Packet-Level Prediction of Mobile-App Traffic using Multitask Deep Learning

Antonio Montieri[a], Giampaolo Bovenzi[a], Giuseppe Aceto[a], Domenico Ciuonzo[a], Valerio Persico[a], Antonio Pescapé[a]

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

Abstract

The prediction of network traffic characteristics helps in understanding this complex phenomenon and enables a number of practical applications, ranging from network planning and provisioning to management, with security implications as well. A significant corpus of work has so far focused on aggregated behavior, e.g., considering traffic volumes observed over a given time interval. Very limited attempts can instead be found tackling prediction at packet-level granularity. This much harder problem (whose solution extends trivially to the aggregated prediction) allows a finer-grained knowledge and wider possibilities of exploitation. The recent investigation and success of sophisticated Deep Learning algorithms is now providing mature tools to face this challenging but promising goal.

In this work, we investigate and specialize a set of architectures selected among Convolutional, Recurrent, and Composite Neural Networks, to predict mobile-app traffic at the finest (packet-level) granularity. We discuss and experimentally evaluate the prediction effectiveness of the provided approaches also assessing the benefits of a number of design choices such as memory size or multimodality, investigating performance trends at packet level focusing on the head and the tail of biflows. We compare the results with both Markovian and classic Machine Learning approaches, showing increased performance with respect to state-of-the-art predictors (high-order Markov chains and Random Forest Regressor). For the sake of reproducibility and relevance to modern traffic, all evaluations are conducted leveraging two real human-generated mobile traffic datasets including different categories of mobile apps. The experimental results witness remarkable variability in prediction performance among different apps categories. The work also provides valuable analysis results and tools to compare different predictors and strike the best balance among the performance measures.

Keywords: Android apps, encrypted traffic, deep learning, mobile apps, multimodal learning, multitask learning, traffic prediction.

1. Introduction

The clear understanding of the processes occurring in networks is paramount for multiple stakeholders, including operators, who aim at the full visibility required by both network management and security [1, 2]. Modeling and predicting network traffic is of the utmost importance to understand traffic peculiarities and properly manage it based on its characteristics. These tools are fundamental to enforce traffic engineering, perform network planning, optimization, provisioning & maintenance, carry out resource allocation and load balancing, manage the QoS, profile user activities, identify anomalies, emulate real traffic for testing purposes, etc.

In practice, the processes to develop such tools have to face the challenges introduced by the nature of the traffic that flows across today's networks. In fact, operators have experienced in the last years tremendous growth of the traffic to be managed in their networks, mostly generated by mobile devices [3].

For instance, according to the latest Ericsson mobility report [4], between Q3 2019 and Q3 2020 (a time-frame including the spread of the current pandemic situation), mobile data traffic grew 50%, driven by both the rising number of smartphone subscriptions and the increasing average data volume per subscription fueled primarily by video content. The report forecasts that mobile subscriptions will reach 8.8 billions by 2026, corresponding to a mobile data traffic of 226 exabytes per month against the 50 exabytes at the end of 2020. In more detail, it is forecasted that the share of video traffic, currently accounting for 66% of all mobile data traffic, will increase to 77% in 2026. Such a huge trend exacerbates the need for accurate modeling and predictability of network traffic generated by mobile devices at fine grain.

However, recent contributions to mobile-traffic prediction mainly focused on traffic at an aggregate scale, both in time (in the order of minutes) and in the nature of traffic (mixing traffic sources, connections, and services) [5, 6, 7, 8]. Opposed to this, a number of problems could benefit from a finer granularity, where the finest is represented by *packet-level*. According to the literature, such an approach results particularly attractive because of its conciseness, flexibility, and its fundamental nature [9]. Additionally, the corresponding applications include all current uses for coarser-grain (aggregated) predictions as a special case. This viewpoint is also motivated by the recent in-

Email addresses: `antonio.montieri@unina.it` (Antonio Montieri), `giampaolo.bovenzi@unina.it` (Giampaolo Bovenzi), `giuseppe.aceto@unina.it` (Giuseppe Aceto), `domenico.ciuonzo@unina.it` (Domenico Ciuonzo), `valerio.persico@unina.it` (Valerio Persico), `pescape@unina.it` (Antonio Pescapé)

terest of a global network solution provider[1]. In fact, research activities in packet-level modeling and prediction are aimed at providing the tools to respond to the real-time needs dictated by rapidly-changing traffic and network conditions. Indeed, per-packet analysis is able to capture traffic peculiarities and variations at *micro-scale*, which would not be visible when performing analysis in an aggregated fashion. Notably, this level of granularity also reflects modern Software-Defined Networking approaches, which manage traffic rules (match-action) at packet level and aim at *per-packet consistency*[2] in network management [10].

Such fine granularity requires more advanced approaches than those usually applied for aggregated network traffic. In fact, the implementation of effective approaches for packet-level traffic characterization and modeling must overcome a number of significant challenges. These issues encompass traffic encryption (with the broad adoption of TLS or other protocols, such as gQUIC), extreme dynamicity and complexity of mobile traffic (with apps weekly updated, potential device/OS/app-version diversity [11, 12], etc.), as well as the lack of public datasets to work with [13].

By extensive research of the related scientific literature, we found recent works aiming at predicting mobile traffic volumes in a given geographical area and dealing with traffic aggregates: despite addressing recent mobile data traffic prediction with advanced methods, these works refer to significantly different phenomenon and problem. We were not able to find a recent proposal or evaluation for packet-level prediction of traffic generated by mobile (and video) apps. Hence, this timely and hard problem remains an open challenge, and even well-established modeling approaches must be re-evaluated and adapted (or possibly superseded) facing this new scenario.

In this work, we aim at investigating the adoption of cutting-edge Deep Learning (DL) methodologies to address the problem of predicting at fine grain (i.e. at *packet level*) the network traffic generated by mobile devices. In detail, we focus on traffic produced by both generic mobile apps (such as those for social, music and audio, and shopping) as well as mobile video apps (video on-demand, video call, short video sharing, and cloud VR) according to the growing interest collected by these apps in the last period. Prompted by the interest of a global network solutions provider, we investigate the applicability of multitask DL architectures to packet-level prediction of network traffic generated by mobile apps, as collected in two human-generated and reliably-annotated datasets. Extracting from traffic traces multimodal features that are accessible also in the common case of encrypted traffic (e.g., information about payload length, inter-arrival times, and TCP flags) we feed a number of state-of-the-art architectures covering the most promising approaches for time series prediction such as Convolutional Neural Networks, Recurrent Neural Networks, and more complex architectures. By designing and deploying a thorough evaluation setup, the prediction performance of the proposed DL-based approaches is compared against state-of-art Markovian and Machine Learning (ML) strategies. As we find that no silver-bullet solution emerges, we discuss the different aspects of the problem (app specificity, biflow initial/ending part) and of the solutions (architecture complexity, memory depth, training procedure, differential Markov analysis), so as to provide the most information and guidance in selecting the approach.

Hence, the contributions of this paper can be summarized as follows:

- we predict the traffic generated by mobile apps at packet-level by means of *multitask Deep Learning* architectures, investigating the impact of packet directions, payload lengths, and inter-arrival times. Indeed, the adoption of multitask learning enables the solution of multiple prediction problems (associated to corresponding traffic parameters) via the design of a *single* DL architecture. Hence, to justify this choice from the performance viewpoint, we preliminarly evaluate the benefits of different design choices leading to *single-modal single-task*, *multimodal single-task*, and *multimodal multitask* architectures in terms of prediction performance as well as model complexity;

- the proposed family of approaches is compared with a relevant set of baselines, ranging from naïve ones to those based on (*i*) multimodal high-order Markov Chains and (*ii*) Machine Learning;

- we evaluate the impact of the memory-window size on prediction performance, and we also investigate the possible advantage obtained in using *exogenous inputs* taken from the traffic data, such as the TCP window sizes, the TCP flags, and the payloads of the transport-layer;

- we analyze the performance of the devised predictors at *fine scale*, focusing on performance trends on the head and the tail of the traffic object considered (i.e. the biflow);

- we provide a first attempt of interpretability of the considered multitask DL predictors, by means of *distillations* toward fixed-order Markov Chains [14];

- we perform all analyses on the *public dataset* Mirage-2019 and the newly-collected Mirage-Video (to be released), adopting the same guidelines as specified in [13], to foster reproducibility.

The paper is organized as follows. Section 2 surveys works closely related to mobile-app network traffic analysis and prediction, positioning our work against past contributions. Sec. 3 describes the considered traffic prediction methodology; the considered experimental setup is described in Sec. 4. Sec. 5 reports the experimental evaluation performed along with in-depth design investigation; finally, Sec. 6 provides conclusions and future perspectives.

[1]NDA prevents the disclosure of further details.

[2]Each packet flowing through the network will be processed according to a single network configuration.

Table 1: Network traffic prediction techniques adopted in related works.

Acronym	Technique
ARIMA	AutoRegressive Integrated Moving Average
ARMA	AutoRegressive Moving Average
ARAR	AutoRegressive AutoRegressive
AVP	Average Value Predictor
CV	Current Value
CNN	Convolutional Neural Network
ConvLSTM	Convolutional LSTM
DCRNN	Diffusion Convolutional Recurrent Neural Network
DQN	Deep Q-Network
DSANet	Dual Self-Attention Network
ESN-DLRS	Echo State Network with Double Loop Reservoir Structure
FARIMA	Fractional Auto-Regressive Integrated Moving Average
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
HW	Holt–Winters
JNN	Jordan Neural Network
k-NNR	K-Nearest Neighbors Regressor
LR	Linear Regressor
LSTM	Long Short-Term Memory
MA	Moving Average
MMG	Markov Modulated Gamma
MC	Markov Chain
MLP	MultiLayer Perceptron
MS-GPR	Multi-Step Gaussian Process Regression
PMF	Persistence Model Forecast
RFR	Random Forest Regressor
RBF	Radial-Basis Function
RCLSTM	Random Connectivity LSTM
RNN	Recurrent Neural Network
SAE	Stacked AutoEncoder
STL	Seasonal-Trend decomposition procedure based on Loess
SVR	Support Vector Regressor
ZP	Zero Predictor

2. Related Work

Forecasting the evolution of network traffic has significantly attracted the interest of the scientific community, which addressed prediction tasks in different flavors and with respect to a number of different practical networking problems. A selection of the most-related works are summarized in Tab. 2, where we highlight their main characteristics. Accordingly, hereafter we discuss them and a more ample set of related works in terms of the prediction granularity (Sec. 2.1), the application context (Sec. 2.2), and the methodology (Sec. 2.3); the section ends with the positioning of our work with regards to the discussed literature (Sec. 2.4).

2.1. Coarse-grained (Aggregated) Prediction

A remarkable amount of works design techniques aimed at *predicting* the evolution of *traffic aggregates* (e.g., volumes or packet rates) over time. In these cases, such studies do not focus on fine-grained inputs to be fed to the proposed models (column *FG-In* in Tab. 2), but rather implement pre-processing strategies to obtain traffic aggregates at different time-resolutions ranging from $\leq$ 1s [16, 19], to few seconds [15, 19, 24], minutes [5, 17, 20, 22, 27], or even hours and days [5, 17, 28]. Such coarse-grained inputs are exploited for a wide range of prediction problems, varying in both the pursued goals and the adopted approaches. Indeed, predicting *aggregated* traffic predates fine-grain predictions, and even new (DL) approaches have been first applied to this arguably simpler problem. The most recent works dealing with prediction of *mobile* traffic consider geographic distribution of data volumes as observed at base stations, e.g. in terms of number of data calls, aggregated with caps at 5MB or 15 minutes duration [5, 6, 7], or at the 3G/4G antennas for 5-minute intervals, differentiating the data transfer (i.e. the sum of upstream and downstream traffic) by application category [8]. An early work along this line is provided by Oliveira et al. [17], comparing performance of simple Neural Network approaches (MLP and Jordan Neural Network) to Deep ones (SAE and deep MLP), showing the best results for the JNN (hence, not one of the DL models). A complex approach (a meta-learning *adversarial* scheme) is proposed by He et al. [32] for short-term prediction of aggregated traffic volume, to inform cellular downlink scheduling and sleep scheduling in user mobile device. Notably, a number of works address the problem in relation to traffic matrices [21, 27, 30]. In the latter cases, the geographical (or topological) distribution of sources and destinations is also taken into account. Hence, the prediction problem benefits from a spatially-characterized information, other than leveraging the (sole) temporal information.

While we have taken into account the proposed methods for our analysis, *we cannot directly compare with this body of results due to the different context and nature of the predicted values*.

