global explanation approach involves the initial computation of *normalized* SHAP values:

$$\widetilde{\phi}_m \triangleq \phi_m \,/\, \sum_{m=1}^{M} \phi_m \tag{4}$$

By using $\widetilde{\phi}_m$ instead of ϕ_m, we can focus on the relative importance of each input and derive importance measures that are independent of the specific confidence levels of the architecture. This normalization ensures consistent aggregation over the test samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ and removes dependence on the architecture's peculiar confidence levels, which can be generally higher or lower. Additionally, we specifically concentrate on aggregating correctly classified DIR samples [33]. This selection allows us to focus on the accurate behavior of a DL-based traffic predictor and interpret its counter-intuitive (while right) decisions *a posteriori*.

2) Assessing the Reliability of DIR Prediction

Other than the performance of the considered DL-based traffic predictors, it is of great importance to evaluate the *reliability* to soft-estimates associated with the prediction of discrete-valued parameters, such as DIR in our case. Reliability evaluation is fundamental in many critical scenarios and constitutes a building block of XAI since it assesses the degree of trustworthiness in providing prediction outputs with high confidence. In other words, it evaluates if DL-based traffic predictors are calibrated (or not) and consequently, if the provided predictions are reliable (or not).

Formally speaking, given an input sample $\boldsymbol{x}$ to the DL-based traffic predictor under analysis, we will analyze the reliability of the confidence vector $\boldsymbol{p}(\boldsymbol{x}) = [p_{up}(\boldsymbol{x})\, p_{dw}(\boldsymbol{x})]$ and of the confidence associated to the predicted DIR $\hat{p}(\boldsymbol{x}) = \max\{p_{up}(\boldsymbol{x}), p_{dw}(\boldsymbol{x})\}$.

In what follows, we introduce a graphical visualization and two metrics to assess calibration [40]. Indeed, a *confidence-calibrated* classifier is such that for each sample, the confidence $\hat{p}$ in the predicted direction equals $\Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\}$, where x_{dir}^{n+1} (resp. $\hat{x}_{dir}^{n+1}$) is the true (resp. predicted) direction. That is, when reporting a confidence of e.g. 80% the predictor actually reaches 80% accuracy (i.e. the confidence value was neither excessively optimistic nor pessimistic).

To visualize the above property for varying $\hat{p}$, we use the *reliability diagrams*, which show the accuracy as a function of the confidence (i.e. $\Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\}$ vs. $\hat{p}$) and compare it with the ideal $\Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\} = \hat{p}$ line, corresponding to a perfectly-calibrated classifier. These diagrams are evaluated by partitioning the predictions into M equally-spaced bins and computing the accuracy for each of them. Let B_m be the set of evaluated samples such that the confidence associated to the predicted app falls within the interval $I_m \triangleq (\frac{m-1}{M}; \frac{m}{M}]$, the corresponding bin accuracy equals $\mathrm{acc}(B_m) = |B_m|^{-1} \sum_{n \in B_m} 1(\hat{x}_{dir}^{n+1}(s) = x_{dir}^{n+1}(s))$, where $x_{dir}^{n+1}(s)$ and $\hat{x}_{dir}^{n+1}(s) \triangleq \arg\max_{i=\{dw,up\}} p_i(s)$ are the true and predicted labels for the s^{th} sample, respectively. Confidence values range in $[1/2\,,\,1]$, since DIR prediction maps into a binary classification task.

To obtain concise metrics of the deviation from a perfect calibration, we integrate the above diagrams with the *Expected Calibration Error (ECE)*, defined as

$$\mathrm{ECE} \triangleq \mathbb{E}_{\hat{p}}\left\{\left|\Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\} - \hat{p}\right|\right\} \tag{5}$$

and the *Maximum Calibration Error (MCE)*, defined as

$$\mathrm{MCE} \triangleq \max_{\hat{p}} \left|\Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\} - \hat{p}\right|. \tag{6}$$

The former metric represents the expected absolute deviation between the confidence and the confidence-conditional accuracy, whereas the latter is the maximum absolute deviation

from the identity line [40]. They can be approximately calculated as

$$\mathrm{ECE} \approx \sum_{m=1}^{M} \left(|B_m| \ / N\right) |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \quad (7)$$

and

$$\mathrm{MCE} \approx \max_{m=1,\dots,M} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)| \quad (8)$$

respectively. The above expressions are based on the overall number of tested samples S and the averaged confidence within the bin B_m. The latter equals $\mathrm{conf}(B_m) = |B_m|^{-1} \sum_{s \in B_m} \hat{p}_{(s)}$, where $\hat{p}(s) \triangleq \max\{p_{dw}(s), p_{up}(s)\}$ denotes the predicted confidence of the s^{th} sample.

The ECE and MCE metrics only consider the confidence in the predicted app, while ignoring the other scores in the softmax distribution. A *stronger* definition of calibration requires the probabilities of all the classes in the softmax distribution to be calibrated, namely to have p_i equal to $\Pr\{\ell = i | \, p_i\}$ for $i = 1, \dots, L$, i.e. having 80% confidence for the i^{th} app leads to 80% probability of observing that app. A concise metric that relies on the above stronger calibration definition is the *Class-Wise Expected Calibration Error (CW-ECE)* [41], defined as

$$\text{CW-ECE} \triangleq \frac{1}{L} \sum_{i=1}^{L} \mathbb{E}_{p_i} \left\{ |\Pr\{\ell = i | \, p_i\} - p_i| \right\}. \quad (9)$$

Such a metric is evaluated as the class-wise sum

$$\text{CW-ECE} = \frac{1}{L} \sum_{i=1}^{L} \text{CW-ECE}_i \quad (10)$$

where

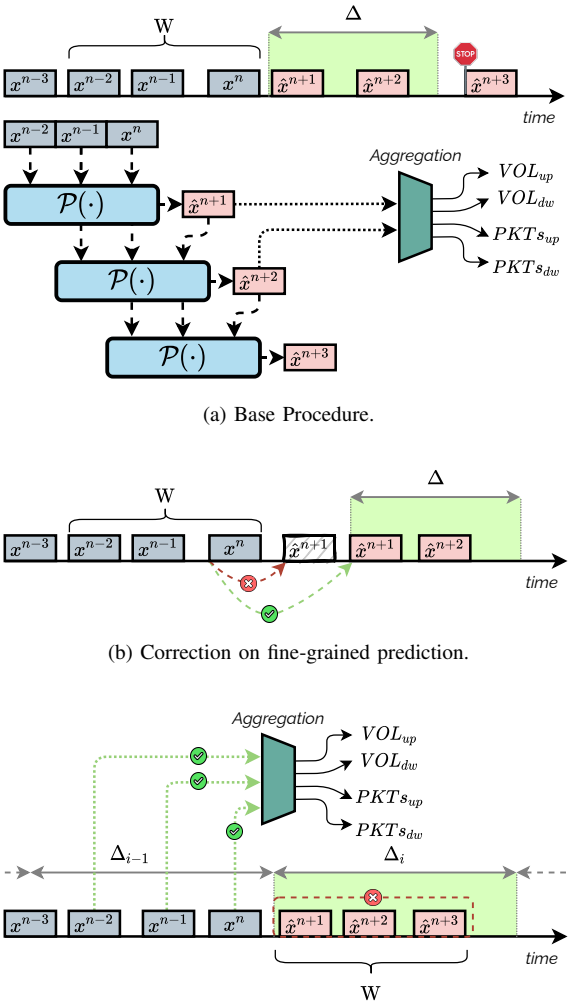
$$\text{CW-ECE}_i \approx \sum_{m=1}^{M} \frac{|B_{m,i}|}{N} \left| \varrho(B_{m,i}) - \mathrm{conf}(B_{m,i}) \right|. \quad (11)$$

In the latter definition, $B_{m,i}$ denotes the set of samples whose prediction for the i^{th} app p_i falls within the m^{th} bin, and $\mathrm{conf}(B_{m,i})$ (resp. $\varrho(B_{m,i})$) the corresponding bin-averaged confidence probability (resp. the proportion of samples labeled as the i^{th} app).

C. From Fine-Grained To Aggregate Traffic Prediction

In this section, we define a first approach that can capitalize on the benefits of having a multitask packet-level predictor.

Specifically, our goal is to exploit the packet-level predictions to forecast aggregates of traffic within a future time interval (Δ), also known as the time horizon. To achieve this, we employ the *recursive* methodology depicted in Fig. 2, which iteratively utilizes the packet-level predictions (i.e. related to `DIR`, `PL`, and `IAT`) to predict the traffic to a coarser granularity (i.e. the number of packets and data volume in both upstream and downstream directions) for the next Δ. This is obtained by *repeatedly* using the fine-grained predictor $\mathcal{P}(\cdot)$. It is worth noting that the main strength of this approach lies in its *flexibility* which makes it possible to *tune* the Δ parameter at the operational stage (i.e. at run-time). This avoids the burden of learning a DL traffic predictor for each Δ of interest.



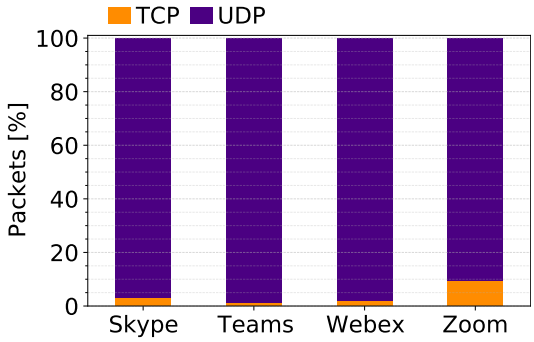
(a) Base Procedure.

(b) Correction on fine-grained prediction.

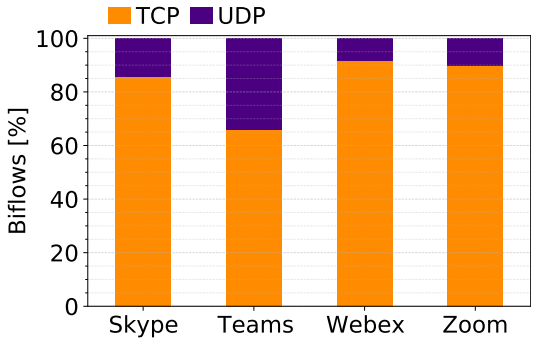
(c) Correction on coarse-grained prediction.

Figure 2: Proposed *recursive* approach we used to predict aggregates of traffic (i.e., number of packets and traffic volume in both upstream and downstream directions) over a time horizon Δ: we exploit a fine-grained (i.e. at the packet level) predictor $\mathcal{P}(\cdot)$ with a memory window of size $W(=3$ in the figure) (a).
Correction on fine-grained prediction (b): the process ensures that all predicted packets fall within the prediction interval Δ.
Correction on coarse-grained prediction (c): the process assumes that the traffic in the next Δ_i is the same as that in the previous Δ_{i-1} when the memory (of size W) consists of only predicted packets.

More in detail, in our procedure (depicted in Fig. 2a), after each Δ, we take the sequence of the most recent W packets $\{\boldsymbol{x}^n, \boldsymbol{x}^{n-1}, ..., \boldsymbol{x}^{n-(W-1)}\}$, and recursively obtain a series of predictions by means of $\mathcal{P}(\cdot)$, denoted as $\hat{\boldsymbol{x}}^j$, such that: $\sum_j \hat{x}^j_{\texttt{IAT}} \leq \Delta$. To do this, due to the fixed memory of $\mathcal{P}(\cdot)$, at step $j > 1$, the prediction $\hat{\boldsymbol{x}}^j$ is obtained by removing the oldest packet from the memory and adding

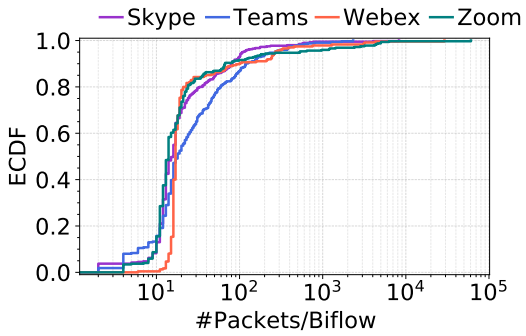


(a) Percentage of Packets.

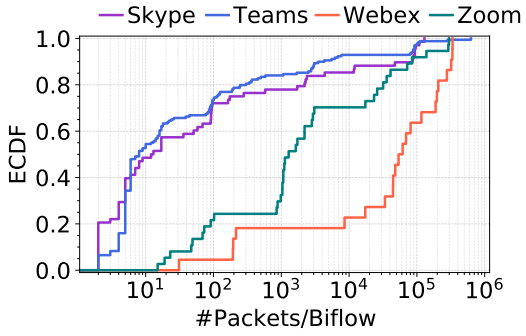


(b) Percentage of Biflows.