2.2. Fine-grained Prediction and Video Traffic

Other works aim at *predicting* the evolution of *finer-grained aspects* characterizing the traffic (i.e. not aggregated). To the best of our knowledge, fine-grained prediction proposals mostly consider video traffic, and specifically *video-frame characteristics* (i.e. application-layer elements) to be predicted [16, 18, 23]. Some works make assumptions about—or try to model—the video-encoding scheme adopted (e.g., considering that the video is generally split into segments each containing a few seconds of content) or the download strategies the clients implement [34]. Some proposals analyze the video traffic at frame (application) level, also making assumptions also about the specific group-of-picture (GOP) patterns adopted for the encoding of the video flow [18, 23, 35, 36, 37].

On the other hand, video traffic is often investigated also in an aggregated fashion. In more general terms, in line with the increasing adoption of the Internet to convey multimedia traffic, many efforts were prompted to characterize and profile video traffic (e.g., identifying the specific streaming phase) [38, 39] and methodologies for predicting the characteristics of adaptive-streaming traffic in real time were extensively investigated in a large body of works [35, 36, 37, 40]. These works are generally aimed at understanding the peculiarities of this traffic (e.g., the streaming strategies adopted [19, 38]), to support traffic engineering and network planning activities.

Table 2: Related works performing prediction of different network traffic parameters and by means of different techniques. The works are reported in chronological order. The last row summarizes the present paper. Meaning of columns and acronyms is reported hereinafter.
Mobile: Mobile traffic data. **Video:** Video traffic data.
Fine-Grain Inputs (FG-In). Fine-Grain Outputs (FG-Out). Exogenous Inputs (Ex-In). Multi-Task (MT). Open Data (OD).
Experimental Validation (EV): Emulation (E), Real (R), Simulation (S).
Parameters: Call Detail Record (CDR), Inter-Arrival Time (IAT), Packet Direction (DIR), Packet Size (PS), Payload Length (PL).
Techniques: for acronyms' definition refer to Tab. 1; + symbol indicates hybrid architectures.
● present, ○ lacking, ◐ partial.

Paper	**Mobile**	**Video**	**FG-In**	**FG-Out**	**Ex-In**	**MT**	**EV**	**OD**	**Parameters**	**Techniques**
A. Dainotti *et al.*, 2008, *Elsevier ComNet* [9]	○	○	●	●	○	●	R	●	PS & IAT	HMM
M. Barabas *et al.*, 2011, *Proc. IEEE ICCP* [15]	○	○	○	○	○	●	R	●	Traffic Volume	MLP
C. Katris and S. Daskalaki, 2015, *Elsevier ESWA* [16]	○	●	◐	◐	○	○	R	●	Traffic Volume Frame Size	FARIMA, RBF, MLP
T.P. Oliveira *et al.*, 2016, *Inderscience IJBDI* [17]	○	○	○	○	○	○	R	●	Traffic Volume	MLP, JNN, SAE
S. Tanwir *et al.*, 2016, *Proc. IEEE ICNC* [18]	○	●	●	●	○	○	R	●	Frame Size	HMM, MMG
A. Biernacki, 2017, *Springer Multimed Tools Appl* [19]	○	●	○	○	○	○	E	○	Video Bw	ARIMA, FARIMA, MLP, RBF
X. Cao *et al.*, 2017, *IEEE Access* [20]	○	●	○	○	○	○	R	○	Traffic Volume	2D-CNN+GRU, ARIMA, SVR, RNN, GRU
C. Huang *et al.*, 2017, *Proc. IEEE PIMRC* [5]	†●	○	○	○	○	●	R	●	† Data CDR count	LSTM, 3D-CNN, 3D-CNN+LSTM, ARIMA, MLP
A. Azzouni *et al.*, 2018, *Proc. IEEE NOMS* [21]	○	○	○	○	○	○	R	●	Traffic Volume	LSTM, MLP, ARMA, ARAR, HW
A. Bayati *et al.*, 2018, *IEEE COMML* [22]	○	○	○	○	○	○	R	●	Traffic Volume	MS-GPR, ARIMA, FARIMA, LSTM, ConvLSTM
A. Kalampogia and P. Koutsakis, 2018, *IEEE TMM* [23]	○	●	●	●	○	○	R	●	B-frame Size	LR, MC
N. Ramakrishnan and T. Soni, 2018, *Proc. IEEE ICMLA* [24]	○	○	○	○	○	○	E/R	◐	Traffic Volume Packet Distr	GRU, LSTM, RNN, CV, MA, ARIMA
J. Zhou *et al.*, 2018, *IEEE Access* [25]	○	○	○	○	○	○	R	●	Traffic Volume	ESN-DLRS
D. Andreoletti *et al.*, 2019, *Proc. IEEE NI* [26]	○	○	○	○	○	○	R	●	Traffic Volume	DCRNN, MLP, LSTM, CNN, CNN+LSTM
Y. Hua *et al.*, 2019, *IEEE ComMag* [27]	○	○	○	○	○	○	R	●	Traffic Volume	RCLSTM, LSTM, ARIMA, SVR, MLP
Y. Huo *et al.*, 2019, *Proc. IEICE APNOMS* [28]	○	○	○	○	○	○	R	●	Traffic Volume	STL+Seq2Seq-LSTM
A. Lazaris *et al.*, 2019, *Proc. IFIP/IEEE IM* [29]	○	○	○	○	○	○	R	●	Traffic Volume	Clustering+LSTM
V. Le *et al.*, 2019, *Proc. IEEE/IFIP IM* [30]	○	○	○	○	○	○	R	●	Traffic Volume	ConvLSTM, ARIMA, LSTM
N. Segolene *et al.*, 2019, *IEEE Access* [31]	○	○	○	○	○	○	S/R	◐	Traffic Volume	LSTM, ARIMA
Q. He *et al.*, 2020, *IEEE JSAC* [32]	○	●	○	○	○	○	R	○	Dw Bytes	PMF, ZP, AVP, LSTM, DQN
G. Aceto *et al.*, 2021, *IEEE TNSM* [14]	●	○	●	●	○	●	R	●	DIR, PL, & IAT	HMM, MC, LR, k-NNR, RFR
L. Nie *et al.*, 2021, *IEEE TII* [33]	○	○	○	○	○	●	R	●	Traffic Volume	LSTM
This Paper	●	●	●	●	●	●	R	●	DIR, PL, & IAT	CNN, LSTM, GRU, SeriesNet, DSANet, MC, LR, k-NNR, RFR

† The work is included as it proposes advanced DL methods, although applied to a different mobile traffic prediction problem (geographic distribution of data calls).

In some cases, the proposals specifically aim at informing the dynamic resource allocation mechanisms (e.g., for bandwidth) or to support adaptive video-streaming and congestion-control algorithms [37, 41]. Multiple-layers analyses have been conducted as well: Katris and Daskalaki [16] investigate the effectiveness of forecasting models regardless the specific abstraction layer, i.e. considering video-frame sizes as well as aggregated volumes and throughput. Differently, Du et al. [41] take into account *traffic intensity* and channel state in order to allocate satellite *transmitting power*.

In other cases, the proposed traffic estimators only focus on data link and network layers information. For instance, Tsilimantos et al. [39] consider parameters such as streaming rate, volumes, burst patterns or flow duration to support *traffic profiling methods*. Other works—rather than aiming at modeling and predicting the sequences of video-frame sizes over time—address the prediction problem focusing on the throughput of the video flows [19, 23, 40].

To the best of our knowledge, *a more limited body of works address prediction at fine grain focusing just on network-layer (i.e. non application-related) aspects, based on packet-level analysis*. Dainotti et al. [9] propose application-specific HMMs to concisely model Internet traffic sources based on inter-packet time and packet size. Their proposal is evaluated using real traffic and specifically considering SMTP, HTTP, online gaming, and online messaging. More recently, Aceto et al. [14] propose HMMs and Markov Chains to characterize and predict network traffic generated by mobile apps. While the work also proposes a reconstructing heuristic to infer the application-layer message size, the input used to train the model and perform a prediction are all strictly related to network layer (namely being the payload size, the inter-arrival time, and the packet direction).

2.3. Prediction Approaches

Focusing on the prediction strategies adopted by these works, from Tab. 2 it is evident that the interest of the scientific community is more and more captured by the rise of ML- and DL-based methodologies. While statistical approaches such as ARMA, ARIMA, or FARIMA [5, 16, 19, 21, 22, 30, 31], RFR [14], or SVR [20, 27] are adopted, they are more often considered as a performance baseline to evaluate ML and DL. Among the latter, CNN [5, 20, 26], LSTM [5, 21, 22, 24, 26, 27, 28, 29, 30, 31, 32], and GRU [20, 24] or their combinations [5, 20, 26] are often reported to achieve better performance than selected baselines. Accordingly, we build on these performance results and investigate these promising approaches. Table 1 summarizes the techniques adopted for traffic prediction found in the reviewed literature together with the associated acronyms, for convenience.

Based on our literature survey, only a limited number of examples exist where either emulation [19, 24] or simulation [31] is adopted (column *EV* in Tab. 2). Most of the proposals are evaluated leveraging real data. Due to the critical role played by data that are fundamental not only in validating and comparing the proposals, the considered works mostly rely on open datasets (column *OD* in Tab. 2). This notwithstanding, works that go against this general trend can be found [19, 20, 32].

2.4. Positioning of Our Work

In our study, we aim at the finest-grained prediction: *packet-level*. Accordingly, rather than aggregated volumes or rates (e.g., [32]), our proposal aims at forecasting per-packet traffic characteristics such as (*i*) packet directions, (*ii*) payload lengths, and (*iii*) inter-arrival times. Note that the aggregated prediction problem is a coarser one and that following our proposed approach, aggregated predictions could be derived by post-processing the outputs obtained via the proposed models.

In terms of granularity and viewpoint, the closest works to ours (as discussed in Sec. 2.2) are represented by [9, 14]. Specifically, the former work focuses on the (preliminary) aspect of traffic modeling of *unidirectional flows* (i.e. neglecting the advantage obtained by taking into account request-response interaction) by means of HMMs on desktop-generated (i.e. non mobile-app generated) traffic.

Conversely, the focus of the recent work [14] is on bidirectional flows generated by mobile apps, similarly to this paper. Nonetheless, the above work is mainly devoted to traffic modeling and characterization, and therein *the prediction task is only tangentially touched*. Indeed, Markov and ML models proposed therein are only preliminarily evaluated for the prediction task on Mirage-2019. On the contrary, in this work we focus entirely on the prediction task, and we propose novel multitask DL approaches to accomplish this objective. Still, for the sake of a complete comparison, the predictors originating from the models in [14] are included in the baselines later introduced in Sec. 4.3. Furthermore, the prediction performance is analyzed in depth via a per-packet-index evaluation. Finally, a *larger experimental evaluation* is performed on *two* datasets, one comprising generic mobile apps (Mirage-2019) and another collecting mobile video apps (Mirage-Video).

While focusing on the prediction of packet directions, payload lengths, or inter-arrival times, in addition to consider the historical time series of the parameters we aim to predict, differently than *all* the listed literature (column *Ex-In* in Tab. 2), we also evaluate the benefits deriving from the adoption of exogenous inputs (such as TCP flags and TCP window size, or transport-layer payload).

Considering the centrality of video traffic and related applications in today's networks, we (also) investigate and evaluate prediction strategies for video traffic prediction. Indeed, video traffic characteristics have been investigated at several levels of abstraction and considering diverse granularity levels. Differently than proposals that exploit application-layer information to model and predict video traffic (e.g., [16]), our methodology does not rely on information other than that related to the IP layer, which is always available also in case typical encryption strategies are put in place. The proposed approach (similarly to [39]), is not based on any specific assumption related to the content of the datagrams composing network flows, is more general, and can be applied to network traffic regardless of the specific application generating the traffic.

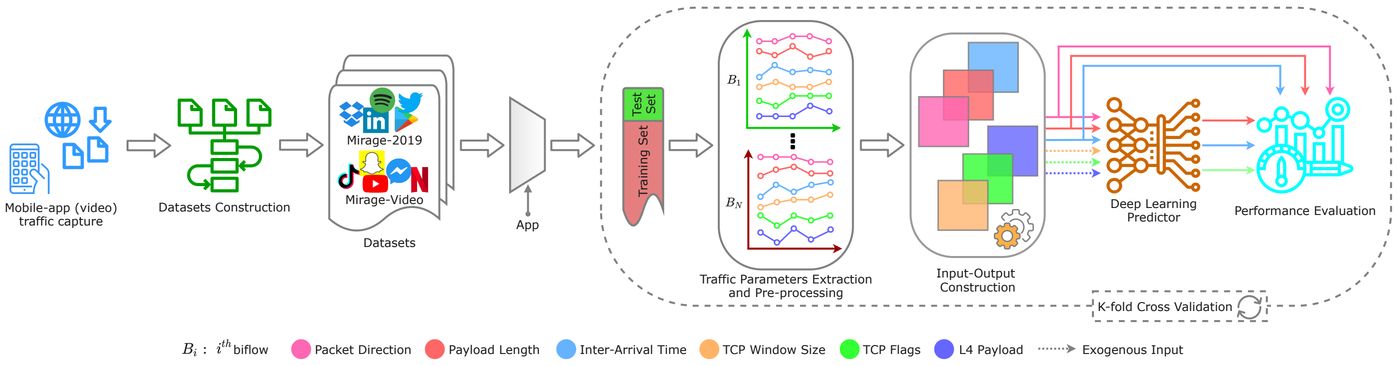


Figure 1: Workflow of the proposed methodology for mobile-app (video) traffic prediction via multitask DL approaches.

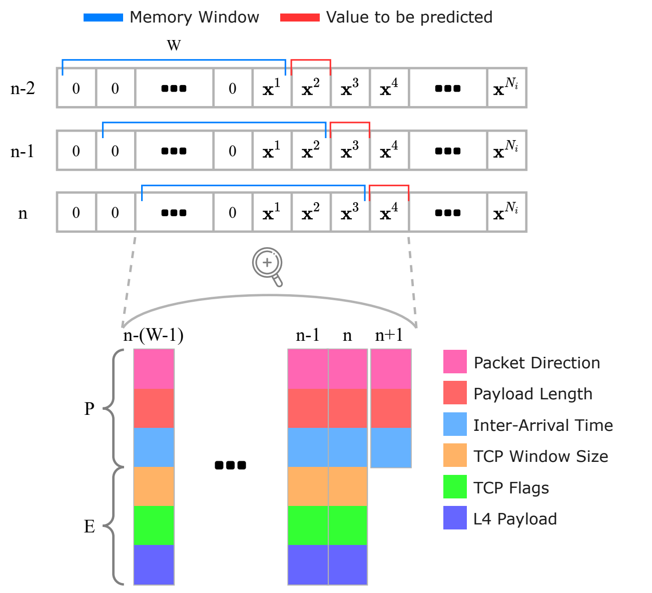


Figure 2: Input-output construction: $\mathbf{x}^n$ is the n^{th} vector element of the multivariate time series of length N_i extracted from the biflow B_i; W is the size of the memory window (input), $n+1$ is the index of the vector value to be predicted (output).

3. Traffic Prediction Methodology

The objective of this work is to predict network traffic generated by a mobile app at the finest (*packet-level*) granularity. The approaches we adopted for traffic prediction leverage real mobile-app traffic datasets. In detail, *labeled* traffic is required (i.e. information about the specific app that generated the traffic is needed) to support the training of the models and evaluate their prediction capability. Beyond this, the proposed methodology is agnostic with respect to the specific traffic to be predicted and can be thus generally applied *as is* to other traffic (e.g., newer mobile-app traffic). For our experimentation, we took advantage of *two* traffic datasets of mobile apps (a generalist one and one specific for video apps) obtained via the Mirage architecture [13], as detailed in later Sec. 4.1.

Figure 1 depicts the workflow based on the proposed methodology for the prediction of mobile-app traffic via DL approaches. After collecting the traffic generated by handheld devices, it is pre-processed in order to extract the traffic parameters of interest (see Sec. 3.2) and build the data structure to feed the models (see Sec. 3.3). Details about the considered DL models are provided in Sec. 3.4. Before describing the blocks constituting the proposed architecture, we provide the formulation of the specific problem we aim to resolve in the next Sec. 3.1.

3.1. Formulation of the Prediction Problem

Traffic prediction can be conducted at different granularity levels that reflect how network packets are aggregated, namely which is the *Traffic Object (TO)* constituting the elementary unit for the prediction task [14]. In detail, we aggregate packets in bi-directional flows (*biflows*), defined by the quintuple (`IP src, IP dst, port src, port dst, protocol`) considering both directions of communication (i.e. the source and the destination pairs are interchangeable).

Considering the sequence of packets of the biflow, the traffic parameters associated to each packet constitute a multivariate time series, whose temporal index is the packet order-of-arrival. Given a biflow up to its n-th packet, the objective of this work is to predict P *traffic parameters* associated to the (n+1)-th packet. These parameters are collected in the vector $\boldsymbol{x}^{n+1}$ and constitute the outputs of the considered DL architectures. The above predictions are based on the observation of previous values of the same traffic parameters (the latest observation represented as $\boldsymbol{x}^n$ for the n-th packet). More in general, the observations refer to a *memory window* of size W, which gathers the observations of the parameters for the W most recent packets. These observations, namely $\boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)}$, are used as inputs of the architectures. The size of the memory window W determines the memory that the model needs to retain to make a prediction. On one hand, a higher value of W typically implies higher complexity (as the total number of input values is proportional to W), with repercussions on memory consumption, training time, and execution time. On the other hand, an increase of the prediction window can provide benefit in prediction performance, by allowing the capitalization of longer-range dependence in the considered traffic parameters. In addition, we also evaluate the benefits obtained when considering *exogenous inputs*, namely E further traffic parameters for each packet observed so far (collected in the vector $\boldsymbol{e}^n$ for the n-th packet) given as input

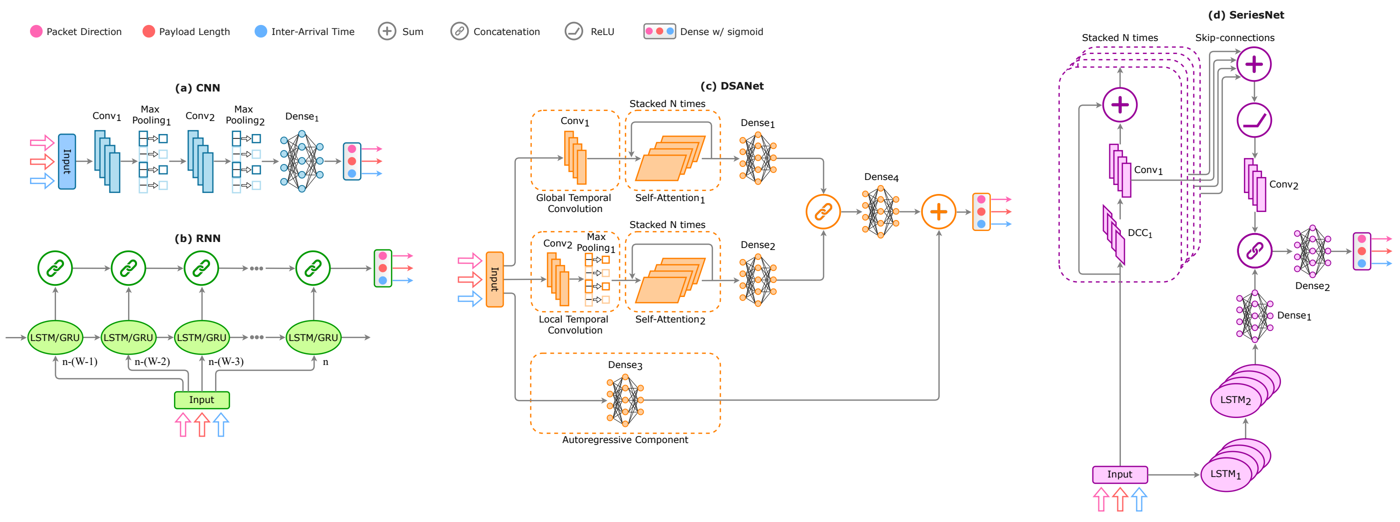


Figure 3: Architectures of considered DL-based approaches for traffic prediction: (a) Convolutional Neural Network (CNN), (b) Recurrent Neural Network (RNN), (c) DSANet, and (d) SeriesNet. All the architectures are terminated by one dense layer (with sigmoid activation) for each feature to predict. Hyperparameters of each layer are reported in Tab. 3.

to aid the DL model in the prediction task, but not included in the set of parameters to be predicted.

Accordingly, the design of a model $\mathcal{M}$ for prediction in our case specializes into:

$$\hat{\boldsymbol{x}}^{n+1} = \mathcal{M}(\underbrace{\boldsymbol{x}^{n}, \boldsymbol{x}^{n-1}, \ldots \boldsymbol{x}^{n-(W-1)}}_{\text{prev. traffic parameters}}, \underbrace{\boldsymbol{e}^{n}, \boldsymbol{e}^{n-1}, \ldots \boldsymbol{e}^{n-(W-1)}}_{\text{exogenous inputs}}) \quad (1)$$

Given the multivariate nature of the prediction task considered, we investigate *multitask architectures* [42] that jointly address multiple prediction tasks, one for each of the parameters P considered.

3.2. Traffic Parameters Extraction and Pre-processing

In this work, we aim at predicting $P = 3$ traffic parameters. In detail, we consider the *direction* (DIR), the *payload length* (PL), and the *inter-arrival time* (IAT) of the packets belonging to the same biflow, following a common choice for fine-grained network traffic description [43]. Specifically, the DIR is a binary value indicating if the packet is downstream or upstream, whereas the PL is the size (in bytes) of the transport-layer payload. Finally, the IAT (in microseconds) is the time between the arrival of two consecutive packets. We recall that the above parameters at preceding instants are also used as inputs in the memory of length W (see Eq. (1)).

Additionally, as anticipated in Sec. 3.1, for each packet we consider as (optional) exogenous inputs: the *TCP window size* (TWIN) being the size of the TCP congestion window[3], the *TCP flags* (FLG) encoded as an eight-bit vector denoting the presence (1) or absence (0) of each flag[4], and the *L4 payload* (PAY) being the transport-layer raw payload of the first 32 packets of each biflow arranged in a byte-wise fashion.

Going into details, traffic parameters are extracted from raw-packet sequences of each biflow which are preliminarly pre-processed to effectively feed the DL models. Firstly, we remove *zero-payload* packets that are assumed to be non-informative since they only reflect transport-layer mechanisms (e.g., TCP handshake, pure acknowledgments, etc.) rather than application behaviors. Consequently, we compute the IATs on the resulting packet sequences. For the IAT, the minimum granularity is set to 1μs. Secondly, to remove the effects of outliers in measuring times, we saturate all the IATs to the 99^{th}-percentile values of their distributions. In more detail, for the Mirage-2019 dataset, the 99^{th}-percentile value equals 1.75s, whereas for Mirage-Video this corresponds to 36.97ms (see later Sec. 4.1). This represents a first indicator of the peculiarity of mobile-app traffic when used for video-intensive applications and the need for having two separate datasets for the experimentation.

To effectively exploit DL models we apply scaling procedures to both the main P traffic parameters and the E exogenous inputs. It is worth noticing that parameters represented as binary values or vectors (i.e. DIR and FLG, respectively) do not need a scaling procedure. Conversely, we use a min-max scaler with $[min, max] = [0, 1]$ to scale the PL and IAT within this range, whereas for the scaling of the TWIN we apply the quantile transformer. Such a method transforms the TWIN parameter to follow a uniform distribution, tending to spread out the most frequent values and reducing the impact of (marginal) outliers.[5] Finally, the PAY parameter (transport-layer payload) is normalized by dividing each byte value by 255.

[3]The window size is multiplied by a scaling factor negotiated between the sender and the receiver (RFC 1323: `https://rfc-editor.org/rfc/rfc1323.txt`) and it is set to 0 for UDP packets.

[4]The TCP flags are—in order—FIN, SYN, ACK, RST, PSH, URG, ECE, and CWR (RFC 3168: `https://tools.ietf.org/html/rfc3168`).

[5]We underline that we have tested the quantile transformer also on PL and IAT without experiencing any difference in experimental results.

3.3. Input-Output Construction

Irrespective of the specific ML/DL model, when dealing with time series a common methodology is applied to construct the input for the predictor and the respective output for its training and validation. To this aim, we propose a windowing approach based on a *sliding memory window* of size of W and with a unit stride, as shown in Fig. 2 (detail in the lower half).

Furthermore, the proposed windowing approach is based on *incremental windowing* [44]; namely, it leverages incrementally-sized sets of samples until reaching the prescribed maximum size W of the memory window, after which the window size remains constant. In this way, the predictions can be made as soon as the first sample is available, in contrast to a fixed windowing that relies on a fixed number of samples inside the memory window and thus needs to collect a number of samples equal to W before performing a prediction. In other words, the incremental-windowing approach extends the fixed-windowing one, enabling to model also the "border effects" (viz. predict early behaviors). Consequently, to allow our models to perform an early prediction and also to cope with biflows having length shorter than W, we apply a (left) *zero-padding* up to W samples (see top of Fig. 2). Note that zero padding is applied to `PAY` also in the case the windows spans over packets beyond the 32^{nd}.