Figure 3: Transport-level protocol distribution in terms of share of packets (a) and biflows (b).



(a) TCP.



(b) UDP.

Figure 4: Biflow-length distribution (in terms of the number of packets) for each app with respect to TCP (a) and UDP (b) protocols. Values on the x-axis are reported in log scale.

the prediction obtained at the previous step $\hat{\boldsymbol{x}}^{j-1}$ as the last observed packet.

In addition, to handle any critical situations that might occur, we apply two corrections to both fine- and coarse-grained predictions. The correction on the fine-grained predictions (depicted in Fig. 2b) ensures that all predicted packets fall within the next Δ. To this end, we enforce that the first predicted packet ($\hat{\boldsymbol{x}}^{n+1}$) arrives at the beginning of the next Δ interval if the corresponding predicted `IAT` would erroneously place such packet before the beginning of the interval. On the other hand, with the correction on the coarse-grained predictions (depicted in Fig. 2c), we avoid the error accumulation of the recursive procedure when leveraging a memory encompassing only predicted packets. Accordingly, in such cases, the predicted aggregate traffic is taken as that observed in the previous Δ_{i-1} interval.

At the end of the above procedure, the set of `PL`s and `DIR`s of the predicted packets are used to compute the volume and number of packets in both upstream and downstream directions (shown as *Aggregation* in Fig. 2). In the following, we refer to these predicted traffic aggregates as $\texttt{VOL}_{\texttt{up}}$, $\texttt{VOL}_{\texttt{dw}}$, $\texttt{PKTS}_{\texttt{up}}$, and $\texttt{PKTS}_{\texttt{dw}}$.

IV. Experimental Setup

In this section, we provide a comprehensive overview of the experimental setup, including details about the apps' and related activities' selection rationale (Sec. IV.A), the collected dataset (Sec. IV.B), and the evaluation metrics (Sec. IV.C) employed for assessing the performance of the prediction models.

A. Apps' and Activities' Selection Rationale

Nowadays CC apps, such as those used for business meetings, classes, and social interaction, are massively exploited in everyday life after their adoption was fueled due to the pandemic years [1]. Therefore, we have specifically selected a subset of *four* CC apps from the MIRAGE-COVID-CCMA-2022 dataset based on their popularity [42]: `Skype`, `Teams`, `Webex`, and `Zoom`.

As previously mentioned, our experimentation focused on specific activities associated with these apps that include:

- Audio-call (`ACall`): a two-way audio transmission between two participants, without any video component.
- Chat (`Chat`): a conversation between two participants, where they exchange textual messages and/or multimedia content such as images or GIFs.
- Video-call (`VCall`): multiple attendees who can transmit both video and audio; this category encompasses various scenarios, including video calls between two or more attendees and webinars or live events with multiple participants.

By selecting these CC apps and their corresponding activities, we aim to capture and analyze the usage patterns and network traffic characteristics associated with different modes of communication.

B. Dataset Description, Processing and Characterization

In this study, we utilize our MIRAGE-COVID-CCMA-2022 public dataset that we have publicly released to foster replicability and reproducibility.[4] In detail, the dataset was collected by students and researchers of the University of Napoli "Federico II" between April and December 2021. The dataset was obtained using the MIRAGE architecture [43], which was specifically optimized for capturing and generating traffic from CC apps.

The experimenters employed three mobile devices: a Google Nexus 6 and two Samsung Galaxy A5, all running Android 10. During each capture session, the experimenters engaged in specific activities using various CC apps, aiming to generate traffic that represents common usage scenarios[5]. Each session produced a PCAP traffic trace and additional `netstat`[6] log files containing information about established network connections. These log files were used to create the ground truth, which involved labeling each biflow with (i) the *Android package-name* of the app and (ii) the specific *activity* performed by the user operating the device[7].

Then, to obtain the P packet parameters from the raw packet sequences, we perform some preprocessing steps. Firstly, we eliminate packets with zero payloads. Next, we recalculate the `IAT`s, ensuring a minimum precision of 1μs. To remove the influence of outliers, we saturate the recalculated `IAT`s to the 99^{th}-percentile value, which is determined as 197.90 ms based on their distributions. Finally, we apply a min-max scaler to normalize `PL` and `IAT` within the range of $[0, 1]$. This scaling procedure is a common practice when utilizing DL models, as described in [9].

Fig. 3 depicts the percentage of packets and biflows related to the TCP and UDP protocols for the traffic used in this study. In particular, we can observe that all the apps tend to generate significantly more TCP biflows: from $\approx 65\%$ (`Teams`) to $\approx 90\%$ (`Webex`) of the total number of biflows. Moreover, although the share of UDP biflows is significantly lower than the TCP ones, UDP packets correspond to 90% of the packets of each app, at least. The above results are likely due to the fact that CC apps combine both protocols to ensure a balance between reliable data delivery and low latency transmission (e.g., to implement control- and data-plane functionalities, respectively), meeting the requirements of communication-and-collaboration scenarios.

To deepen the above characterization, Fig. 4 reports for each app the distribution of the number of packets per biflow, according to the transport-layer protocol. Specifically, regarding the TCP protocol (Fig. 4a), we observe that $\approx 90\%$ of the biflows of each app have at most 100 packets. Also, with the only exception of `Teams`, 20% of biflows have less than ≈ 10 packets. Conversely, when looking at the UDP protocol (Fig. 4b), the observed behavior is more related to the specific app. While for `Skype` and `Teams` $\approx 70\%$ of the biflows has less than 100 packets, `Webex` and `Zoom` expose longer UDP biflows with more than 100 packets in more than 80% of the cases.

Finally, starting from the collected traffic, we applied the incremental windowing approach, described in Sec. III to construct the input for the DL architectures. More in detail, the resulting dataset consists of 806 k samples for `Skype`, 1.9 M samples for `Teams`, 2.5 M samples for `Webex`, and 1.2 M samples for `Zoom`.

C. Evaluation Procedure and Metrics

The evaluation of traffic prediction strategies is conducted using a robust stratified five-fold cross-validation setup: 80% of the biflows are allocated for the training/validation set, with 80% of this subset being the actual training set and 20% assigned as the validation set; the remaining 20% of biflows are designated as the test set for evaluation purposes. The validation set is utilized to effectively implement the early-stopping technique, which helps prevent overfitting during training. Consequently, we calculate the average value and standard deviation of each evaluation metric over the five folds.

To assess the prediction performance of the binary `DIR` traffic parameter, we employ the *G-mean* metric:

$$\text{G-mean} \triangleq \sqrt{\rho_{dir}^{\text{dw}}\, \rho_{dir}^{\text{up}}} \tag{12}$$

where $\rho_{dir}^{\text{dw}} \triangleq \Pr(\hat{x}_{dir} = \text{DW} \mid x_{dir} = \text{DW})$ and $\rho_{dir}^{\text{up}} \triangleq \Pr(\hat{x}_{dir} = \text{UP} \mid x_{dir} = \text{UP})$, and the variable x_{dir} represents the sequence of actual `DIR`s, while the variable $\hat{x}_{dir}$ represents the predicted `DIR`s. The terms DW and UP indicate the downstream and upstream directions, respectively. The probability of accurately predicting the `DIR` of downstream/upstream packets (ρ_{dir}^{dw}/ρ_{dir}^{up}) is computed by dividing the number of correct predictions in a specific direction by the total number of samples associated with that true direction.

Conversely, to evaluate the prediction performance of `PL` and `IAT`, we employ the *Root Mean Squared Error (RMSE)* metric:

$$\text{RMSE}_p \triangleq \sqrt{\frac{1}{\bar{N}} \sum_{j=1}^{\bar{N}_B} \sum_{n=1}^{\bar{N}_j - 1} \left[\hat{x}_p^{n+1}(\bar{B}_j) - x_p^{n+1}(\bar{B}_j)\right]^2} \tag{13}$$

[4] http://traffic.comics.unina.it/mirage/mirage-covid-2022

[5] Each traffic capture session spanned $15 \sim 80$ mins based on the activity and 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.

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

[7] Activity labels were manually assigned based on the knowledge of the individual activity performed by the user during the specific capture.

with $\bar{N}$ being the total number of predictions, $x_p^{n+1}(\bar{B}_j)$ the value of the p^{th} traffic parameter (with $p \in \{\texttt{PL}, \texttt{IAT}\}$) observed for packet $n+1$ from the j^{th} biflow $\bar{B}_j$ (of length $\bar{N}_j$), and $\hat{x}_p^{n+1}(\bar{B}_j)$ the corresponding value provided by the prediction model.

To provide a meaningful reference point for comparison, we compare the performance of DL models against that of a packet-level baseline predictor. Specifically, the latter makes predictions by assuming that the next observation value is equal to the current observation value: $\hat{x}_{\text{RLP}}^{n+1} \triangleq x^n$. We refer to such a baseline as `Repeat-Last-Packet` (`RLP`).

Similarly, for the prediction of traffic aggregates (i.e. number of packets and traffic volume), we consider `Repeat-Last-Aggregate` (`RLA`) as a reference baseline. The `RLA` predictor forecasts the aggregates of traffic within a time interval Δ as equal to the aggregates of traffic observed during the previous time interval.

D. Implementation Details

For implementing and testing the DL architectures described in Sec. III.A we exploit the model provided by `Keras` (https://keras.io) Python API running on top of `TensorFlow 2` (https://www.tensorflow.org/). Input data are formatted in Parquet and optimally managed via `Apache PyArrow` (https://arrow.apache.org/). Data pre- and post-processing have been performed mainly by means of `numpy` (https://numpy.org/) and `pandas` (https://pandas.pydata.org/) libraries. For the evaluation metrics reported in Sec. IV.C, we use the implementation of `scikit-learn` (https://scikit-learn.org/). Finally, the graphical data representation has been obtained using `matplotlib` (https://matplotlib.org/) and `seaborn` (https://seaborn.pydata.org/) libraries.

To ensure a fair comparison among the various architectures, we set uniform values to all adjustable hyperparameters related to the training process of the models. We train all the models for a total of 100 epochs using the Adam optimizer, with a learning rate of 0.001, and a batch size of 32. To avoid overfitting, we use a validation-based early-stopping technique, where we set the patience and min_delta parameters to 4 and 0.0001, respectively. When using DEEP SHAP, we used a background set of 500 samples randomly selected from the training set of the current fold.

To foster the replicability and reproducibility of our analysis, we have publicly released the code of the DL architectures leveraged herein, along with pre-processed data, hyperparameters setting, and example usages.[8]

V. Experimental Evaluation

In the following, we first investigate traffic prediction performance for all apps and activities, attained by different multitask DL models, with the goal of understanding whether it is better to train a single model for all apps or a specific one for each of them (Sec. V.A). Then, we delve into the results, investigating per-packet performance and error distributions (Sec. V.B). Concerning AI trustworthiness, we focus on the prediction of the direction of the next packet: we explain the model behavior via DEEP SHAP (Sec. V.C), and we assess its reliability via calibration analysis (Sec. V.D). Next, we deepen the impact of transport-layer protocols on packet-level prediction performance (Sec. V.E). Finally, we analyze the suitability of the packet-level traffic prediction approach for forecasting aggregate traffic characteristics (i.e., number of packets and traffic volume) in a time interval of fixed but arbitrary duration (Sec. V.F).

Table 3: ECE, MCE, and CW-ECE related to `DIR`-prediction when using $\texttt{CNN}_{\texttt{ALL}}$ with a memory size set to $W = 10$. Results refer to the prediction of traffic generated by each app, regardless of the specific activity. For each metric, the best and the worst calibrated apps are highlighted in green and red, respectively. Results are in the form $avg. \pm std.$ obtained over 5-folds.

App	ECE [%]	MCE [%]	CW-ECE [%]
Skype	3.69 (±0.59)	6.63 (±2.23)	4.46 (±1.35)
Teams	7.02 (±4.16)	28.57 (±18.22)	9.69 (±5.44)
Webex	2.11 (±0.83)	4.95 (±2.59)	4.36 (±1.98)
Zoom	5.35 (±2.42)	10.26 (±3.12)	6.60 (±2.51)

Table 4: ECE, MCE, and CW-ECE related to `DIR`-prediction when using $\texttt{CNN}_{\texttt{ALL}}$ with a memory size set to $W = 10$. Results refer to the prediction of traffic generated by each activity, regardless of the app activity. For each metric, the best and the worst calibrated activities are highlighted in green and red, respectively. Results are in the form $avg. \pm std.$ obtained over 5-folds.