As a result, for the generic time instant n, this procedure results in an input matrix $\boldsymbol{I}^n$ (of size $(E+P)\times W$) and a prediction vector $\boldsymbol{x}^{n+1}$ (of size P, representing the desired output), constructed applying the windowing approach described above to each of the biflows within the considered set (e.g., those that are generated by the same app). The bottom of Fig. 2 depicts the resulting structures highlighting with different colors the traffic parameters, corresponding to one prediction sample. Accordingly, for the i^{th} biflow, with corresponding length N_i, this procedure generates $N_i - 1$ prediction samples.

3.4. Deep Learning-based Approaches for Traffic Prediction

In this section, we describe the DL models used to perform traffic prediction and their training procedure. In particular, these DL models can be divided into three macro-categories, namely (*i*) convolutional, (*ii*) recurrent, (*iii*) and composite neural architectures. We provide the details on both architectures of DL models employed and related hyperparameters in Fig. 3 and Tab. 3, respectively.

Convolutional Neural Networks (CNNs). The CNNs are DL models inspired by the functioning of human-being visual cortex that have found huge applicability in image analysis, natural-language processing, and, more recently, in traffic analysis [45]. A CNN consists of a sequence of *convolutional layers*, encompassing *transition-invariant filters* with a certain "receptive field" that represents their (limited) extent. These filters are convolved with the input to automatically extract features from an input region whose dimensions depend on the receptive field. To reduce model complexity and mitigate overfitting, in a CNN architecture, each convolutional layer is commonly followed by a *pooling layer* that performs a down-sampling of the intermediate representation output of the convolution. Common choices are max-pooling and average-pooling. Herein, we consider the CNN architecture depicted in Fig. 3a, whose hyperparameters (cf. Tab. 3) are set based on state-of-the-art works [45, 46].

Recurrent Neural Networks (RNNs). The RNNs are DL architectures, characterized by loopy connections, that recall the values over time (via a state vector $\boldsymbol{h}[t]$) of an input sequence of length T. The most common variants of RNNs are the *Long Short-Term Memory (LSTM)* and the *Gated Recurrent Unit (GRU)* networks. Both offer a solution to the "short-term memory" limitation of RNNs—as they can model dynamic temporal-behaviors with *long-term dependencies*—and can be defined in a *bidirectional* variant, in which their internal representation is split into forward and backward directions[6]. LSTM and GRU networks are made of elementary components (the *units*) having internal mechanisms (the *gates*) that can regulate the information flow by learning which values in a time-sequence are important to *recall* or *forget*. The main difference between LSTM and GRU is that the latter has a simpler structure than the former (i.e. two vs. three gates), and consequently it has fewer parameters to train. In the next analyses, we employ the LSTM and GRU shown in Fig. 3b and detailed in Tab. 3.

Composite Neural Networks. In addition to CNNs and RNNs, we also investigate *DSANet* [47] and *SeriesNet* [48], being state-of-the-art composite neural networks successfully used for multivariate time-series forecasting in banking and financial domains.

The *dual self-attention network (DSANet)* is a composite DL architecture proposed in [47] for forecasting multivariate time series that exhibit dynamically-periodic or non-periodic behaviors. Specifically, DSANet utilizes two parallel 1D-CNN-based components: (*i*) *global temporal convolution* and (*ii*) *local temporal convolution* that are in charge of embedding each univariate time series composing the multivariate input into two representation vectors with global (viz. long-term) and local (viz. short-term) temporal patterns, respectively. Each representation of such a univariate time series is given as input to a *self-attention module [49] with residual connections* to capture the dependencies between the extracted representations. The specific architecture of DSANet is illustrated in Fig. 3c. The hyperparameters[7] are given in Tab. 3.

The *SeriesNet* [48] is a DL model based on *Dilated Causal Convolutions (DCCs)*, originally proposed in [50]. Causal convolutions ensure that the model does not depend on future time-steps (i.e. the model preserves the input ordering), whereas dilated convolutions significantly extend the receptive field of

[6]We compared the performance of unidirectional and bidirectional versions (not shown for brevity), and having found equivalent outcomes, we selected the simplest (unidirectional) one.

[7]Hyperparameters have been tuned based on common values suggested in state-of-the-art works [47] and validated with a set of experiments not shown for the sake of brevity.

filters by skipping input values with a certain step[8]. In our analyses, we employ an extended version (with the stacking of two LSTM layers) of the original SeriesNet proposed in [48], whose architecture is depicted in Fig. 3d, while the hyperparameters of each layer are reported in Tab. 3.

Multitask Loss Specification and Training Procedure. DL methods share the property that the *training phase* is performed in an iterative fashion leveraging the *stochastic gradient descent* (first-order) optimization algorithm (which approximates the optimal solution) for finding the minimum of a *cost (or loss) function*. The training phase is performed leveraging a subset of size N_B of the biflows associated to the considered app, and constituting the training set $\mathcal{T}$. The latter is formally defined as

$$\mathcal{T} \triangleq \bigcup_{i=1}^{N_B} \left\{ \boldsymbol{I}^n(B_i),\, \boldsymbol{x}^{n+1}(B_i) \right\}_{n=1}^{N_i} \tag{2}$$

In other terms, the training set corresponds to the union of N_B sets, with the i^{th} set containing all the prediction samples associated to the biflow B_i (of length N_i), obtained according to the procedure described in Sec. 3.3. For the sake of a compact notation, in what follows we denote the overall number of samples within $\mathcal{T}$ with N. We recall that the considered prediction setup differs from those commonly employed in time-series prediction which leverage part of past observations to update the model or to learn a time-series-specific one. Such a setup would probably get better performance, but its real-world implementation is practically infeasible in online context due to the complexity related to the fine-grained prediction task and the sophisticated prediction model considered. On the basis of these considerations, we have chosen the *biflow-based cross-validation granularity*.

Moreover, given the multitask nature of the employed architectures, the loss function to optimize (viz. minimize) depends on the specific parameter to predict (viz. the prediction task to address). In this work, we are concerned with the prediction of P *traffic parameters* of the next packet—constituting the outputs of the considered DL architectures—collected in the vector $\boldsymbol{x}^{n+1}$. Accordingly, we aim to minimize a *weighted sum* of the losses of the P prediction tasks considered, namely:

$$\mathcal{L}\left(\boldsymbol{\theta}_{\text{shared}}, \left\{\boldsymbol{\theta}_p\right\}_{p=1}^{P}\right) \triangleq \sum_{p=1}^{P} \lambda_p\, \mathcal{L}_p\left(\boldsymbol{\theta}_{\text{shared}}, \boldsymbol{\theta}_p\right) \tag{3}$$

Since our architecture is in charge of solving multiple learning tasks at once, the weight λ_p represents the preference level of the p^{th} task in the multitask objective function to be optimized. In detail, we optimize these weights resulting in $\lambda_1 = \lambda_2 = 0.45$ and $\lambda_3 = 0.10$, where $p = 1$, $p = 2$, and $p = 3$ are associated to the `DIR`, `PL`, and `IAT` prediction tasks, respectively, thus focusing less on the latter. In the above equation $\boldsymbol{\theta}_{\text{shared}}$ collects the set of parameters associated to the layers shared by the different tasks, whereas $\boldsymbol{\theta}_p$ collects the parameters associated to the task-specific layers of the p^{th} task.

[8]In a dilated convolution, the filter is applied over an area larger than its size, corresponding to a zero-dilation of the original filter.

In detail, we aim to minimize the *binary cross-entropy* loss function for the prediction of binary `DIR`:

$$\mathcal{L}_{dir}^{\text{bce}}(\cdot) \triangleq -\frac{1}{N} \sum_{i=1}^{B} \left\{ \sum_{n=0}^{N_i-1} x_{dir}^{n+1}(B_i)\, \log(p_{dir}^{n+1}(B_i)) \right. \\ \left. + (1 - x_{dir}^{n+1}(B_i)) \log(1 - p_{dir}^{n+1}(B_i)) \right\} \tag{4}$$

Differently, for the prediction of the p^{th} non-binary parameter, such as `PL` and `IAT`, the considered DL predictors are trained to minimize the *Mean Squared Error (MSE)* loss, namely:

$$\mathcal{L}_{p}^{\text{mse}}(\cdot) \triangleq \frac{1}{N} \sum_{i=1}^{B} \sum_{n=1}^{N_i-1} (\hat{x}_p^{n+1}(B_i) - x_p^{n+1}(B_i))^2 \tag{5}$$

In both Eqs. (4) and (5), we recall that N denotes the total number of training samples, i.e. the cardinality of $\mathcal{T}$, and N_i the number of packets of the i^{th} biflow B_i. Additionally, $x_{dir}^{n+1}(B_i)$ denotes the `DIR` traffic parameter associated to the $(n+1)^{th}$ packet of the biflow B_i, whereas $p_{dir}^{n+1}(B_i)$ denotes the prediction probability associated to $\hat{x}_{dir}^{n+1}(B_i) = \text{DW} = 1$. Similarly, $x_p^{n+1}(B_i)$ denotes the p^{th} traffic parameter associated to the $(n+1)^{th}$ time instant of the biflow B_i, whereas $\hat{x}_p^{n+1}(B_i)$ denotes the corresponding prediction.

We perform the optimization by employing the *Adam* optimizer [51] with a batch size of 32, a learning rate of 10^{-3}, and exponential decay rates for the estimates of the first-order and second-order moments equal to 0.9 and 0.999 (Keras default values), respectively. Overall, each DL architecture is trained for 150 epochs. Also, to prevent overfitting we leverage the *early-stopping* technique with a patience of 4 epochs and a minimum delta of 10^{-4} measured on the training loss.

4. Experimental Setup

In this section, we define the experimental setup adopted to conduct our experimental evaluation. We first give the details on the collected datasets in Sec. 4.1. Then, the evaluation setup and metrics employed to assess prediction performance are reported in Sec. 4.2. Finally, Sec. 4.3 describes the baselines against which the DL predictors are compared.

4.1. Datasets Description

The experimental evaluation performed herein relies on two mobile-app traffic datasets[9]: Mirage-2019 and Mirage-Video. The former collects traffic generated by generic mobile apps (i.e. the set encompasses diversified use cases and purposes), whereas the latter includes traffic gathered by mobile-apps generating different kinds of video traffic. Both datasets have been collected employing the Mirage architecture [13] in the ARCLAB laboratory of the University of Napoli “Federico II” by more than 280 human experimenters that have collaborated on a

[9]Mirage-2019 is currently available at `http://traffic.comics.unina.it/mirage` for reproducibility. Differently, Mirage-Video is planned to be released in the near future.

Table 3: Hyperparameters of DL-based architectures for traffic prediction depicted in Fig. 3.

DL Model	#TP [k]	Layer	Details
CNN	441	Conv_1	Filters: 32, Kernel size: 5, Activation: ReLU
		Max $\mathrm{Pooling}_1$	Pool size: 3
		Conv_2	Filters: 64, Kernel size: 5, Activation: ReLU
		Max $\mathrm{Pooling}_2$	Pool size: 3
		Dense_1	Neurons: 128, Activation: ReLU
RNN	164/124	LSTM/GRU	Type: Unidirectional, Units: 200, Activation: Sigmoid
DSANet	201	Conv_1	Filters: 30, Kernel size: 10, Activation: ReLU
		Conv_2	Filters: 7, Kernel size: 5, Activation: ReLU
		Max $\mathrm{Pooling}_1$	Pool size: 3
		Self-$\mathrm{Attention}_1$	N: 4, h: 3, d_{model}: 7, d_{ff}: 4
		Dense_1	Neurons: 128, Activation: ReLU
		Self-$\mathrm{Attention}_2$	N: 4, h: 3, d_{model}: 7, d_{ff}: 4
		Dense_2	Neurons: 128, Activation: ReLU
		Dense_3	Neurons: 200, Activation: ReLU
		Dense_4	Neurons: 200, Activation: ReLU
SeriesNet	532	DCC_1	N: 7, Filters: 32, Kernel size: 2, Dilation: 2^i, Activation: SELU
		Conv_1	N: 7, Filters: 1, Kernel size: 1, Activation: Linear
		Conv_2	Filters: 1, Kernel size: 1, Activation: Linear
		LSTM_1	Type: Unidirectional, Units: 200, Activation: Sigmoid
		LSTM_2	Type: Unidirectional, Units: 200, Activation: Sigmoid
		Dense_1	Neurons: 128, Activation: ReLU
		Dense_2	Neurons: 128, Activation: ReLU

#TP stands for Number of Trainable Parameters.
For all DL models, the Convolutional (Conv) layers are one-dimensional.
DSANet: in the N stacked Self-Attention layers (encompassing a multi-head attention and a position-wise feed-forward network), h denotes the number of attention layers (heads) in multi-head attention, d_{model} denotes the dimension of queries, keys, and values of multi-head attention, and d_{ff} denotes the inner layer dimensionality of the position-wise feed-forward network [48, 49]. A 10% dropout is applied to each sub-layer composing the convolutional-based components.
SeriesNet: in the DCC layer, the dilation value is set to 2^i, with $i \in \{0, N-1\}$ and N denoting the number of stacked DCC layers. SELU stands for Scaled Exponential Linear Unit. An 80% dropout is applied after the last two stacked DCC layers to deal with time series with shorter lengths.

volunteer basis. We have employed three mobile devices (i) Xiaomi Mi5, (ii) Google Nexus 7, and (iii) Samsung Galaxy A5 with Android 6.0.1 to collect the traffic generated by generic and video mobile-apps during the time periods May 2017 - May 2019 and Jun. 2019 - Mar. 2020, respectively.

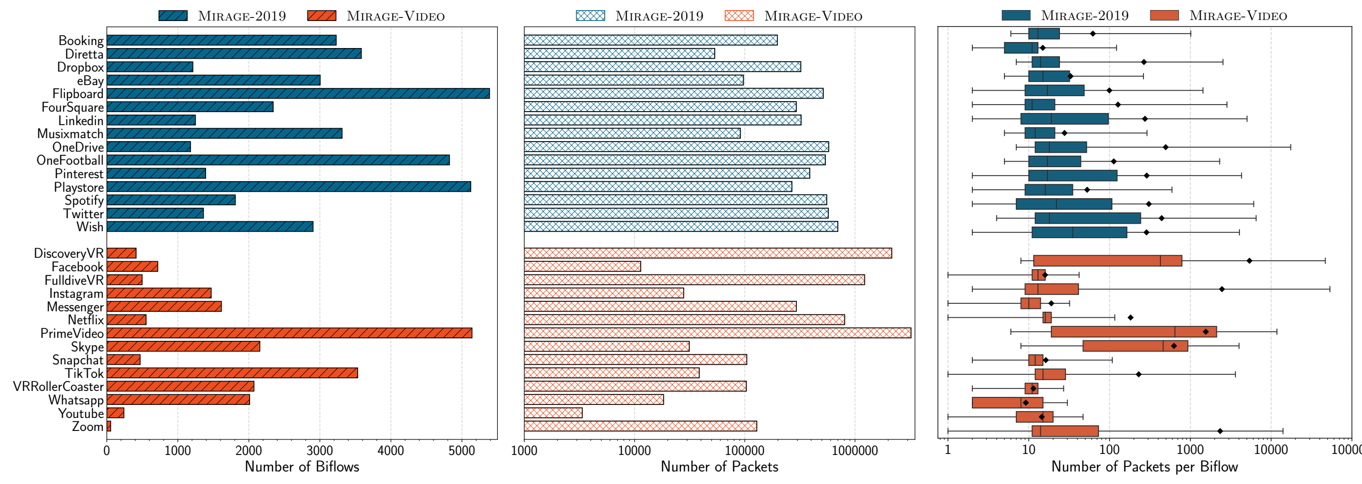
Along with each mobile-traffic trace in PCAP format, the Mirage architecture also collects a system log-file with ground-truth information. The latter allows to reliably label each biflow with the corresponding Android-package name matching the related 5-tuple. Specifically, we consider both network-related system-calls (via `strace`) and established network-connections (via `netstat`), and associate each *<IP:port>* in the socket descriptor with the name of the Android package that has made the call or to which the socket belongs (i.e. that is listening on the *<IP:port>* pair of the socket). Every capture session in both datasets refers to the latest version of the app at the time of the capture.

On the whole, the Mirage-2019 dataset [13] gathers the traffic generated by 40 Android apps belonging to 16 different categories (according to the Google Play Store[10]). In detail, the experimenters have mimicked the everyday usage of apps to explore their most-common functionalities (e.g., login, registration, usual activities, etc.). Altogether, Mirage-2019 consists of more than 4600 (biflow-labeled) traffic traces each generated in a capture session of 5 ÷ 10 mins. For brevity, the following analysis focuses only on a subset comprising 15 apps. This was meant to avoid redundancy in the analyses and to guarantee variety to the mobile apps of Mirage-2019, i.e. by considering apps belonging to different categories.

On the other hand, Mirage-Video encompasses the traffic of 14 Android video apps characterized by different purposes, functionalities, and video contents, namely: *cloud VR* (e.g., `DiscoveryVR`, `FulldiveVR`, and `VRRollerCoaster`), *short video* (e.g., `Instagram`, `Snapchat`, and `TikTok`), *video chat* (e.g., `Messenger`, `Skype`, `Whatsapp`, and `Zoom`), and *video on demand* (e.g., `Facebook`, `Netflix`, `Prime Video`, and `Youtube`) apps. In this case, the duration of capture sessions depends on the type and content of videos played (e.g., a sequence of one-minute-long short videos, a two-hour-long movie on demand, etc.). It is worth to notice that for what concerns Mirage-Video, users have specifically leveraged the video-based functionalities of the app during the capture, also when apps would have been allowed for other usages.

Figure 4 provides a characterization of Mirage-2019 (top)

[10]`https://play.google.com/store/apps`



(a) Biflows. (b) Packets. (c) Distribution of biflow lengths.

Figure 4: Number of biflows (a), number of per-app packets (b), and distribution of the number of packets per biflow (c) for the applications in both Mirage-2019 and Mirage-Video. The values displayed in (a), (b), and (c) do not include zero-length biflows and zero-payload packets. In (c), the whiskers represent the 1st and 99th percentiles, and the black diamond depicts the mean value.

and Mirage-Video (bottom), focusing on the sets of apps employed in the next analyses. In detail, Fig. 4a and Fig. 4b report the number of biflows and packets for each app, respectively. Comparing the values for different apps, it is evident that these values remarkably depend on the specific app considered: for 14 apps out of the 29 considered, the dataset contains more than 2000 biflows. The number of biflows available for each app ranges from 55 (for `Zoom`) to 5389 (for `Flipboard`). On the other hand, the dataset contains at least 10000 packets for all the apps but `Youtube`. In fact, certain apps are characterized by fewer but longer biflows (e.g., `Dropbox`, `DiscoveryVR`, and `Zoom`) or vice-versa (e.g., `Diretta` and `TikTok`). In order to further investigate these discrepancies, Fig. 4c reports the distribution of the number of packets in a biflow, for each app. This analysis highlights that despite the variability in biflow lengths depends on the specific app (see different spans for inter-quartile ranges in Fig. 4c) the traffic collected for most of the apps results in biflows ranging from 100 to 1000 packets, on average. In more detail, a number of apps for which a reduced number of biflows is available (such as `DiscoveryVR`, `FulldiveVR`, and `Zoom`) are associated to longer biflows, on average. On the one hand, this explains the discrepancies in values shown in Fig. 4a and Fig. 4b. On the other hand, it witnesses the richness in terms of variety of network behaviors that different apps in the datasets assume.

4.2. Evaluation Setup and Metrics

We evaluate the performance of mobile-traffic prediction via a *ten-fold cross-validation* setup, that uses (for each fold) 90% of the biflows belonging to a given app for building the training set and the remaining 10% for test set construction (i.e. for evaluation purposes), representing a solid assessment procedure. Consequently, the overall performance of each considered model is obtained by collecting the (possibly-different) results pertaining to the ten different folds and performing summary statistics (i.e. average and standard deviation) of the evaluation metrics considered.

Regarding performance measures, we use the *G-mean* as a compact measure to evaluate the prediction performance of the (binary-valued) `DIR` traffic parameter, namely:

$$\text{G-mean} \triangleq \sqrt{\rho_{dir}^{\text{dw}}\,\rho_{dir}^{\text{up}}} \tag{6}$$

with $\rho_{dir}^{\text{dw}} \triangleq \Pr(\hat{x}_{dir} = \text{DW} \mid x_{dir} = \text{DW})$ and $\rho_{dir}^{\text{up}} \triangleq \Pr(\hat{x}_{dir} = \text{UP} \mid x_{dir} = \text{UP})$. In this case, x_{dir} is associated to the sequence of `DIR`s assumed by real traffic and $\hat{x}_{dir}$ the sequence of `DIR`s provided by the prediction model, whereas DW and UP indicate downstream and upstream direction, respectively. The probabilities of correctly predicting the direction of downstream (ρ_{dir}^{dw}) and upstream packets (ρ_{dir}^{up}) are simply estimated as the fraction of correct downstream and upstream prediction events (for direction), respectively, divided by the total number of predictions $\bar{N}$.

Differently, to assess the predictive capabilities of `PL` and `IAT`, we leverage the Root Mean Squared Error (RMSE) defined as:

$$\text{RMSE}_p \triangleq \sqrt{\frac{1}{\bar{N}} \sum_{j=1}^{\bar{N}_B} \sum_{n=1}^{\bar{N}_j - 1} \left[\hat{x}_p^{n+1}(\bar{B}_j) - x_p^{n+1}(\bar{B}_j)\right]^2} \tag{7}$$

where $\bar{N}$ denotes the total number of predictions (the cardinality of the test set), $\hat{x}_p^{n+1}(\bar{B}_j)$ 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 $x_p^{n+1}(\bar{B}_j)$ the corresponding value provided by the prediction model.

4.3. Considered Baselines

DL-based approaches described in Sec. 3.4 are compared with different baselines belonging to the (*a*) Markovian and (*b*) Machine Learning families. Additionally, we show also the results obtained with the (naïve) *current-value baseline* (hereinafter simply referred to as *Baseline*), whose predictions are $\hat{\boldsymbol{x}}^{n+1}_{\text{base}} \triangleq \boldsymbol{x}^n$, that is it predicts the next observation value with the current one.

Markovian Approaches. We consider (multimodal) high-order Markov Chains (MCs), that—once trained via a maximum-likelihood procedure—leverage the *predictive distribution* $\Pr(\boldsymbol{x}^{n+1} \,|\, \boldsymbol{x}^n, \ldots, \boldsymbol{x}^0)$ to make the predictions.[11] The latter is obtained from the joint initial distribution $\Pr(\boldsymbol{x}^{W-1}, \ldots, \boldsymbol{x}^0)$—for $n = -1, \ldots, W-2$—or from the transition distribution $\Pr(\boldsymbol{x}^{n+1} \,|\, \boldsymbol{x}^n, \ldots, \boldsymbol{x}^{n-(W-1)})$—for $n \geq (W-1)$—with W being the order of the MC. The actual predictions $\hat{\boldsymbol{x}}^{n+1}$ are obtained from the predictive distribution via the posterior mean[12], which minimizes the *Mean Squared Error*:

$$\hat{\boldsymbol{x}}^{n+1}_{\text{mmse}} \triangleq \sum_{j=1}^{S} \boldsymbol{s}_j \; \Pr(\boldsymbol{x}^{n+1} = \boldsymbol{s}_j \,|\, \boldsymbol{x}^n, \ldots \boldsymbol{x}^{n-(W-1)}) \qquad (8)$$

with S being the Cartesian product of the individual modality spaces.

Machine Learning (ML) Approaches. We also investigate the performance of *three* state-of-the-art ML-based regressors:

- *Linear Regressor (LR)*: it models the relationship between a set of input variables collected in the random vector $\boldsymbol{x}$ (i.e. the observations within the memory window) and the dependent output variable $\boldsymbol{y}$ (i.e. the predicted features of the next observation), using a linear relationship. Typically, to determine the best fit, the *least-squares method* is employed, thereby minimizing the sum of squares of the deviations between each point and the multiple-regression line.
- *k-Nearest Neighbors Regressor (k-NNR)*: it is an ML algorithm that performs predictions based on the assumption that similar (input) elements are closer together (i.e. lead to almost the same output). Specifically, the k-NNR uses the *k*-nearest neighbors found within the training set on the basis of a certain distance metric (e.g., Minkowski distance) to predict unknown samples. The prediction is then obtained as the (weighted) average value of the relevant neighbors.
- *Random Forest Regressor (RFR)*: it is an ensemble of T decision trees, therefore constituting a "forest" of simpler estimators. The forest is built at training time exploiting the ideas of bagging and random-feature selection to mitigate over-fitting. Specifically, each tree is trained on (*i*) different bootstrap realizations of the training set and (*ii*) with some additional randomization during the tree construction to minimize the mean squared error between the predictions and the actual values. After training, the prediction is made by taking the (weighted) average of the responses of the T trees.

[11]The Markov property allows the use of a sliding window consisting of the last W observations (cf. Sec. 3.3), that are used as input to the algorithm.

[12]We remark that a posterior mode estimator, corresponding to a maximum a-posteriori approach, can be also considered (as in [14]). However, the posterior mode was shown to perform worse than the mean in our preliminary results, not shown for brevity. For this reason, we did not report it in the subsequent analysis.