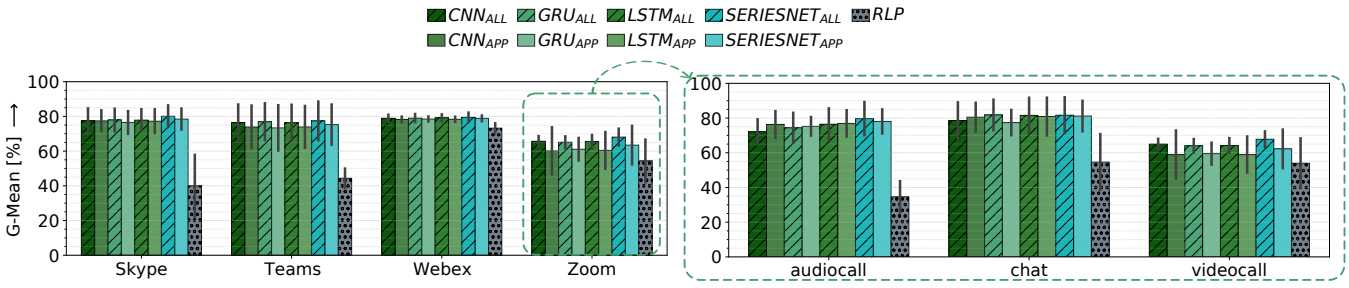
Activity	ECE [%]	MCE [%]	CW-ECE [%]
ACall	2.02 (±0.78)	3.77 (±1.65)	3.38 (±2.13)
Chat	2.85 (±0.94)	9.49 (±2.91)	3.94 (±1.79)
VCall	5.15 (±3.99)	26.16 (±20.07)	6.32 (±4.45)

A. Do we need a dedicated model for each app?

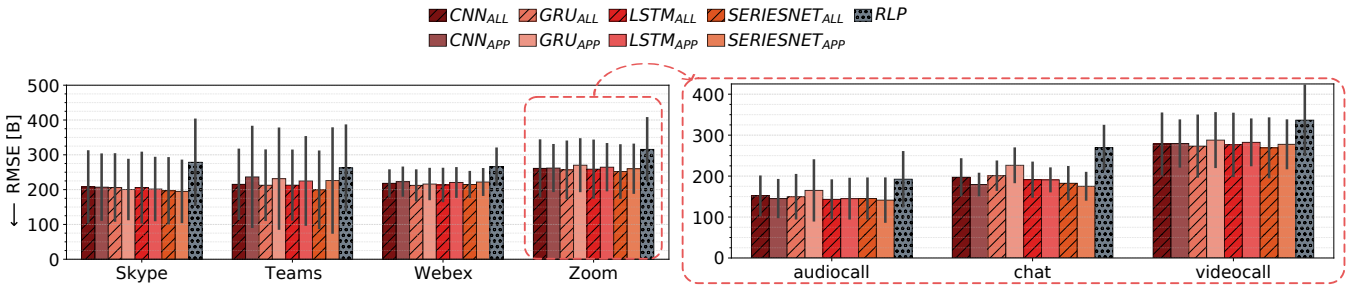
Preliminary results—not reported for brevity—showed that $W = 10$ constitutes the best trade-off between the complexity of the model (growing with W) and the effectiveness of the prediction. Therefore, we provide an overview of the performance of the multitask DL models used for predicting `DIR`, `PL`, and `IAT` by adopting a memory window size $W = 10$. Specifically, we evaluate the effect of having one single overall model for all apps instead of one separate model for each of them. To this end, Fig. 5 summarizes the performance of all DL models trained according to different strategies (i.e. `ALL` vs. `APP`) with respect to all the apps considered. In addition, for each app, we also provide the performance achieved by the corresponding `RLP` *baseline*.

DL models always outperform `RLP`, especially on `Skype`: we observe the largest gap on all traffic parameters (i.e. $\approx -40\%$ G-mean on `DIR`, and $\approx +84$ B and $\approx +26$ ms RMSE on `PL` and `IAT`, respectively). Conversely, we observe that for the same parameters, the smallest gap is obtained on

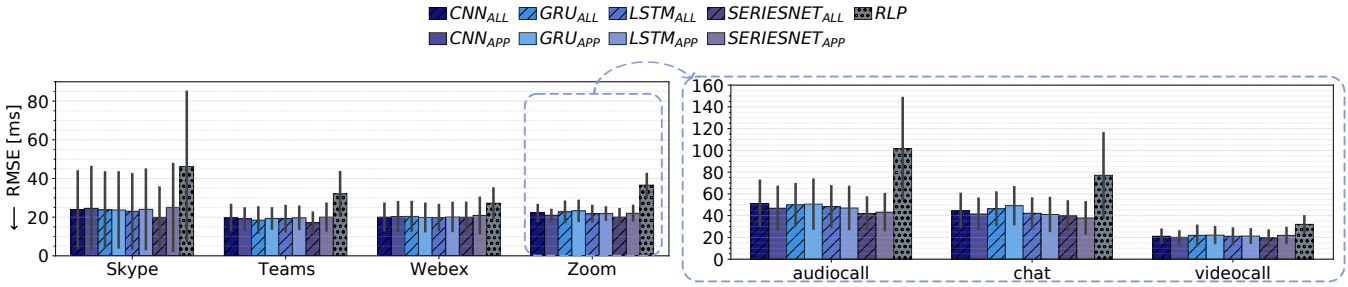
[8] https://github.com/IdioGuarino/AFTER



(a) DIR.



(b) PL.



(c) IAT.

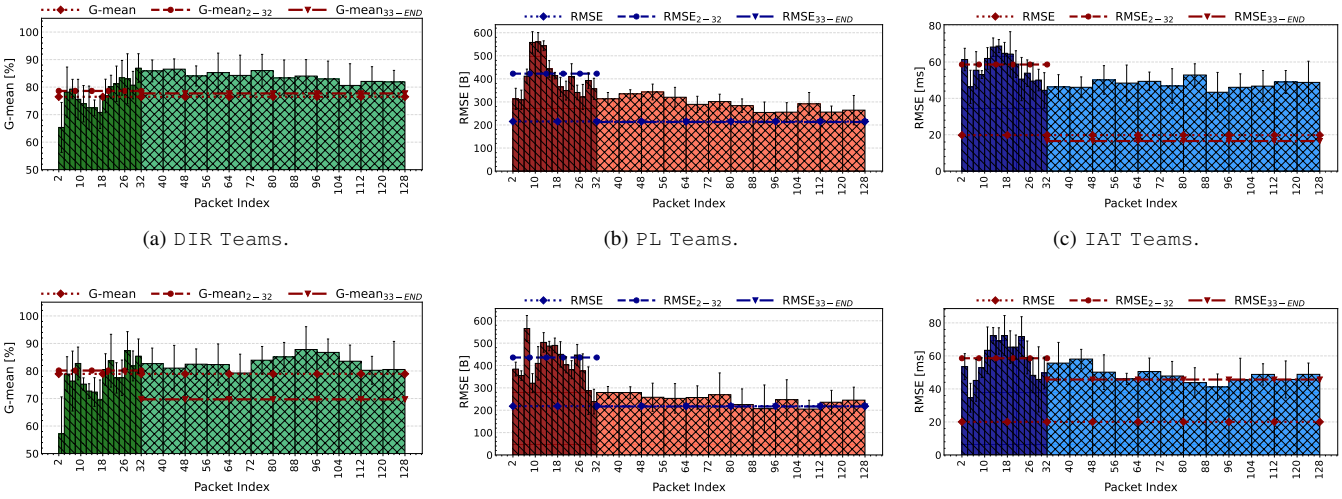
Figure 5: Prediction performance of CNN, LSTM, GRU, SeriesNet, and RLP on DIR (a), PL (b), and IAT (c). Results on the *left* refer to the prediction of traffic generated by *each app regardless of the specific activity*. Results on the *right* refer to Zoom *when performing a specific activity* (b, d, f)—i.e. Chat, ACall, and VCall. Memory size is set to $W = 10$ and both APP and ALL training strategies are considered. The arrow close to y-axis shows the desired trend. Results are in the form $avg. \pm std.$ obtained over 5-folds.

Webex (i.e. $\approx -6\%$ G-mean, and $\approx +54$ B and $\approx +7$ ms RMSE on PL and IAT, respectively).

On the other hand, looking at the training strategies, we note that the overall model (ALL) outperforms or equals per-app models (APP)[9]. Despite this result may seem counterintuitive, it is strongly related to the intrinsic nature of CC apps and, accordingly, to their generated traffic. Indeed, since different activities (i.e. ACall, VCall, and Chat) are shared among different CC apps, exploiting a traffic-prediction model tailored for a given app is likely to provide slightly degraded performance compared to the ALL one. On the other hand, using an ALL model has inherent advantages related to the unnecessity of training, deploying, and managing multiple models and of having an upstream traffic classifier to guide the selection of the specific APP model.

Additionally, by taking into account the prediction of a specific traffic parameter, we observe that both the examined strategies achieve $\approx 80\%$ G-mean when predicting DIR (Fig. 5a). Performance on Zoom represents an exception, showing a G-mean of $\approx 65\%$. A similar outcome is also observed for PL prediction (Fig. 5b) where Zoom exhibits a higher RMSE of $\approx +50$ B compared to the other apps. A slightly different—although more stable—performance picture is evident for IAT prediction (Fig. 5c), for which Skype exhibits the worst RMSE (i.e. $\approx +5$ ms than the other apps) and presents also the highest variability. Further investigations (not reported for brevity) have shown that this phenomenon can be attributed to the higher variability of IATs corresponding to ACall and VCall activities.

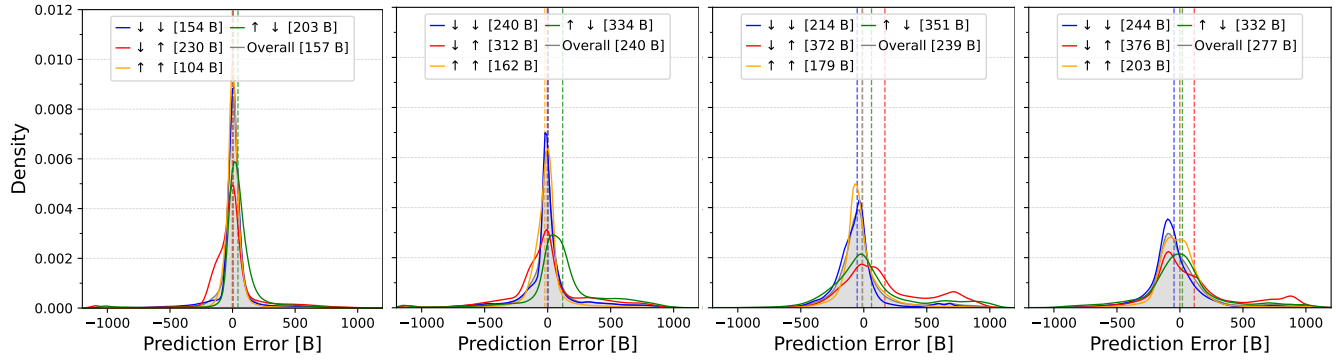
[9]Similar outcomes were also obtained by comparing the ALL and APP training strategies using well-known ML models—i.e., a Random Forest Regressor for PL and IAT prediction and a Random Forest Classifier for DIR prediction—instead of DL models.



(a) DIR Teams. (b) PL Teams. (c) IAT Teams.

(d) DIR Webex. (e) PL Webex. (f) IAT Webex.

Figure 6: Per-packet index performance in terms of G-mean for DIR and RMSE for PL and IAT of the first 128 packets. Results refer to Teams (a-c) and Webex (d-f) and are in the format $avg \pm std$ obtained over the 5-folds. Horizontal lines report the overall G-mean/RMSE, the G-mean/RMSE of the first 32 packets (G-mean/RMSE$_{2-32}$), and the G-mean/RMSE of the remaining ones (G-mean/RMSE$_{33-END}$) of each biflow.



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

Figure 7: Probability density of prediction error on PL. The probabilities are conditioned to different combinations of (predicted, true) direction predictions, and overall (e.g., ↑ ↑ represents correct prediction of upstream direction, ↓ ↑ an erroneous prediction of downstream for an actually upstream packet). The number in brackets is the RMSE for the specific combination, and overall.

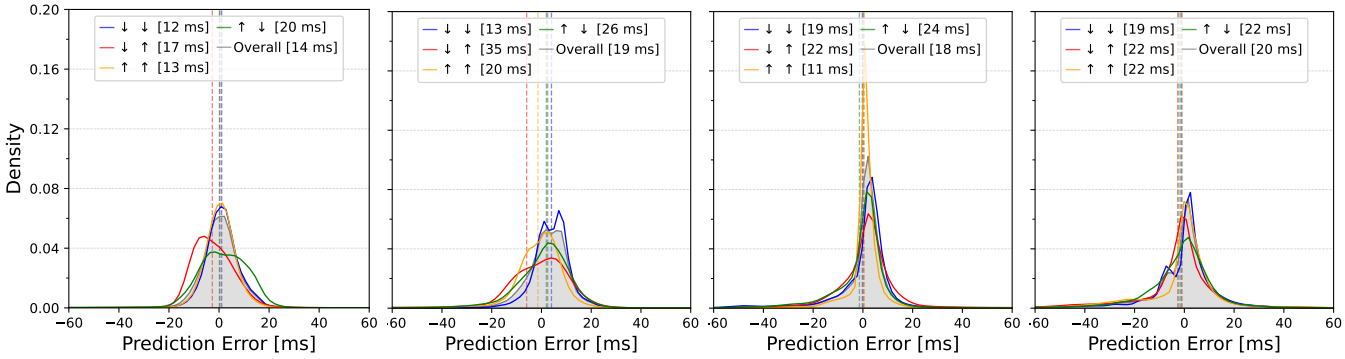
Since the error on DIR has the highest impact on the use of fine-grained (viz. packet-level) prediction results, we analyze the performance of the most "problematic" app when predicting the DIR parameter in more depth. Hence, we dissect the predictions related to Zoom traffic according to the specific activity performed by the user (i.e. ACall, Chat, and VCall) in the *right column* of Fig. 5.

As depicted in Figs. 5a–5b, the worst DIR and PL prediction performance is obtained on VCall, where we observe the lowest G-mean ($\approx$62%) and the greatest RMSE ($\approx$280 B), respectively. An opposite trend can be observed when moving to the IAT prediction (Fig. 5c) which exhibits an RMSE on VCall less than half of that on Chat and 2.5× lower than that on ACall. Also in this case, all DL models always outperform RLP, especially for the prediction of DIR and IAT on ACall (i.e. $\approx -46\%$ G-mean and $\approx +60$ ms RMSE, respectively) and of PL on Chat (i.e. $\approx +94$ B RMSE).

Finally, in examining the influence of the particular DL architecture utilized, it becomes evident that it has a minimal impact on the predictive performance, regardless of whether we consider the entire traffic generated by a given application or analyze the traffic based on different activities[10].

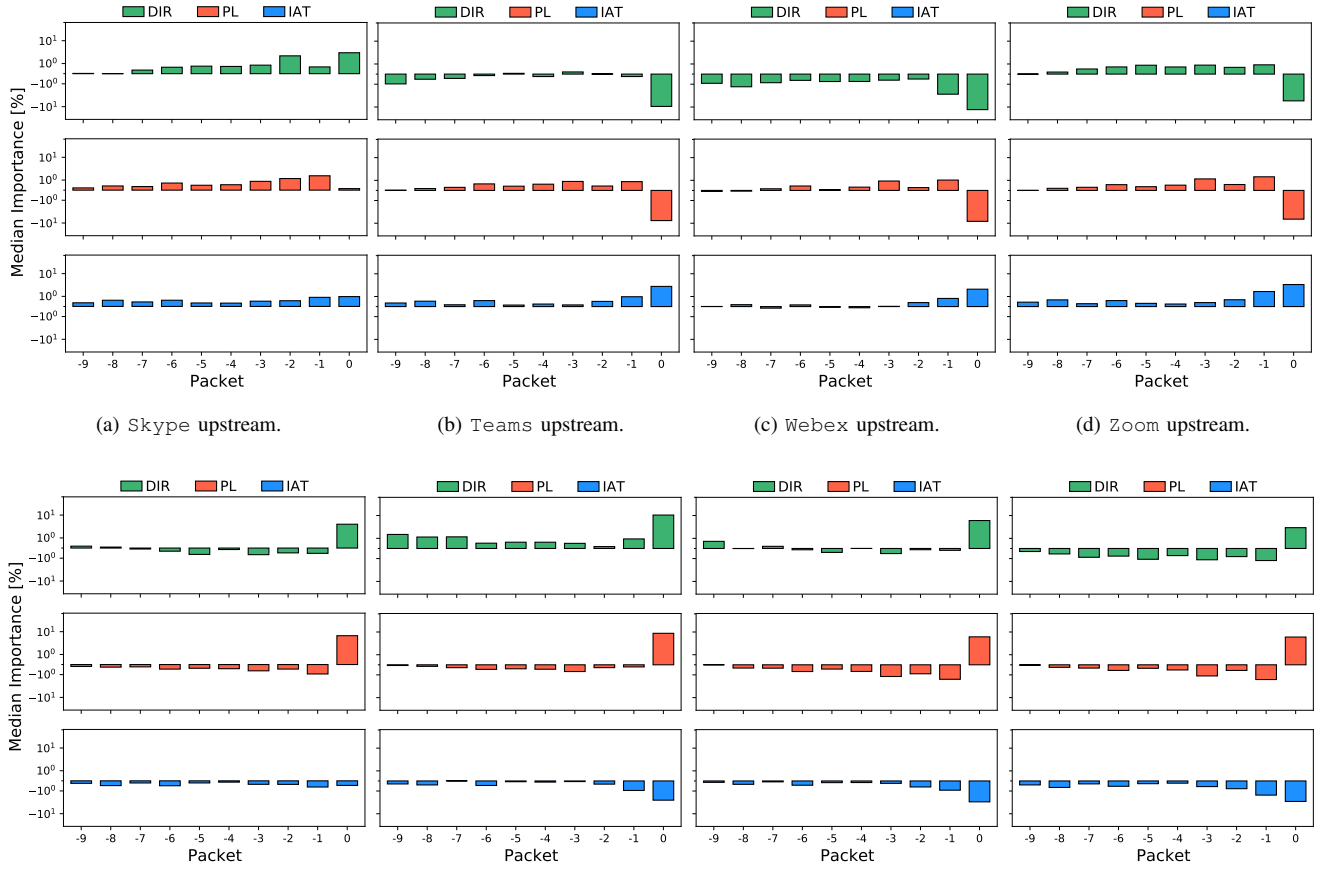
Despite the comparable performance, the complexity of the considered architectures varies significantly. We quantify it with the number of trainable parameters: more trainable

[10]Similar considerations can be drawn also for Skype, Teams, and Webex whose per-activity performance figures are not shown for the sake of brevity.



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

Figure 8: Probability density of prediction error on IAT. The probabilities are conditioned to different combinations of (predicted, true) direction predictions, and overall (e.g., ↑ ↑ represents correct prediction of upstream direction, ↓ ↑ an erroneous prediction of downstream for an actually upstream packet). The number in brackets is the RMSE for the specific combination, and overall.



(a) Skype upstream. (b) Teams upstream. (c) Webex upstream. (d) Zoom upstream.

(e) Skype downstream. (f) Teams downstream. (g) Webex downstream. (h) Zoom downstream.

Figure 9: Median importance (in log-scale) of each packet parameter (i.e., DIR, PL, and IAT) on the prediction of DIR for Skype, Teams, Webex, and Zoom. The x-axis reports the packet index in chronological order, with 0 for the last observed packet, −1 the previous one, and so forth. Each app is separately analyzed according to the (true) direction of packets, as upstream ((a), (b), (c), and (d)) and downstream ((e), (f), (g), and (h)).

parameters result in a longer training time. CNN is the least complex architecture with 27.5 K parameters, while GRU, LSTM, and SeriesNet are significantly more complex and have +96.1 K (GRU), +135.3 K (LSTM), and +502.3 K (SeriesNet) more trainable parameters.

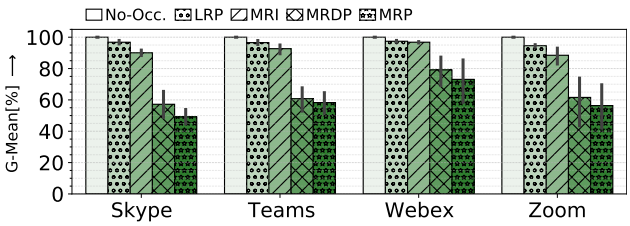


Figure 10: Results obtained without occlusion (viz. *No-Occ.*) are compared with those obtained by occluding (a) the Least Recent Packet (viz. *LRP*), (b) the Most Recent IAT (viz. *MRI*), (c) the Most Recent DIR and PL (viz. *MRDP*), and (d) the Most Recent Packet (viz. *MRP*) as input to the model. The results refer to the samples whose DIR was correctly predicted by the CNN model (ALL variant) without occlusion. Memory size is set to $W = 10$. The arrow close to the y-axis shows the desired trend. Results are in the form $avg. \pm std.$ obtained over 5-folds.

Take-home: *Training a single model on the entire traffic of CC apps is sufficient, as there is no significant benefit in training specialized models for each individual app. This finding carries significant practical implications as it eliminates the need for training, deploying, and managing multiple models. Additionally, when examining the performance of the different apps, we observe that Zoom poses the hardest challenges for prediction, particularly in terms of PL and DIR, whereas the other apps demonstrate better and more consistent performance. Among the Zoom-related activities, VCall proves to be the most difficult to predict still in terms of PL and DIR. Finally, despite the similar performance of the considered DL architectures, they exhibit considerable differences in terms of computational complexity, with CNN being the least complex. Consequently, in the subsequent analyses, we focus on the CNN model trained on the entire traffic (namely, the ALL variant), as it represents the best trade-off between complexity and prediction performance.*

B. Deepening Traffic Prediction Performance

In the following, we delve into the prediction performance by evaluating: (i) how well traffic parameters can be estimated during the initial biflow lifetime (via per-packet-index performance); (ii) how much correct/wrong DIR estimates affect PL and IAT predictions (via the conditional distributions of the prediction errors).

1) How does prediction performance vary along a biflow?

We aim to evaluate whether (and how) prediction performance varies along a biflow. Hence, we provide a *per-packet-index performance* analysis, where the evaluated metrics (i.e. G-mean for DIR and RMSE for PL and IAT) are computed considering packets at the same position in biflows, namely sharing the same index or falling within the same interval of indexes. Specifically, we focus on the head of the biflows (i.e. the first 128 packets), showing aggregated performance each 2 packets until the 32^{nd} and each 8 packets for the remaining segment.

Fig. 6 reports the results for Teams and Webex (Skype and Zoom show similar patterns, thus, they are omitted for brevity). For both apps, the analysis witnesses that the packets from 2 to 32 are harder to be predicted in terms of PL and IAT. Conversely, predicting their direction is easier. In fact, the G-mean resulting from the prediction of the DIR of the packets from 2 to 32 (G-mean_{2-32}) varies from 79% (Teams) up to 80% (Webex) and is higher than that observed on the remaining part of the biflow ($\text{G-mean}_{33-\text{END}}$), on average: 78% (Teams) and 70% (Webex). On the contrary, the prediction error incurred for PL and IAT on the initial part of the biflows (RMSE_{2-32}) is higher than the error on the remaining part ($\text{RMSE}_{33-\text{END}}$), on average: up to +220B for Webex and +42ms for Teams on PL and IAT, respectively.