5. Experimental Evaluation

In this section, we report experimental evaluation performed leveraging Mirage-2019 and Mirage-Video datasets. More specifically, in Sec. 5.1, we preliminarily assess the validity of using multitask architectures and multiple monitoring parameters (i.e. multimodal multitask predictors). Then, in Sec. 5.2, we evaluate the impact of the memory-window size W on the overall performance. Accordingly, in Sec. 5.3, we provide a performance comparison for the considered prediction approaches on the two datasets. Such performance analysis is complemented by a complexity viewpoint in Sec. 5.4.

The second part of the present section delves with more in-depth design and analysis. Accordingly, we investigate the potential advantage given by the additional use of exogenous inputs in Sec. 5.5. The performance of the resulting predictors is then investigated at fine scale, by focusing on the head and tail of biflows in Sec. 5.6. Finally, an interpretability analysis of the behavior of the predictors is provided in Sec. 5.7.

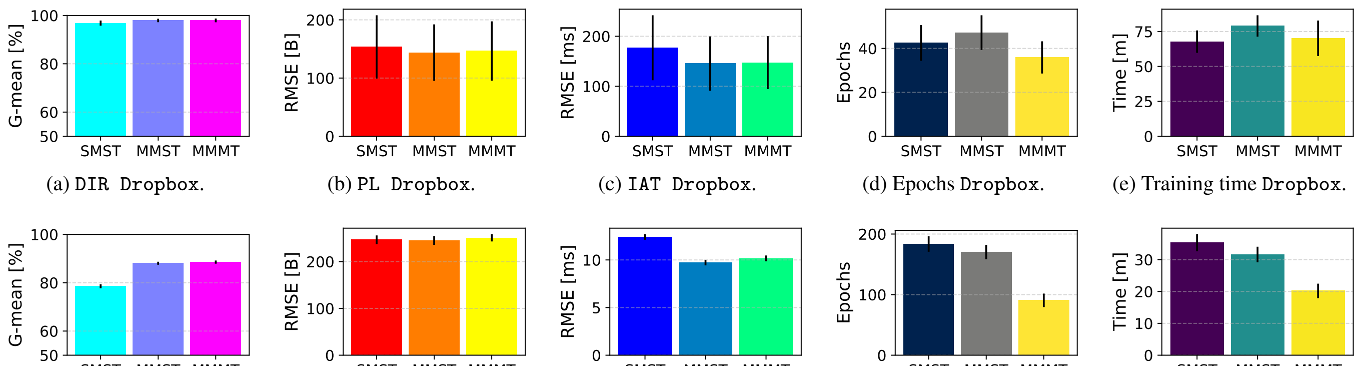
5.1. Single-task vs Multitask Predictors

First of all, we focus on the *actual need* for using a multitask DL predictor for forecasting the three considered relevant traffic parameters: `DIR`, `PL`, and `IAT`. Accordingly, in Fig. 5 we report the prediction performance of one of the considered multitask DL architectures (namely a CNN[13] with a memory-window size of $W = 30$) on two relevant apps taken from Mirage-2019 and Mirage-Video, namely `Dropbox` (top row) and `TikTok` (bottom row), respectively.

The proposed multitask CNN architecture (referred shortly to as `MMMT`, i.e. *multimodal multitask*), is compared against *two* relevant architectural baselines:

1. three single-task CNN architectures each *predicting one traffic parameter* based only on the last $W = 30$ samples of the *same* parameter (e.g., predicting next `DIR` based on the last W values of `DIR` observed). The first baseline is referred to as *single-modal single-task* (`SMST`);
2. three single-task CNN architectures, each *predicting one traffic parameter* based on the last $W = 30$ input parameters (e.g., predicting next `DIR` based on the last W values of (`DIR, PL, IAT`) observed). The second baseline is referred to as *multimodal single-task* (`MMST`).

[13]We refer only to CNN for conciseness. Similar considerations can be made for the other DL predictors considered.



(a) `DIR` `Dropbox`. (b) `PL` `Dropbox`. (c) `IAT` `Dropbox`. (d) Epochs `Dropbox`. (e) Training time `Dropbox`.

(f) `DIR` `TikTok`. (g) `PL` `TikTok`. (h) `IAT` `TikTok`. (i) Epochs `TikTok`. (j) Training time `TikTok`.

Figure 5: Comparison between different design implementations of CNN architectures: single-modal single-task (`SMST`), multi-modal single-task (`MMST`), and multi-modal multi-task (`MMMT`) architectures over `DIR`, `PL`, and `IAT` in terms of prediction performance, number of epochs, and training time.

For both apps, we analyze: (a) the prediction performance regarding each traffic parameter predicted (i.e. in terms of G-mean for `DIR` and RMSE for `PL`/`IAT`) and (b) the complexity associated to each solution (reporting the cumulative number of epochs for which the solution is trained and the overall training time). Indeed, the above analysis is intended to assess the benefit granted by multitask learning in training/using a *single DL architecture* to predict multiple traffic parameters. This is in contraposition with respect to the aforementioned two baselines, *both requiring a number of DL architectures equal to the number of traffic parameters to be predicted.* Accordingly, this constitutes a sharp difference in terms of the relevant computational complexity. However, while successful capitalization of multitask learning is expected to outperform SMST baselines also in terms of prediction performance (because of the use of multi-modality of the inputs), the corresponding comparison with MMST is less obvious and worth investigating.

Indeed, results highlight that the `SMST` solution tends to perform the worse (in terms of prediction performance) over the three considered traffic parameters, due to the input separation in the three `SMST` architectures. This is especially true for `IAT` prediction and `DIR` prediction on `TikTok`. Additionally, the adoption of an `MMST` solution (i.e. a full separate architecture for the prediction of each parameter) does not lead to an appreciable prediction-performance improvement with respect to `MMMT` (i.e. a single architecture performing a joint multitask prediction). However, the adoption of a `MMMT` predictor is particularly appealing from the complexity viewpoint, due to the significantly-lower training time involved, witnessed by a lower number of total epochs required for training one architecture in the place of three.

Main Remarks: On average, `SMST` architectures are outperformed by both `MMST` and `MMMT` solutions which fruitfully benefit from multi-modality and expose better performance in `DIR`, `PL`, and `IAT` prediction (i.e. result in higher G-mean and lower RMSE). In addition, `MMMT` architectures are specifically appealing from the complexity viewpoint, with reduced training time w.r.t. to `MMST`. Indeed, the training phase for the latter architectures is accomplished via a higher number of epochs due to the need for using a different DL architecture to predict each traffic parameter (viz. task).

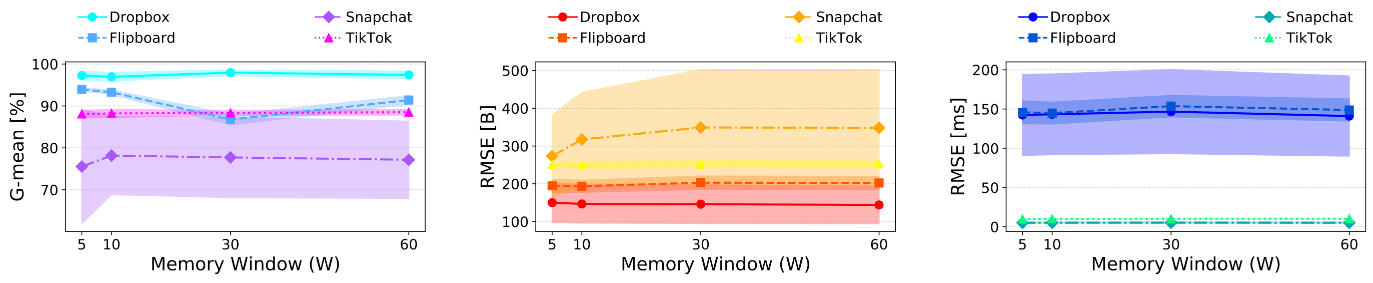
5.2. Impact of the Memory-Window Size

In this section, we provide an investigation of the impact of the memory-window size W on the performance of DL architectures, considering a CNN model as an example and taking into account four apps (namely two generic, i.e. `Dropbox` and `Flipboard`, and two video, i.e. `Snapchat` and `TikTok`). This analysis is useful to properly tune the memory window, in order to select a common value that gives suitable performance over all the apps.

The results, depicted in Fig. 6, show the G-mean for `DIR` and the RMSE for both `PL` and `IAT`, by varying the memory-window size $W \in \{5, 10, 30, 60\}$. By inspecting Fig. 6a, we can notice that the G-mean on `DIR` shows a saturation of performance gain once reached the $W = 30$ memory-window size for `Dropbox`, `TikTok`, and `Snapchat`. `Flipboard` represents an exception, presenting the opposite behavior with the G-mean that drops reaching $W = 30$ and rises with $W = 60$. Focusing on `PL` RMSE in Fig. 6b, performance gains saturate (again) with a memory-window size of 30, when considering `Dropbox`, `Flipboard`, and `TikTok`, while a counterintuitive behavior is shown by `Snapchat`, which exhibits an RMSE growth up to $W = 30$ at which it reaches a plateau. Finally, when considering the `IAT` performance in terms of RMSE (Fig. 6c), it results to be practically insensitive to W.

In conclusion, a conservative choice can be drawn from the highlighted results, which is the selection of $W = 30$ as the target memory-window size for all the considered apps of both datasets, because of the trade-off between the complexity of the model (which grows with the size of memory window) and the effectiveness of a prediction being capable to model enough history of biflows, and because the `DIR` output gains from this choice on most of the considered apps.

Main Remarks: In spite of rare non-straightforward trends, best performance figures are observed by adopting a memory of



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

Figure 6: Comparison of CNN prediction performance against memory-window size W for DIR, PL, and IAT.

size $W = 30$. The latter choice also represents the best trade-off between model complexity and prediction performance.

5.3. *Overall Picture of Prediction Performance*

In this section, we provide an overall view of the prediction performance achieved over *all* the apps of the two datasets considered. Accordingly, Fig. 7 shows the prediction performance of multitask DL models on DIR, PL, and IAT for mobile apps belonging to the MIRAGE-2019 dataset when a memory-window size $W = 30$ is employed. Similarly, Fig. 8 shows the prediction performance of multitask DL models on mobile video apps belonging to MIRAGE-VIDEO in the same setup. In both cases, the performance of the baselines described in Sec. 4.3 is reported for the sake of a complete assessment and comparison. To ease the reading, for each app and traffic parameter we highlight the best performing predictor with a "★" marker.

Referring to the DIR parameter (Fig. 7a), for 10 out of 15 considered apps in MIRAGE-2019, multitask DL architectures are able to outperform all the considered baselines, namely the (current-value) Baseline, ML approaches, and Markovian approaches. Remarkably, when multitask DL architectures do not perform the best, MCs represent the highest-performing alternative in 4 out of 15 apps, thus confirming their appeal in modeling and prediction of mobile traffic [14]. A more pronounced result holds for the apps taken from MIRAGE-VIDEO (Fig. 8a), for which multitask DL architectures are *always* able to surpass the considered baselines. Interestingly, for 12 out of 14 apps, SeriesNet is able to outperform all the other predictors. In the two remaining cases (i.e. PrimeVideo and TikTok), the highest performance is achieved by a multitask GRU. By looking at the app-averaged performance (*AVERAGE*, reported on the leftmost group of both Figs. 7a and 8a), it is apparent that on MIRAGE-2019 (resp. MIRAGE-VIDEO) dataset the best DIR predictor is a multitask GRU (resp. SeriesNet).

Differently, concerning PL (Figs. 7b and 8b) and IAT (Figs. 7c and 8c) parameters, the situation is more varied, with many apps (in both MIRAGE-2019 and MIRAGE-VIDEO datasets) leading to an ML-based predictor (namely, the RFR) performing the best. Specifically, although the PL average performance of the best multitask DL predictor is not far from that of the RFR, in the case of IAT, multitask DL architectures incur only in a slight RMSE degradation, with GRU model being always among the best three predictors for both PL and IAT, on average.