Take-home: *Consistent discrepancy is found between the lower prediction capabilities achieved for the beginning part and the higher capabilities attained for the rest of the biflows, although being less evident in some cases (e.g., DIR for Teams).*

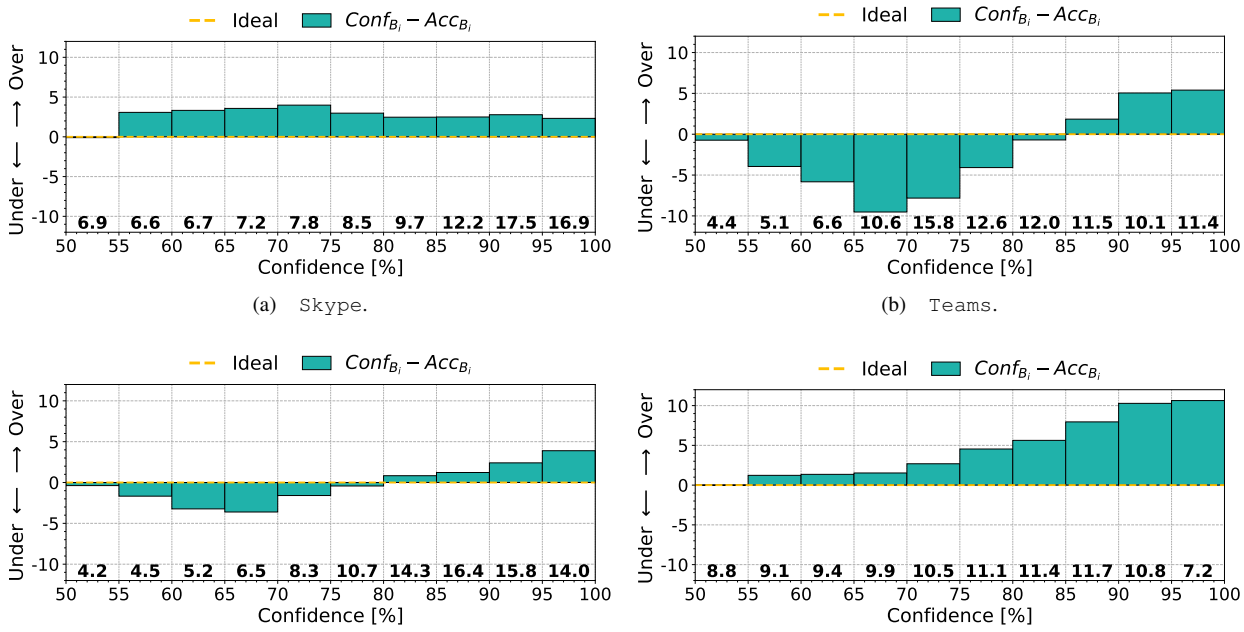
2) What kind of errors do the models make?

We aim at characterizing the errors the prediction models make. Figs. 7 and 8 report the prediction error $(\hat{x}^{n+1}_{(\cdot)} - x^{n+1}_{(\cdot)})$ associated with PL and IAT, respectively, for all the apps (grey curves). By construction, negative values in the distribution are related to under-estimation (i.e. the prediction is *lower* than the actual value) for either PL or IAT, while positive ones report over-estimation (i.e. the prediction is *higher* than the actual value).

Overall, the errors span almost the whole theoretical $(-1470, 1470)$ B range for PL, while they lie in the range $(-198, 196)$ ms for IAT. However, errors small in magnitude are extremely more frequent than errors with high magnitude. In all the cases, the bias of the distribution is placed around 0 (at a distance always ≤ 9 B and ≤ 2 ms for PL and IAT, respectively). For PL prediction, the RMSE varies between 157 B and 277 B (for Skype and Zoom, respectively). For IAT prediction, the RMSE varies between 14 ms and 20 ms (for Skype and Zoom, respectively).

In order to analyze the relationship between the predictions on DIR and the other two metrics (PL and IAT), Figs. 7 and 8 report also the breakdown of the error distribution conditioned on true and predicted DIR, i.e. $(\hat{x}^{n+1}_{(\cdot)} - x^{n+1}_{(\cdot)})|\hat{x}_{\text{dir}}, x_{\text{dir}}$. Observing the impact of correct/wrong predictions of DIR on the error incurred for the other two metrics, it is evident that when the models make mistakes in predicting DIR, higher errors are recorded for both PL and IAT regardless of the specific app.

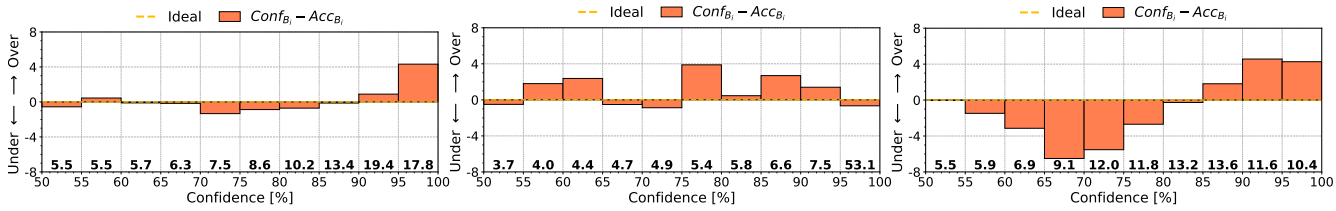
Specifically, the observed behavior also depends on the app. Focusing on PL (Fig. 7), while for Skype wrongly predicting DIR has no remarkable impact on PL, the same does not apply to the other apps, where over-estimation of PL is observed, on average. In fact, the observed bias equals



(a) Skype. (b) Teams.

(c) Webex. (d) Zoom.

Figure 11: Reliability diagrams related to DIR-prediction task for Skype (a), Teams (b), Webex (c), and Zoom (d) when using the ALL model. As the DIR-prediction task represents a binary classification problem, the confidence interval varies in the range [50%, 100%]. Confidence is divided into 10 bins. The number at the bottom of each bar reports the percentage of samples within the corresponding bin.



(a) ACall. (b) Chat. (c) VCall.

Figure 12: Reliability diagrams related to DIR-prediction task for ACall (a), Chat (b), VCall (c) when using the ALL model. As the DIR-prediction task represents a binary classification problem, the confidence interval varies in the range [0.5, 1]. Confidence is divided into 10 bins. The number at the bottom of each bar reports the percentage of samples within the corresponding bin.

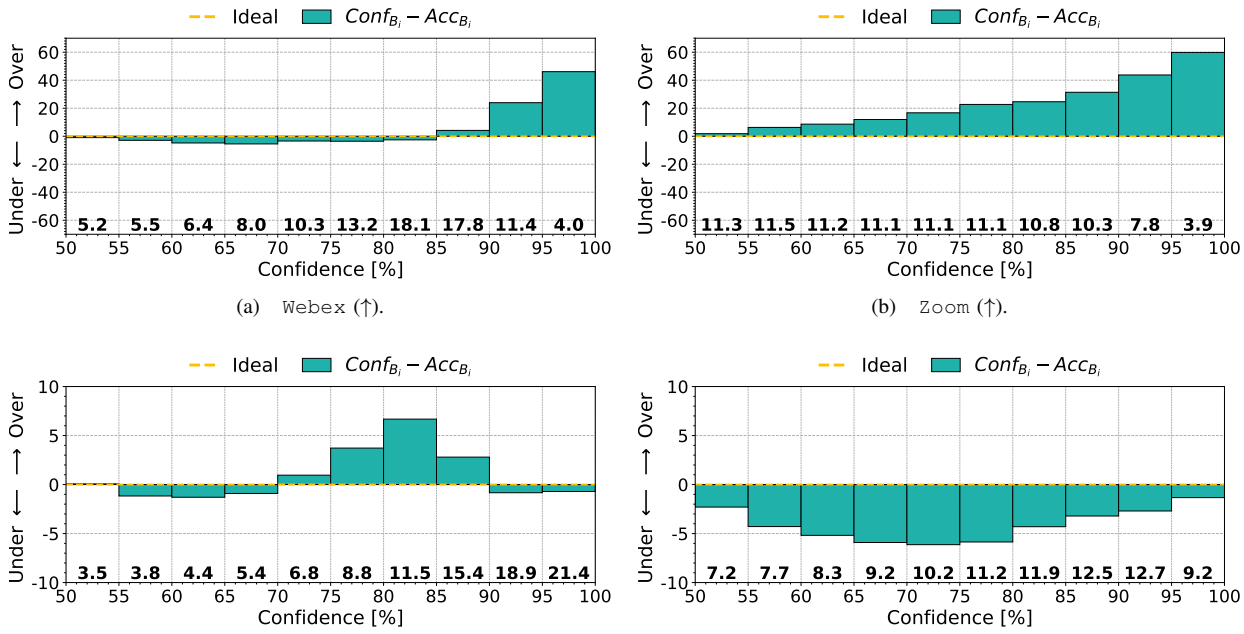
to 119 B (resp. 114 B) for Teams (resp. Zoom) when $(\hat{x}_{\text{dir}}, x_{\text{dir}}) = (\uparrow, \downarrow)$ (resp. $(\hat{x}_{\text{dir}}, x_{\text{dir}}) = (\downarrow, \uparrow)$). For Webex the bias is 167 B and 61 B when $(\hat{x}_{\text{dir}}, x_{\text{dir}}) = (\downarrow, \uparrow)$ and $(\hat{x}_{\text{dir}}, x_{\text{dir}}) = (\uparrow, \downarrow)$, respectively.

On the other hand, concerning IAT (Fig. 8) remarkable under-estimation of the parameter is observed when $(\hat{x}_{\text{dir}}, x_{\text{dir}}) = (\downarrow, \uparrow)$ for Skype and Teams (-3 ms and -6 ms, respectively). Interestingly, for Teams if $x_{\text{dir}} = \downarrow$, when $\hat{x}_{\text{dir}} = \downarrow$ the incurred bias is larger than when $\hat{x}_{\text{dir}} = \uparrow$.

Overall, PL under-estimation is more frequent than over-estimation for Zoom, Webex, and Teams, while the opposite holds for Skype. This tendency is confirmed when restricting the observation to cases where DIR is not mistaken. On the other hand, when DIR is wrongly predicted, an inversion in this can take place. On the contrary, the probability of over-estimating IAT is higher than under-estimating it for all the apps.

Errors on PL (both over- and under-estimation) larger than 200 B appear in less 25% of the cases for all the apps (less than 10% of the cases for Skype). Similarly, errors on IAT with magnitude larger than 10 ms appear for less than 30% of the cases.

Take-home: *The occurrence of under-estimating PL is more common than over-estimating it for Zoom, Webex, and Teams, while the opposite trend is observed for Skype. This pattern remains consistent when considering only cases where DIR is correctly predicted. However, when DIR is incorrectly predicted, there is a reversal of this tendency. Finally, for all the apps, the probability of over-estimating IAT is higher than under-estimating it.*



(a) Webex (↑). (b) Zoom (↑).

(c) Webex (↓). (d) Zoom (↓).

Figure 13: Reliability diagrams related to DIR-prediction task for Webex (a, c) and Zoom (b,d) when using the ALL model and according to the correct DIR prediction (↑=upstream, ↓=downstream). As the DIR-prediction task represents a binary classification problem, the confidence interval varies in the range [50%, 100%]. Confidence is divided into 10 bins. The number under each bar reports the percentage of samples within the corresponding bin.

C. Interpreting DIR prediction via DEEP SHAP: how do inputs affect DIR prediction?

Herein, relying on DEEP SHAP, we examine the relative influence of the three traffic parameters—i.e., DIR, PL, and IAT—extracted from the last W observed packets of each biflow on the prediction of DIR.

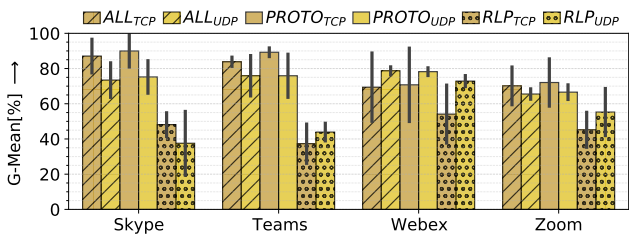
As a result, for each traffic parameter, Fig. 9 depicts the median importance values across the last 10 packets (since $W = 10$) of each biflow that are used to feed the model to obtain the prediction of the next packet. More in detail, the figure provides a detailed breakdown of Skype, Teams, Zoom, and Webex for each packet direction (i.e. upstream ↑ and downstream ↓). As can be seen, for a given traffic parameter, the importance associated with packets in the sequence varies according to the predicted direction.

In particular, for almost all apps, all observed input sequences positively contribute to the prediction when the model correctly predicts the *upstream* direction (↑, see Figs. 9a–9d), with the most recent packets usually having greater importance. Interestingly, for Teams and Webex, unlike PL and IAT, the DIR of the last 10 observed packets always has a negative effect on the prediction of the *upstream* direction. At the same time, when we examine DIR and PL of the last observed packet, we find that it has no effect or even works against a proper direction prediction (negative score). This holds especially for Zoom and Teams, where the last observed packet has a non-negligible negative importance—which also corresponds to the highest values in magnitude—suggesting that this generally leads the model to predict the *downstream* direction instead of the correct *upstream* one.

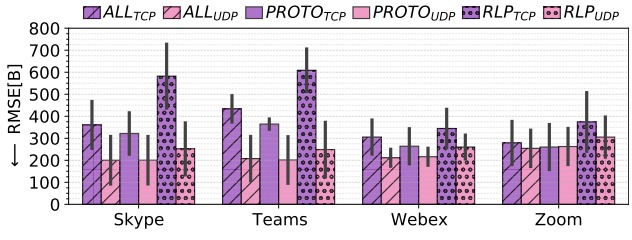
Conversely, moving to the prediction of *downstream* direction (↓, see Figs. 9h–9g), we notice an opposite trend, where the DIR and PL of the last packet have a positive impact on the prediction. Furthermore, their importance (in terms of magnitude) is significantly greater than that of the other packets. Based on this observation, we can infer that the DIR and PL of the last packet lead the model to accurately predict the *downstream* direction. Lastly, it is worth noting that, unlike all other apps, in the case of Teams, the DIR of all observed packets has a positive impact on the prediction of the downstream direction of the next packet.