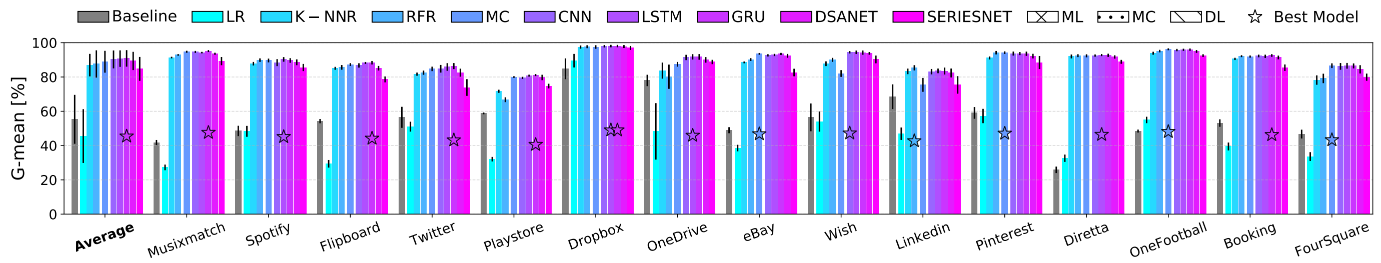
Main Remarks: The best predictor varies with both the app and the parameter to predict. Concerning DIR, on average, (multitask) DL architectures outperform the others (the best DL architecture achieves approximately +3% G-mean w.r.t. the best baseline, on average). On the other hand, RFR provides the best performance for PL and IAT prediction, on average, with DL approaches reporting marginal degradation (+5 B *and* +8.7 ms *RMSE for PL and IAT, respectively).*

5.4. *Investigation of Computational Complexity*

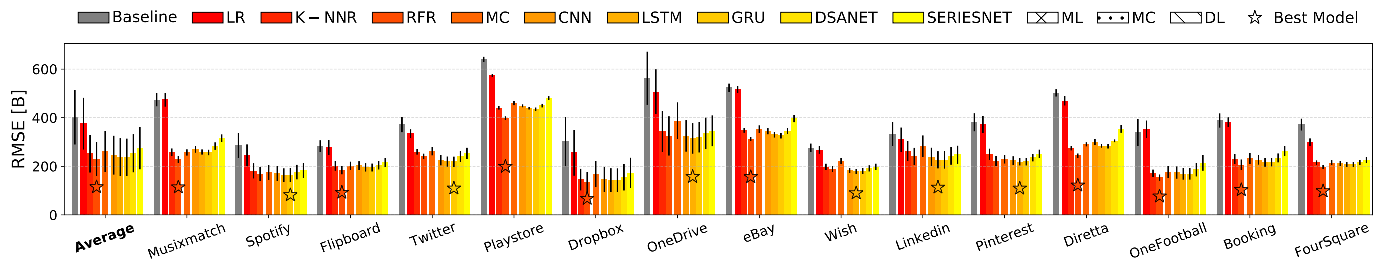
To assess the difference in complexity among the different DL methods, we analyze the *training time*, the number of training *epochs*, and the number of *trainable parameters* for each approach. We recall that training epochs are capped to a maximum of 150, but *early-stopping* causes an end of this phase if convergence on the loss is achieved (cf. Sec. 3.4): a higher number of epochs needed to converge can be associated with a higher complexity of the architecture. Similarly, a higher number of trainable parameters is also an index of more complex architectures.[14] It is worth to notice that, since different types of elementary layer (e.g., recurrent vs. convolutional) also contribute to the above aspect, we center the discussion on complexity using overall (re-)training time to provide more directly actionable information.

In Fig. 9 we report these metrics and the corresponding prediction performance for DIR, PL, and IAT. The considered DL methods are sorted by *increasing training time*, and values refer to a non-video app (Flipboard, Fig. 9a) and a short-video one (TikTok, Fig. 9b). Several considerations can be derived from these results. First, it is evident that the training time varies significantly across the different DL methods, and between the two apps (belonging to two different categories), ranging from ≈ 6

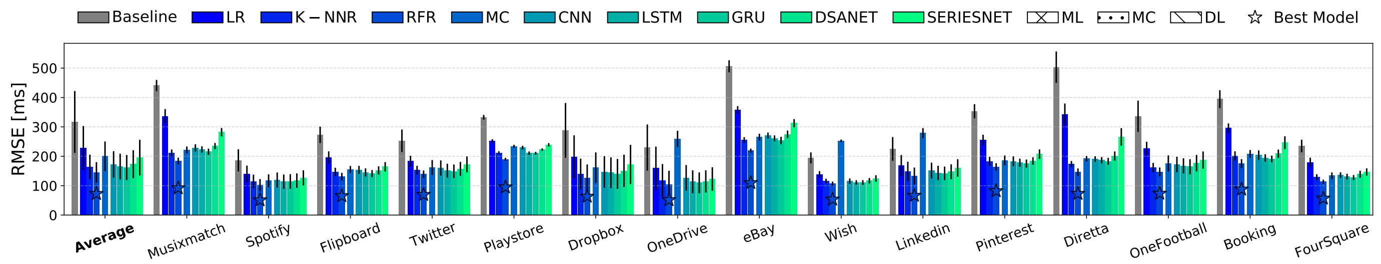
[14]We highlight that the number of trainable parameters is intrinsically related to the specific architecture of the DL model employed (see Fig. 3 and Tab. 3) and naturally changes with the model itself (i.e. it is a metric to evaluate the complexity of the DL model). Nevertheless, to foster a fair comparison between the different architectures, we set identical tunable parameters concerning all controllable aspects of models' training procedure, namely: total number of epochs, optimizer, batch size, learning rate, and early-stopping parameters (cf. Sec. 3.4 for details).



(a) DIR



(b) PL



(c) IAT

Figure 7: Prediction performance of DL models on mobile apps belonging to MIRAGE-2019 on DIR (a), PL (b), and IAT (c). The best performing predictor for each app and traffic parameter is highlighted via a "★" marker.

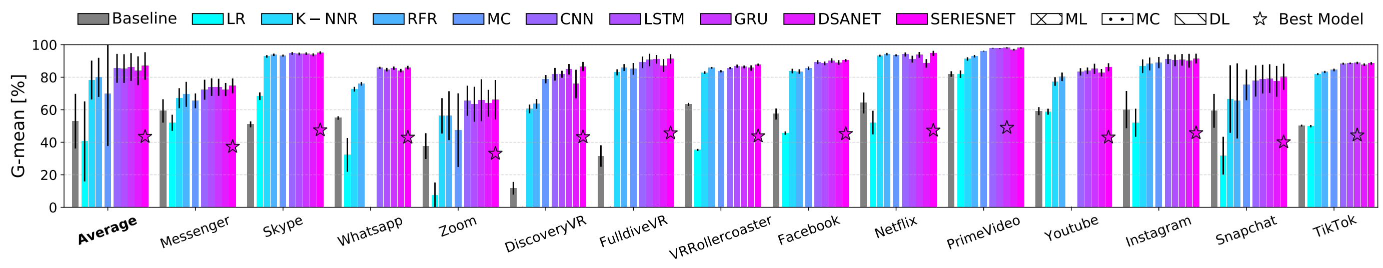
minutes for CNN on TikTok, to almost 690 minutes for GRU on Flipboard. However, GRU on Flipboard (resp. CNN on TikTok) does not show the highest (resp. lowest) number of training epochs when compared to the other DL models trained on the same app. Similarly, the number of trainable parameters is not in a monotonic relationship with the training time for either app. In addition, the observed prediction performance does not grow monotonically with training time, number of epochs, or trainable parameters. For instance, the worst DIR prediction performance on Flipboard is attained by SeriesNet, which is associated with the largest number of trainable parameters; also, although CNN and SeriesNet expose remarkable different training time, their DIR prediction performance is practically equivalent when considering TikTok.

Main Remarks: Higher complexity does not necessarily relate with better performance. Since the best multitask DL architecture varies with the application / predicted traffic parameter, this outcome opens the possibility for selecting architectures based on the specific goal, investigating the existing trends and identifying the proper solution based on better (per-parameter) prediction performance and/or shorter (re-)training time.

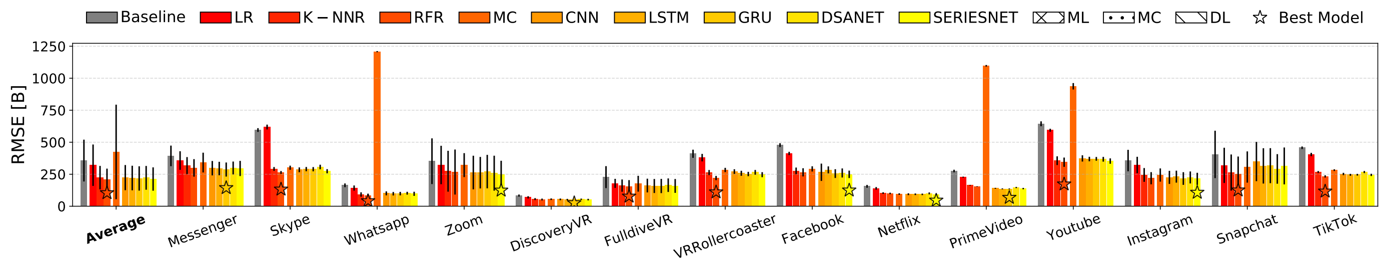
5.5. *Prediction Performance with Optional Inputs*

In this section, we assess the possible improvements in terms of DIR (Fig. 10), PL (Fig. 11), and IAT (Fig. 12) led by considering in the prediction task the use of exogenous inputs. For the sake of brevity, we focus on the performance of Snapchat, TikTok, Dropbox, and Flipboard. Specifically, we investigate the possible advantage from introducing exogenous inputs, namely: TWIN, FLG, and PAY (as described in Sec. 3.2). For brevity, we focus only on three multitask DL predictors: CNN, LSTM, and GRU. Herein, we investigate *four* different additions of exogenous inputs: (*i*) TWIN, (*ii*) FLG, (*iii*) PAY, and (*iv*) TWIN + FLG. These (groups of) inputs are adopted in addition to the non-exogenous ones (DIR, PL, and IAT), which are collectively reported as 3F in the figures.

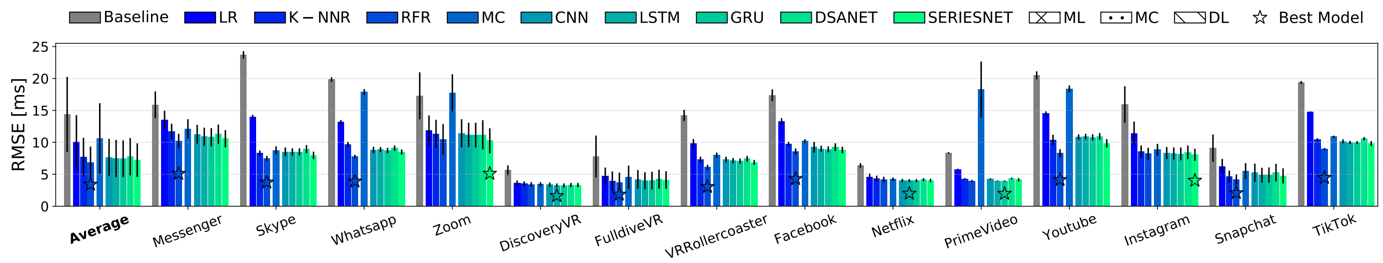
As a general comment, the use of the considered combinations of exogenous inputs provide some prediction gains (although not significant) and results vary from app to app, also depending on the peculiar DL architecture and the specific traffic parameter (the best configurations are marked with a "★" in the figures). Concerning the DIR prediction (Fig. 10), the only app that does not benefit from any addition of exogenous inputs



(a) DIR



(b) PL



(c) IAT

Figure 8: Prediction performance of DL models on mobile video apps belonging to Mirage-Video on DIR (a), PL (b), and IAT (c). The best performing predictor for each app and traffic parameter is highlighted via a "★" marker.

is Dropbox, whereas for the other three apps the situation is more varied. For instance, the sole use of TWIN provides improved prediction performance for Snapchat on CNN/LSTM, while the use of FLG (alone or in combination with TWIN) is able to improve the G-Mean of TikTok. Conversely, the use of PAY is beneficial to the GRU in the case of Snapchat and TikTok.

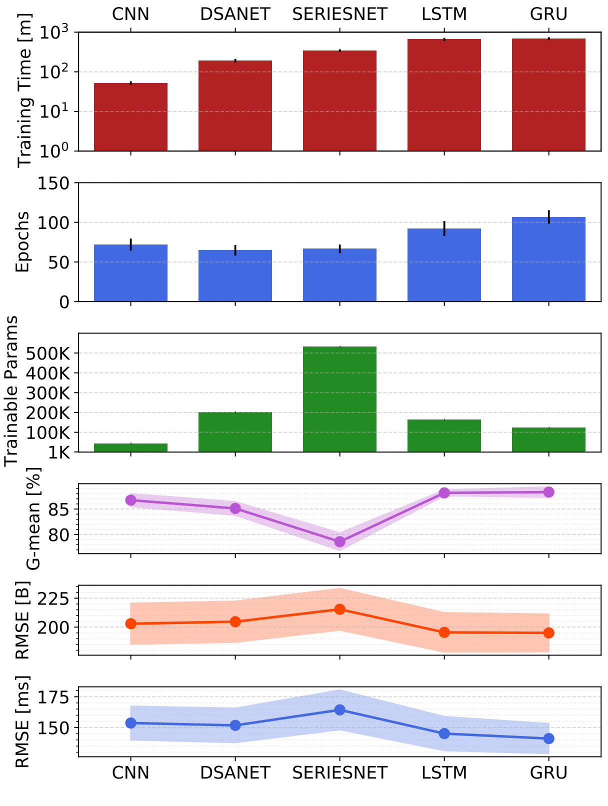
Differently, concerning PL prediction (Fig. 11), it is interesting to note that the use of TWIN + FLG is able to provide an RMSE reduction for all the three considered multitask DL architectures on TikTok. A similar observation applies to Flipboard when considering PAY effects on CNN and GRU architectures. Finally, focusing on IAT prediction (Fig. 12), a notable pattern can be observed for TikTok, where the sole use of TWIN is able to reduce the RMSE in the prediction of IATs. A similar observation can be drawn for Flipboard, where TWIN is beneficial for both CNN and LSTM, whereas the GRU performance are improved by the use of PAY.