We conducted an *occlusion analysis* to quantitatively assess the above findings. Occlusion analysis is a perturbation technique that examines the effect of occluding certain inputs on the output of DNNs [44]. Accordingly, we evaluated how the performance of CNN (variant ALL) varies by occluding different traffic parameters used as model inputs on samples whose direction is correctly predicted. To this end Fig. 10 depicts the performance obtained by occluding: (i) the Least Recent Packet (viz. *LRP*), (ii) the Most Recent IAT (viz. *MRI*), (iii) the Most Recent DIR and PL (viz. *MRDP*), and (iv) the Most Recent Packet (viz. *MRP*).[11] These occlusion

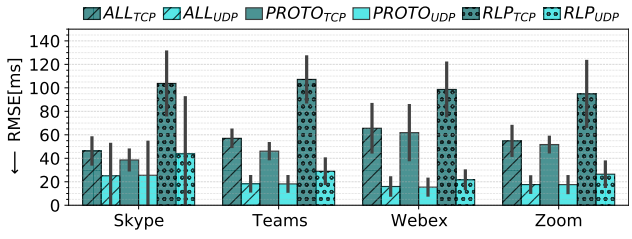
[11]The occluded traffic parameter has been replaced with the corresponding padding value, i.e. 0 for PL and IAT and 0.5 for DIR.



(a) DIR.



(b) PL.



(c) IAT.

Figure 14: Prediction performance of CNN and RLP on DIR, PL, and IAT w.r.t. UDP and TCP protocols. For CNN, the results obtained by utilizing the ALL training strategy are compared with those achieved by using the PROTO strategy. Results refer to the prediction of traffic generated by all apps with a memory $W = 10$. Results are in the form $avg. \pm std.$ obtained over 5-folds.

choices reflect the importance of the input parameters shown in Fig. 9.

We note that occluding the parameters of the least recent packet results in a slight worsening of performance (i.e., $\leq 5.4\%$ of G-mean). In contrast, partially or completely occluding the parameters of the most recent packet results in a significant worsening (i.e. $\leq 44\%$ of G-mean). Further focusing on the traffic parameters of the most recent packet, we note that simultaneously occluding DIR and PL results in a greater performance loss (i.e. $\leq 43\%$ of G-mean for Skype) than occluding the sole IAT (i.e. $\leq 12\%$ of G-mean for Zoom). The aforementioned findings are consistent with that shown in Fig. 9, where it can be seen that DIR and PL of the most recent packet have a similar impact on the predicted DIR, and their importance, in magnitude, is considerably greater than that of IAT. Conversely, the features of the least recent packet have a minor impact on the model outcome.

Take-home: *The prediction of the direction of the next packet, based on the observation of the last* 10 *most recent packets, is mostly influenced by the* $3 \sim 4$ *more recent packets observed. Moreover, the direction and transport-layer payload length of the last observed packet can be either highly beneficial (downstream case) or detrimental (upstream case) to properly predict the direction of the next packet.*

D. Calibration analysis: how reliable is the prediction of DIR?

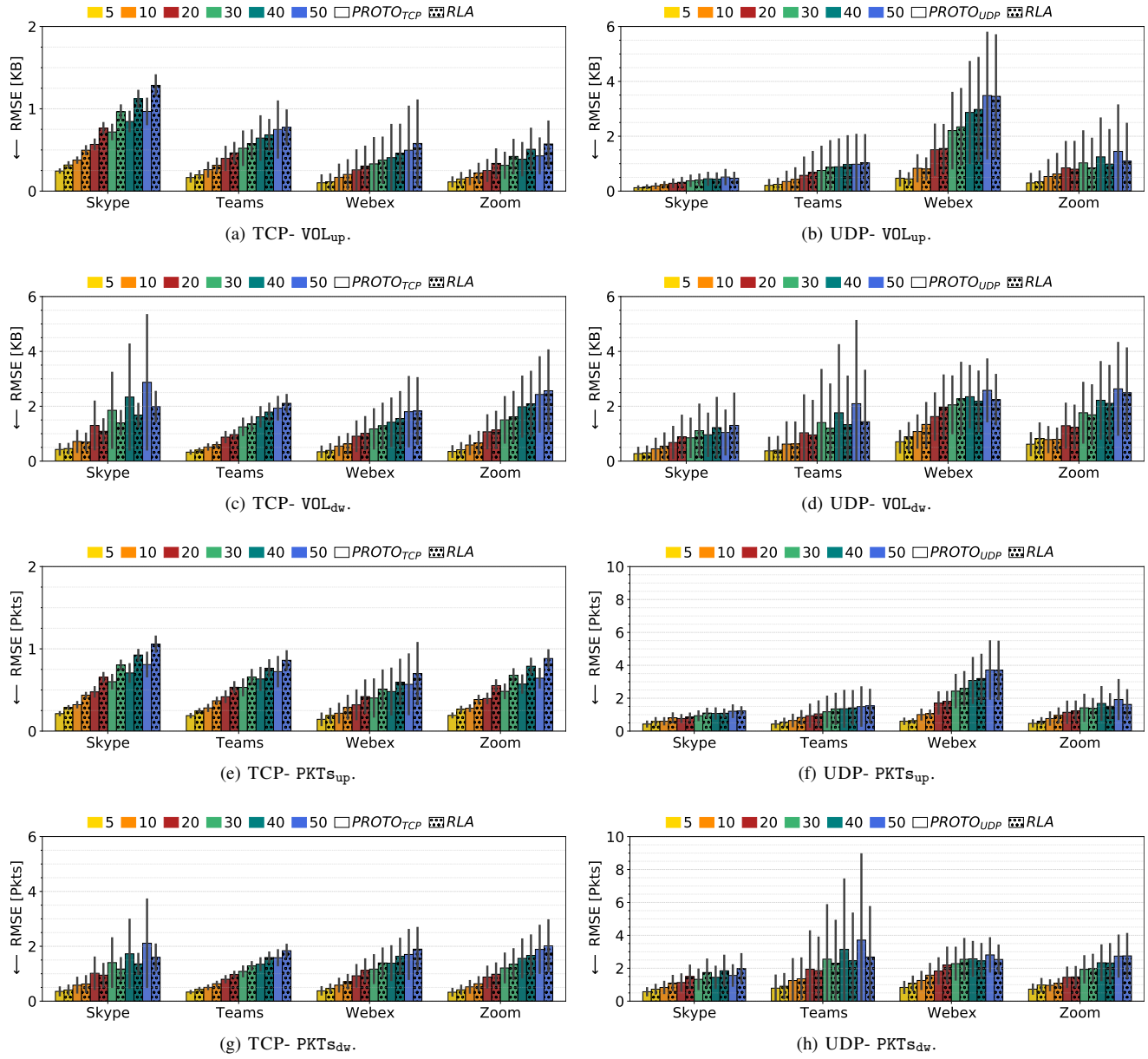
We now focus on assessing the reliability of the model on the prediction of DIR in terms of the calibration metrics defined in Sec. III.D, namely the (*i*) ECE, (*ii*) MCE and (*iii*) CW − ECE. To this end, in Tabs. 3 and 4, we report the above values (in percentage form) obtained by considering the app and the activity performed by the user, respectively.

Focusing on the app, we note that in the case of Webex, the model has the best calibration when used to predict the direction of the next packet with respect to all the metrics. This result is in agreement with Fig. 5a in which it is shown that the model achieves the best performance in the case of Webex (i.e. $\approx 80\%$ of G-mean). Conversely, the model is less calibrated, particularly in the case of Teams for which the MCE is significantly higher (up to $6\times$). Also, it is interesting to note that although the prediction performance associated with Teams is significantly better than that obtained for Zoom (see Fig. 5), in the former case, the model is less calibrated, especially in terms of MCE.

Finally, moving to the activity, we note that while for ACall and Chat the model has a good calibration (except for Chat in terms of MCE), this does not hold for VCall to which corresponds an ECE/CW − ECE and an MCE that are $2\times$ and $7\times$ higher, respectively.

Then, to visualize in detail how $\Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\}$ varies with $\hat{p}$, in Figs. 11–12, we show the accuracy as a function of the confidence by means of a variant of the *reliability diagrams* described in Sec. III. Therein, the *difference* $\hat{p} - \Pr\left\{\hat{x}_{dir}^{n+1} = x_{dir}^{n+1} | \hat{p}\right\}$ is reported on the y-axis: a perfectly-calibrated classifier implies a null difference on all the bins, whereas an over-confident (resp. under-confident) model is associated to a positive (resp. negative) difference.

Looking at the results obtained per app, we note that the samples are uniformly distributed across the bins. Moreover, as highlighted in Figs. 11a and 11c, for Skype and Webex we observe a slight overall over-confidence and a near-ideal behavior, respectively. In contrast, referring to Figs. 11b and 11d, in the case of Teams and Zoom we observe a more pronounced over- and under-confidence that varies with confidence. Specifically, for Teams, we observe under-confident (resp. over-confident) behavior, especially when the confidence ranges in $60 - 75\%$ ($90 - 100\%$). On the other hand, for Zoom, we observe a trend that indicates that the model is more over-confident the more confidence with which it makes predictions. This means that increased confidence does not correspond to increased accuracy. To



(a) TCP- $\texttt{VOL}_{\texttt{up}}$. (b) UDP- $\texttt{VOL}_{\texttt{up}}$.

(c) TCP- $\texttt{VOL}_{\texttt{dw}}$. (d) UDP- $\texttt{VOL}_{\texttt{dw}}$.

(e) TCP- $\texttt{PKTs}_{\texttt{up}}$. (f) UDP- $\texttt{PKTs}_{\texttt{up}}$.

(g) TCP- $\texttt{PKTs}_{\texttt{dw}}$. (h) UDP- $\texttt{PKTs}_{\texttt{dw}}$.

Figure 15: Prediction performance (RMSE) of CNN trained on TCP (*left column*) and UDP (*right column*) traffic for $\texttt{VOL}_{\texttt{up}}$ (a, b), $\texttt{VOL}_{\texttt{dw}}$ (c, d), $\texttt{PKTs}_{\texttt{up}}$ (e, f), and $\texttt{PKTs}_{\texttt{dw}}$ (g, h) as $\Delta \in \{5, 10, 20, 30, 40, 50\}$ ms. CNN performance is compared against RLA predictor. Results refer to the prediction of traffic generated by all apps with a memory $W = 10$.

deepen this finding, in Figs. 13, we present a similar analysis for Webex and Zoom, distinguishing based on the true DIR.

As shown, the calibration of the analyzed apps varies by direction. Specifically, we observe a slight over-confidence for the downstream direction (i.e., $\downarrow$) for Webex and a slight under-confidence for Zoom. This especially holds for samples whose confidence levels are in the range $80 - 85\%$ range for Webex and in the range $60 - 80\%$ for Zoom. This corresponds to a gap of up to 7% between the expected confidence and the corresponding accuracy. Conversely, in the upstream direction (i.e., $\uparrow$), both Webex and Zoom show a trend where the predictor becomes more over-confident as its confidence level increases. This trend is particularly noticeable when the confidence level is $\geq 90\%$ for Webex, and $\geq 5\%$ for Zoom. In such cases, the gap between the expected confidence level and the accuracy is $\geq 10\%$. It is worth noting that while this behavior affects only 15% of Webex samples, it affects 66% of Zoom samples. Based on these findings, we deduce that the poor performance in predicting DIR for Zoom (cf. Fig. 5a) is mainly due to inaccurate prediction of the upstream direction. This is further supported by the fact that the recall rate, not shown

for brevity, is only about $\approx 55\%$ for the upstream direction, while it is $\approx 80\%$ for the downstream direction.

Moving to the calibration results related to the specific activity, for ACall and VCall (see Figs. 12a and 12c), the related prediction tasks present the best and worst calibration, respectively. We note that while for ACall the confidence of each bin is very close to the corresponding accuracy (close-to-zero difference), for VCall the model is particularly under-confident and over-confident with respect to the samples falling in bins $2-6$ and $8-10$, respectively. Finally, for Chat (see Fig. 12b), we note that most of the samples (53%) fall in the last bin. This indicates an optimistic behavior of the model in the presence of Chat traffic that does not affect its calibration.

Take-home: *Generally, the predictor exhibits good calibration, i.e., the confidence level associated with the predictions reflects its reliability. The level of miscalibration is typically* $< 2\%$*, with the highest level of over-confidence observed in the case of* Zoom*, where it still remains below* 10%*. In addition, for* Zoom*, when analyzing the calibration w.r.t. the upstream direction we notice that as the confidence level increases, the predictor becomes more over-confident. This results in a gap* $\geq 10\%$ *between the expected confidence level and the actual accuracy, which affects* $\approx 66\%$ *of its samples. This is related to the poor performance achieved for* DIR *prediction shown in Sec. V.A.*

E. Do we need a dedicated model for TCP *and* UDP *protocols?*

We have previously shown in Sec. V.A that the CNN architecture achieves the best trade-off between performance and computational complexity when trained on all apps (ALL strategy). Additionally, Sec. IV revealed that CC apps utilize both TCP and UDP protocols in their operations. Indeed, while they tend to establish numerous TCP connections (viz. biflows), the majority of data (in terms of both packets and volume) are transmitted through UDP biflows. This is generally because TCP is mainly used for various control and data management purposes, while UDP handles real-time communications related to audio or video traffic in the case of CC apps, which require low latency and fast transmission (cf. Fig. 3). Accordingly, here we investigate the benefits of having a separate model for each transport protocol (ref. PROTO strategy) against employing a single model to handle both TCP and UDP traffic (ref. ALL strategy). In Fig. 14, we present the performance results obtained by these models for each app, distinguishing between specific protocols and training strategies. For the sake of completeness, we also include the breakdown of performance achieved by the RLP predictor. Overall, our results show that the effectiveness of the two strategies depends on the app and the protocol: while for UDP traffic, the performance is unaffected by the training strategy, for TCP traffic training a specific model for the protocol results in a considerable performance improvement for all prediction tasks. In particular, the improvement on DIR prediction is up to $+5\%$ of G-mean, and up to ≈ -70 B and ≈ -11 ms of RMSE for PL and IAT, respectively.

By breaking results down on specific apps and protocols, we observe that prediction of DIR performs better on TCP traffic for nearly all apps except Webex. However, this trend reverses when predicting PL and IAT, where models for TCP traffic exhibit poorer performance. Furthermore, our results reveal that CNNs consistently outperform the RLP predictor for all predicted parameters, irrespective of the protocol and training strategy. The difference is particularly pronounced for TCP traffic generated by Skype and Teams, with CNNs outperforming the RLP predictor by approximately $\approx +40\%$ of G-mean for DIR and ≈ -240 B and ≈ -60 ms of RMSE for PL and IAT, respectively. Similar trends are also observed for Webex and Zoom, where the difference between CNNs and the RLP predictor is up to $+27\%$ for DIR and up to -120 B and -40 ms on PL and IAT, respectively. When analyzing UDP traffic, the performance gap between CNNs and the RLP predictor is still evident but less prominent. For Skype, CNN models outperform the RLP predictor by $\approx +40\%$ for DIR, ≈ -50 B for PL, and ≈ -20 ms for IAT.

Take-home: *The effectiveness of a specific training strategy (i.e., ALL and PROTO) varies depending on the transport-level protocol: for all apps, training a dedicated model for* TCP *traffic (i.e., the PROTO strategy) generally leads to notable improvements in performance across all prediction tasks. However, the same approach does not show similar benefits for* UDP *traffic, as the training strategy does not have a noticeable impact. Additionally, the prediction accuracy for DIR is generally higher for* TCP *compared to* UDP *traffic, while the reverse trend is noticed for PL and IAT predictions.*

F. Is Packet-Level Prediction suitable to Predict Traffic Aggregates?

Herein, we analyze the suitability of packet-level predictors to forecast aggregate traffic (i.e., the number of packets and the traffic volume) in a given time interval Δ. Therefore, applying the methodology outlined in Sec. III.F, we compare the performance of a CNN trained leveraging the PROTO strategy (i.e., two dedicated models designed for TCP and UDP traffic) against a simple RLA predictor.

Fig. 15 illustrates the performance—in terms of RMSE—for each protocol when predicting the number of packets (viz. $\texttt{PKTs}_*$) and the traffic volume (viz. $\texttt{VOL}_*$), for both upstream and downstream directions. The performance is reported for different time horizons, namely $\Delta \in \{5, 10, 20, 30, 40, 50\}$ ms.

At a high level, we observe that the error increases with Δ. The greater unpredictability of traffic over extended time intervals could be attributed to its greater variance and uncertainty [45] and also to the growth of prediction errors accumulated through the recursive procedure adopted. This variability is particularly evident in VoIP traffic, encompass-

ing both audio and video, as it tends to produce *micro-bursts* that are difficult to predict——i.e., a notable surge in data transmission within a brief duration, followed by relatively quieter periods.

We also note that this error varies depending on three factors: (i) traffic direction, (ii) protocol, and (iii) app considered.

Taking a closer look, we observe that the errors in the upstream direction increase gradually and tend to stay within a relatively narrow range (averaging around ≈ 1.3 KB and ≈ 1 packet) for most applications and protocols, except for Webex on UDP traffic. In contrast, the downstream direction shows a more abrupt and highly-variable change, with errors reaching higher average levels (up to $\approx$ 3 KB and $\approx$ 4 packets).

When analyzing the performance on TCP traffic (see *left* column of Fig. 15), CNN proves to be a superior choice for predicting all features compared to RLA, regardless of the prediction interval Δ. This advantage is particularly evident when considering the traffic direction for Skype, Zoom, and Teams. Specifically, for Skype (resp. Zoom), the mean RMSE of CNN is at least $\approx 23\%$ (resp. $\approx 24\%$) and $\approx 23\%$ (resp. $\approx 27\%$) lower for $\texttt{VOL}_{\texttt{up}}$ and $\texttt{PKTs}_{\texttt{up}}$, respectively. Similarly, for Teams, the difference is at least 8% and 27% lower for $\texttt{VOL}_{\texttt{dw}}$ and $\texttt{PKTs}_{\texttt{dw}}$, respectively.

Interestingly, the only scenario where this trend is reversed is observed on Skype in the downstream direction when $\Delta \geq 30$ ms (see Figs. 15c and 15g). In such cases, the RMSE of the CNN on $\texttt{VOL}_{\texttt{dw}}$ and $\texttt{PKTs}_{\texttt{dw}}$ is up to $+45\%$ and $+32\%$ higher than that of RLA.

However, when moving on UDP traffic (see *right* column of Fig. 15), no consistently predominant approach can be identified. Nonetheless, the results obtained indicate that CNN is the most favorable option for the majority of scenarios, particularly when it comes to predicting the number of packets.

Specifically, compared to RLA, we find that CNN provides a more accurate packet number estimation for applications such as Skype, Teams, and Webex in the upstream direction (Fig. 15f), as well as for Skype and Zoom in the opposite direction (Fig. 15h). However, it is important to note that for certain applications (e.g., Teams and Zoom in the downstream and upstream directions, respectively), while CNN outperforms the RLA predictor on short prediction intervals (i.e., $\Delta \leq 20$ ms), the trend reverses on longer ones.

Lastly, when examining the results on traffic volumes (see Figs. 15b and 15d), we notice a more diverse pattern between CNN and RLA methods. In the case of Teams (Skype), the CNN consistently surpasses the RLA approach exhibiting a reduced error ranging from -4% (-10%) to -18% (-24%) on the upstream (downstream) direction.

In the other scenarios, the CNN remains the superior option for predicting aggregate traffic within relatively short time intervals (i.e., $\Delta \leq 30$ ms). However, CNN is less effective over longer time horizons. Nonetheless, it is worth noting that having a more accurate predictor over short time intervals facilitates constant monitoring of network performance and real-time optimization. These aspects are essential when dealing with CC apps, especially when these are used by the user to perform VoIP calls (i.e., audio and video), which require a good level of QoS and at the same time tend to cause intense activity on the network.

Take-home: *The proposed approach outperforms RLA in most scenarios, especially when used to predict aggregate traffic over relatively short time intervals (e.g., $\Delta \leq 30$ ms). Within this range, the proposed approach yields an average error of up to ≈ 1.3 KB (≈ 1.6 KB) and 1 (2) packet(s) on* TCP *(*UDP*) traffic protocol. Overall, the error is limited to $\approx$ 3 KB ($\approx$ 3.5 KB) and 2 (4) packets on* TCP *(*UDP*). These findings are significant as they indicate that the proposed approach represents a first step towards the development of more responsive networks. Indeed, accurate short-term predictions facilitate real-time optimization of network capacity, enabling effective bandwidth adaptation and resource allocation as needed. This has the potential to enhance the network's performance and responsiveness in dynamic and ever-changing traffic conditions.*

In particular, we analyzed the calibration breaking down in the correctly predicted direction.

VI. Discussion

The experimental analysis highlights some interesting takeaways, *summarized and discussed in the broader spectrum of existing literature*. First, our results have shown that a CNN trained on the whole traffic of all apps represents a better trade-off between prediction performance and complexity compared to per-app models. Architecture-wise, the CNN results align with those reported in [9], which however refer to a larger and different set of mobile apps. Conversely, regarding the training strategy, negligible performance gains originating from the design of per-app predictors have been already found in the context of model-based predictors (i.e. multimodal Markov-chains [46]) and confirmed herein. Then, focusing on the different transport-layer protocols used by the apps under analysis, we found that training a dedicated model for TCP traffic generally leads to notable improvements in traffic-prediction performance across all the tasks considered.

Going deeper, the results show a variety of behaviors for the different apps, with Zoom being the hardest to predict, especially regarding the direction (65% G-mean) and the payload length (260 B RMSE) of the next packet. Such behavior was preliminarily observed also in [9], where the predictability of apps including video streaming (i.e. belonging to the Mirage-VIDEO dataset [9]) was shown to be lower with respect to non-video apps (i.e. belonging to the Mirage-19 dataset [43]). Within the former set, the performance of Zoom was worse also in comparison to other apps accounting for video dissemination (e.g., Netflix

and PrimeVideo achieving $\geq 90\%$ G-mean on prediction of direction). Conversely, regarding the *activities* that can be performed by these apps, video-call is the hardest one for what concerns the prediction of the direction and the payload length, and the easiest concerning timing prediction. Interestingly, video-call is the most identifiable activity when performing traffic classification, see e.g. [42].

Overall, the predictor results very well-calibrated when predicting `DIR`. Therefore the confidence reported with the prediction is actually representative of its reliability (generally $< 2\%$ miscalibration, with the worst-case of $< 10\%$ overconfidence for `Zoom` only). The fine-grained interpretation of the DL models highlights that using a limited memory of previously observed packets, the predictions are mainly influenced by the most recent packets. A dominating short-term evolution behavior of biflows was also preliminarily observed thanks to the proficient use of higher-order Markov Chains with a very limited memory as predictors [22]. Furthermore, in most cases, the last observed packet can be either very informative (for `Skype` and `Zoom` downstream) or confounding for the model (for `Zoom` upstream). These results point to possible applications and direction-specific improvements, following a per-sample inspection of misleading inputs.

Finally, taking advantage of the benefits offered by the fine granularity prediction approach (i.e. at packet level), we made a first attempt to solve coarser granularity prediction problems (e.g., to predict traffic volume and number of packets in a given time interval). While different sophisticated design solutions for coarser-granularity prediction exist—e.g. [21]—the flexible-granularity property of our approach represents a novel aspect of this work. Our results show that performance varies depending on several factors such as the transport-layer protocol, traffic direction, and the app considered. The best results are obtained for `Webex` on TCP—with an average error of up to 0.5 KB (1.8 KB) and ≈ 1 (≈ 2) packets in the upstream (downstream) direction—and for `Skype` on UDP—with an average error of up to 0.5 KB (1 KB) and ≈ 1 (≈ 2) packets in the upstream (downstream) direction.

VII. Conclusions and Future Directions

We tackled *packet-level traffic prediction* via Deep Learning architectures (a CNN, a GRU, an LSTM, and a SeriesNet) using the publicly released MIRAGE-COVID-CCMA-2022 dataset as a valuable test bench. We employed XAI approaches (i.e. DEEP SHAP) to contrast the black-box nature of these models and obtain actionable insights on the importance of specific subsets of input data and evaluated the reliability of predicting `DIR` (via a calibration analysis). Third, we have identified the impact of employing different training strategies on the design of predictors to assess the actual need for multiple model training, deployment, and management.

Finally, taking advantage of the benefits offered by the fine granularity prediction approach (i.e., at packet-level), we made a first attempt to solve coarser granularity prediction problems (e.g., to predict traffic volume and number of packets in a given time interval). Our results showed that performance varies depending on several factors such as the transport layer protocol, traffic direction, and the app considered.

Future directions will include: (i) interpretability analysis for payload length and inter-arrival and general use of XAI toward improvement of multitask predictors, (ii) design of a more advanced approach to predict traffic aggregates from packet-level predictions, (iii) lifelong learning to cope with concept drift due to app aging, and (iv) federated-learning exploiting multiple vantage points (also including different stakeholders and institutions).

ACKNOWLEDGMENT

This work is partially supported by the "ADDITIONAL" Project funded by Vietsch Foundation, the Italian Research Program "PON *Ricerca e Innovazione* 2014-2020 (PON R&I) REACT-EU – Asse IV – Azione IV.4", the "Centro Nazionale HPC, Big Data e Quantum Computing – ICSC", and the "RESTART" Project funded by MUR. Also, this study is partially carried out within the "xInternet" project—funded by the Ministero dell'Università e della Ricerca—within the PRIN 2022 program (D.D.104–02/02/2022). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

References

[1] Sandvine, "The Global Internet Phenomena Report COVID-19 Spotlight." Jan 2023.

[2] A. Feldmann, O. Gasser, F. Lichtblau, E. Pujol, I. Poese, C. Dietzel, D. Wagner, M. Wichtlhuber, J. Tapiador, N. Vallina-Rodriguez, O. Hohlfeld, and G. Smaragdakis, "The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic," in *ACM Internet Measurement Conference (IMC)*, 2020, p. 1–18.

[3] M. Candela, V. Luconi, and A. Vecchio, "Impact of the COVID-19 pandemic on the Internet latency: A large-scale study," *Comp. Networks*, vol. 182, 2020.

[4] I. Guarino, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Characterizing and modeling traffic of communication and collaboration apps bloomed with covid-19 outbreak," in *2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI)*, 2021, pp. 400–405.

[5] L. Grote, I. Kunze, C. Sander, and K. Wehrle, "Instant messaging meets video conferencing: Studying the performance of im video calls," in *Proceedings of the Network Traffic Measurement and Analysis Conference (TMA '23)*. IFIP/IEEE, 6 2023.

[6] L. Velasco, R. Casellas, S. Llana, L. Gifre, R. Martínez, R. Vilalta, R. Muñoz, and M. Ruiz, "A control and management architecture supporting autonomic NFV services," *Photonic Network Communications*, vol. 37, no. 1, pp. 24–37, 2019.

[7] W. Jiang, "Cellular traffic prediction with machine learning: A survey," *Expert Systems with Applications*, vol. 201, p. 117163, 2022.

[8] G. O. Ferreira, C. Ravazzi, F. Dabbene, G. C. Calafiore, and M. Fiore, "Forecasting network traffic: A survey and tutorial with open-source comparative evaluation," *IEEE Access*, 2023.

[9] A. Montieri, G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapè, "Packet-level prediction of mobile-app traffic using multitask Deep Learning," *Computer Networks*, vol. 200, p. 108529, 2021.

[10] I. Guarino, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Fine-grained traffic prediction of communication-and-collaboration apps via deep-learning: a first look at explainability," in *IEEE International Conference on Communications (ICC)*, 2023, pp. 1–6.

[11] A. Dainotti, A. Pescapé, P. Salvo Rossi, F. Palmieri, and G. Ventre, "Internet traffic modeling by means of Hidden Markov Models," *Elsevier Computer Networks*, vol. 52, no. 14, pp. 2645–2662, 2008.

[12] C. Katris and S. Daskalaki, "Comparing forecasting approaches for Internet traffic," *Expert Systems with Applications*, vol. 42, no. 21, pp. 8172–8183, 2015.

[13] T. P. Oliveira, J. S. Barbar, and A. S. Soares, "Computer network traffic prediction: a comparison between traditional and deep learning neural networks," *International Journal of Big Data Intelligence*, vol. 3, no. 1, pp. 28–37, 2016.

[14] S. Tanwir, D. Nayak, and H. Perros, "Modeling 3D video traffic using a Markov modulated gamma process," in *IEEE International Conference on Computing, Networking and Communications (ICNC)*, 2016, pp. 1–6.

[15] C. Huang, C. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017, pp. 1–6.

[16] A. Kalampogia and P. Koutsakis, "H.264 and H.265 Video Bandwidth Prediction," *IEEE Transactions on Multimedia*, vol. 20, no. 1, pp. 171–182, 2018.

[17] N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 187–193.

[18] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 114–119, 2019.

[19] Y. Huo, Y. Yan, D. Du, Z. Wang, Y. Zhang, and Y. Yang, "Long-term span traffic prediction model based on STL decomposition and LSTM," in *20th IEEE Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2019, pp. 1–4.

[20] C. Zhang, M. Fiore, and P. Patras, "Multi-Service mobile traffic forecasting via convolutional long Short-Term memories," in *IEEE International Symposium on Measurements & Networking (M&N)*, 2019, pp. 1–6.

[21] Q. He, A. Moayyedi, G. Dán, G. P. Koudouridis, and P. Tengkvist, "A Meta-Learning Scheme for Adaptive Short-Term Network Traffic Prediction," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2271–2283, 2020.

[22] G. Aceto, G. Bovenzi, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Characterization and Prediction of Mobile-App Traffic Using Markov Modeling," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 907–925, 2021.

[23] S. Saha, A. Haque, and G. Sidebottom, "Deep Sequence Modeling for Anomalous ISP Traffic Prediction," in *IEEE International Conference on Communications*, 2022, pp. 5439–5444.

[24] Y. Fang, S. Ergüt, and P. Patras, "SDGNet: a handover-aware spatiotemporal graph neural network for mobile traffic forecasting," *IEEE Commununications Letters*, vol. 26, no. 3, pp. 582–586, 2022.

[25] K. Amarasinghe, K. Kenney, and M. Manic, "Toward Explainable Deep Neural Network Based Anomaly Detection," in *IEEE 11th International Conference on Human System Interaction (HSI)*, 2018, pp. 311–317.

[26] A. Dethise, M. Canini, and S. Kandula, "Cracking Open the Black Box: What Observations Can Tell Us About Reinforcement Learning Agents," in *ACM Workshop on Network Meets AI & ML (NetAI)*, 2019, p. 29–36.

[27] A. Morichetta, P. Casas, and M. Mellia, "EXPLAIN-IT: Towards Explainable AI for Unsupervised Network Traffic Analysis," in *3rd ACM CoNEXT Workshop on Big DAta, Machine Learning and Artificial Intelligence for Data Communication Networks (Big-DAMA)*, 2019, p. 22–28.

[28] S. Rezaei, B. Kroencke, and X. Liu, "Large-Scale Mobile App Identification Using Deep Learning," *IEEE Access*, vol. 8, pp. 348–362, 2020.

[29] C. Beliard, A. Finamore, and D. Rossi, "Opening the Deep Pandora Box: Explainable Traffic Classification," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 1292–1293.

[30] X. Wang, S. Chen, and J. Su, "Real network traffic collection and deep learning for mobile app identification," *Hindawi Wireless Communications and Mobile Computing*, 2020.

[31] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: Encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183, p. 102985, 2021.

[32] I. Akbari, M. A. Salahuddin, L. Ven, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "A look behind the curtain: traffic classification in an increasingly encrypted web," *Proceedings of the ACM on Measurement and Analysis of Computing Systems (POMACS)*, vol. 5, no. 1, pp. 1–26, 2021.

[33] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.

[34] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial Network Traffic: Towards Evaluating the Robustness of Deep-Learning-Based Network Traffic Classification," *IEEE Transactions on Network and Service Management*, vol. 18, no. 2, pp. 1962–1976, 2021.

[35] K. Fauvel, A. Finamore, L. Yang, F. Chen, and D. Rossi, "A Lightweight, Efficient and Explainable-by-Design Convolutional Neural Network for Internet Traffic Classification," in *ACM 29th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[36] T. Zhang, H. Qiu, M. Mellia, Y. Li, H. Li, and K. Xu, "Interpreting AI for Networking: Where We Are and Where We Are Going," *IEEE Commununications Magazine*, vol. 60, no. 2, pp. 25–31, 2022.

[37] X. Wang, S. Chen, and J. Su, "Automatic Mobile App Identification From Encrypted Traffic With Hybrid Neural Networks," *IEEE Access*, vol. 8, pp. 182 065–182 077, 2020.

[38] K. MacMillan, T. Mangla, J. Saxon, and 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.

[39] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *NeurIPS'17*, 2017, pp. 4765–4774.

[40] M. Kull, M. Perello Nieto, M. Kängsepp, T. Silva Filho, H. Song, and P. Flach, "Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with Dirichlet calibration," in *Advances in neural information processing systems*, vol. 32, 2019, pp. 1–11.

[41] J. Mukhoti, V. Kulharia, A. Sanyal, S. Golodetz, P. H. Torr, and P. K. Dokania, "Calibrating deep neural networks using focal loss," in *34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 15 288–15 299.

[42] I. Guarino, G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and 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.

[43] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "MIRAGE: Mobile-app Traffic Capture and Ground-truth Creation," in *IEEE 4th International Conference on Computing, Communications*

and Security (ICCCS), Oct. 2019, pp. 1–8.

[44] A. Nascita, F. Cerasuolo, D. Di Monda, J. T. A. Garcia, A. Montieri, and A. Pescapè, "Machine and deep learning approaches for iot attack classification," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2022, pp. 1–6.

[45] C. Bergmeir and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation," *Information Sciences*, vol. 191, pp. 192–213, 2012.

[46] I. Guarino, A. Nascita, G. Aceto, and A. Pescapè, "Mobile network traffic prediction using high order markov chains trained at multiple granularity," in *IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI)*, 2021, pp. 394–399.

Idio Guarino is a Ph.D. student at the Department of Electrical Engineering and Information Technology of the University of Napoli Federico II. He received the Master Degree in Computer Engineering in July 2020 from the same University, defending a thesis on the modeling and prediction of traffic generated by mobile applications using Markovian approaches. He currently collaborates with members of the Traffic research group at the University of Naples Federico II, which carries out its activities in the area of Computer Networks. In detail, his work focuses on the analysis of network traffic, aimed at the definition of innovative methodologies, based on artificial intelligence, to support the classification and prediction of network traffic.

Giuseppe Aceto is an Associate Professor at the University of Napoli Federico II. He has a Ph.D. in telecommunication engineering from the same University. His work falls in monitoring of network performance and security (focusing on censorship) both in traditional and SDN network environments. He is also working on bioinformatics and ICTs applied to health. He is the recipient of the best paper awards at IEEE ISCC 2010 and IEEE ICCCS 2019, the Computer Networks 2020 Best Paper Award, and the 2018 Best Journal Paper Award by IEEE CSIM.

Domenico Ciuonzo is a Tenure-Track Professor at University of Napoli Federico II. He holds a Ph.D. from the University of Campania L. Vanvitelli. He is the recipient of two Paper awards (IEEE ICCCS 2019 and Elsevier ComNet 2020), the 2019 IEEE AESS Exceptional Service award, the 2020 IEEE SENSORS COUNCIL Early-Career Technical Achievement award and the 2021 IEEE AESS Early-Career Award. His research interests are data fusion, network analytics, IoT, and AI.

Antonio Montieri is an Assistant Professor at the Department of Electrical Engineering and Information Technology of the University of Napoli Federico II. He received his Ph.D. degree in Information Technology and Electrical Engineering in April 2020 and his MS Degree in Computer Engineering in July 2015, both from the same University. He is the recipient of the Computer Networks 2020 Best Paper Award, the best paper award at IEEE ICCCS 2019 and IEEE ISCC 2022, and the best poster award at the TMA Ph.D. School 2016. His work concerns network measurements, (encrypted and mobile) traffic classification, traffic modeling and prediction, and monitoring of cloud network performance. Antonio has co-authored more than 40 papers in international journals and conference proceedings.

Valerio Persico is an Assistant Professor at DIETI, University of Napoli Federico II, where he received the Ph.D. in Computer and Automation Engineering in 2016. His work concerns network measurements, cloud-network monitoring, and Internet path tracing and topology discovery. He has co-authored more than 50 papers in international journals and conference proceedings.

Antonio Pescapè is a Full Professor of computer engineering at the University of Napoli "Federico II". His work focuses on Internet technologies and more precisely on measurement, monitoring, and analysis of the Internet. Recently, he is working on bioinformatic and ICTs for a smarter health. Antonio has co-authored more than 200 conference and journal papers and is the recipient of a number of research awards. Also, he has served as an independent reviewer/evaluator of research projects/project proposals co-funded by several governments and agencies.