Main Remarks: While the addition of exogenous inputs does not significantly enhance the prediction performance in general, for specific choices of mobile app and architecture it is worth considering. 3F+TWIN+FLG proved to be the most effective in the considered examples.

5.6. Per-packet-index Performance

In this section, we provide a fine-grained investigation of prediction performance metrics (i.e. G-mean for DIR and RMSE for PL and IAT). We refer to this analysis as *per-packet-index performance*, because the evaluated metrics are computed considering packets lying at the same position in biflows, i.e. sharing the same index (or falling within the same interval of indexes). The idea of this analysis is to evaluate whether and how prediction performance varies along a biflow, i.e. if packets appearing at specific positions in the biflows are responsible for worse or better performance. The main intent is to detect possible consistent discrepancies between the beginning of the biflows (where "introductory" information exchanges are likely to happen) and the rest, as well as at remarking the peculiarities of the results attained for the ending packets of biflows (where a "closing" exchange of messages could be present).

Accordingly, the analysis focuses on (*i*) the head of the biflows (i.e. the first 128 packets) and (*ii*) their tail (i.e. the last


(a) Flipboard.

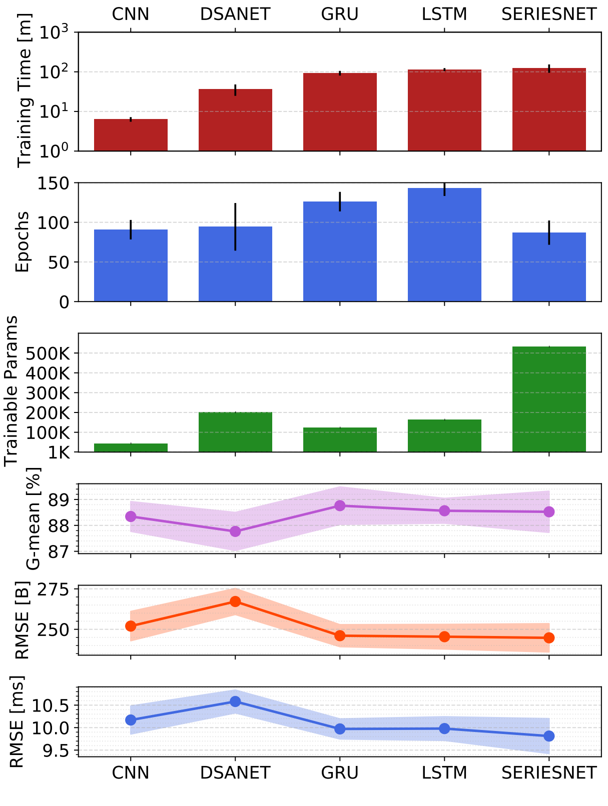
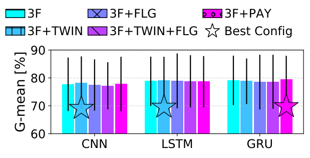
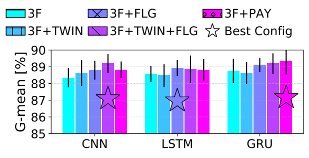
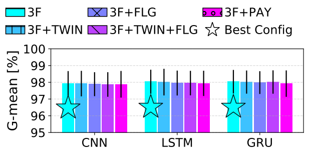

(b) TikTok.

Figure 9: Comparison of complexity of DL predictors in terms of training time, number of epochs, and number of trainable parameters (sorted by the first); below, the corresponding performance is reported for DIR, PL, and IAT. Values refer to a non-video application (a) Flipboard and a short-video one (b) TikTok.


(a) Snapchat


(b) TikTok


(c) Dropbox

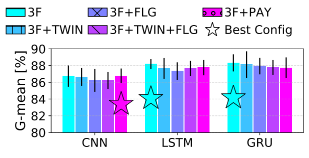
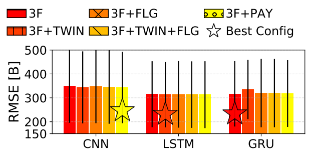
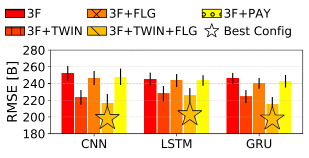
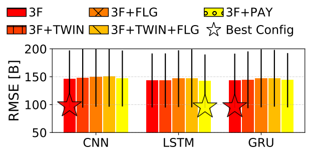

(d) Flipboard

Figure 10: Prediction performance for DIR when feeding the DL models with optional inputs (i.e. TWIN, FLG, and PAY).


(a) Snapchat


(b) TikTok


(c) Dropbox

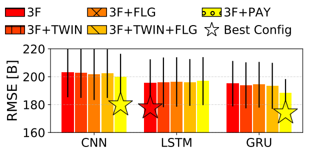
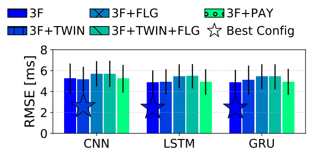
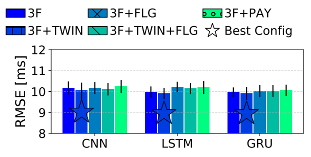

(d) Flipboard

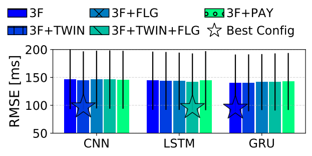
Figure 11: Prediction performance for PL when feeding the DL models with optional inputs (i.e. TWIN, FLG, and PAY).



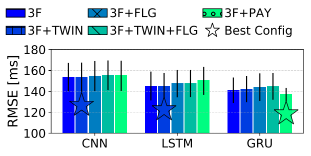
(a) Snapchat



(b) TikTok



(c) Dropbox



(d) Flipboard

Figure 12: Prediction performance for IAT when feeding the DL models with optional inputs (i.e. TWIN, FLG, and PAY).

16 packets). In the former case, we show aggregated performance each 2 packets until the 32nd and each 8 for the remaining until the 128th, along with (*a*) the overall G-mean/RMSE, (*b*) the G-mean/RMSE of the first 32 packets, and (*c*) the G-mean/RMSE of the remaining packets until the *end* of each biflow. In the latter case, we show per-packet performance index-by-index (viz. without aggregation). Note that these specific visualization choices are loosely informed by previous experiments and do not heavily affect the nature of the outcomes and their discussion.

For brevity we report the results attained for two generic and two video apps, i.e. Dropbox and Flipboard, and Snapchat and TikTok, respectively, and conducted by modeling $P = 3$ inputs (i.e. DIR, PL, and IAT) with GRU. Results are depicted in Figs. 13 and 14 and Figs. 15 and 16, for head and tail, respectively. Is is worth to underline that the index of packets starts from 2 because the second packet in the biflow is the first we are able to provide a prediction for (based on the very first packet of the same biflow). Consistently with previous analyses, the results are averaged over the 10-folds for each index or interval of indexes.

We highlight that the tail analysis requires also to properly filter the traffic data: to ensure that biflows are aligned on the last effective packet (viz. biflows are "complete") we leverage a simple heuristic based on the presence of FIN or RST TCP flags at the end of each biflow, by discarding biflows that do not contain any of these flags. Hence, we conservatively discard all the "incomplete" TCP biflows and the UDP biflows. Is worth to underline that this filtering phase results in discarding 6.59% of biflows, by considering the four analyzed apps on the whole; specifically, 10.31% of Dropbox, 3.64% of Flipboard, 16.38% of Snapchat, and 8.52% of TikTok.

Going into details, Fig. 13 shows the performance metrics of the two generic apps, namely Dropbox (on the left) and Flipboard (on the right). By looking at Figs. 13a and 13b, the prediction performance of DIR shows an application-specific behavior: Dropbox reports a slightly lower G-mean for the first 32 packets with respect to the remaining, whereas Flipboard DIR for the first 32 packets is easier to predict compared with the remaining. In general, focusing on the first 32 packets, it is evident that the G-mean is higher for DIR prediction of the first 10 packets, for both apps.

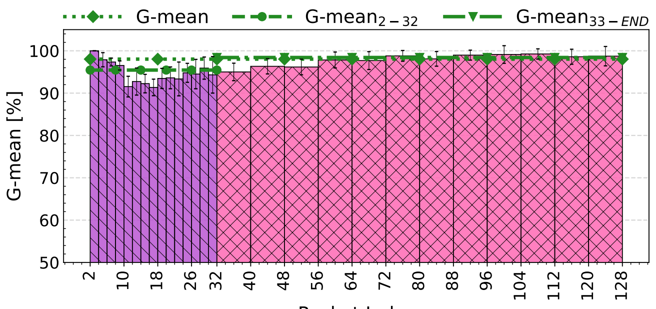
On the other hand, when considering PL and IAT prediction performance (shown in Figs. 13c and 13d, and in Figs. 13e and 13f, respectively) the behavior of the two apps is quite similar, with the RMSE on the first 32 packets being at least twice (viz. worse) than the RMSE for the packets in the rest of the biflow, i.e. the relative reduction of RMSE for the first 32 packets with respect to the remaining ones ranges from $\approx 50\%$ for Dropbox PL and Flipboard PL and IAT, to $\approx 75\%$ showed by Dropbox IAT. More in detail, focusing on the first 32 packets, those exhibiting the lowest RMSE on the PL are in the index interval $6 - 7$ for Dropbox, whereas the hardest PLs to predict are in the index interval $4 - 5$ for Flipboard. Also, for both apps the lowest RMSE on the IAT is achieved on the first 10 packets.

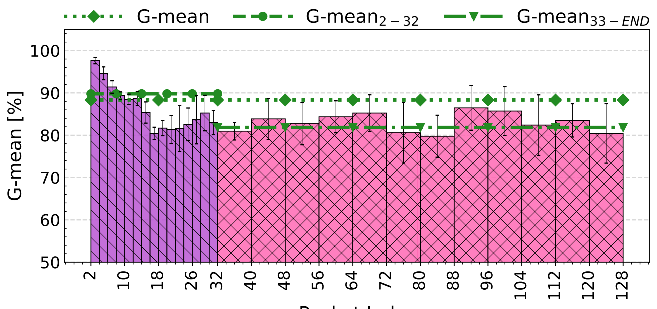
Figure 14 shows the results of the same analysis on video apps, namely Snapchat and TikTok. Firstly, we can notice that Fig. 14 confirms that TikTok is characterized by short biflows, i.e. up to 64 packets, in line with the characterization provided in Fig. 4c. Then, also in this case, when considering the G-mean of DIR, the behaviors vary with the specific app, with Snapchat (Fig. 14a) showing better performance on the first 32 packets and TikTok (Fig. 14b) reporting the opposite outcome. Conversely, when considering PL and IAT RMSE, the lowest errors are achieved for the packets beyond the 32nd. In particular, Snapchat PL presents an error peak for the first two packets.

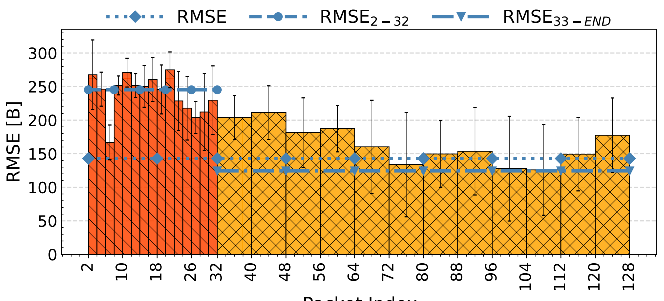
Finally, Figs. 15 and 16 analyze the per-packet index performance for biflow tails. Figure 15 highlights the degradation of performance starting from the second-to-last index, when considering DIR, PL, and IAT for both generic apps. Interestingly, this trend does not apply when considering video apps (Fig. 16), where the performance are quite balanced over the last 16 packets, with the exception of PL prediction performance on TikTok shown in Fig. 16d, where the RMSE trend becomes descending (corresponding to a performance improvement) starting from the second-to-last index.

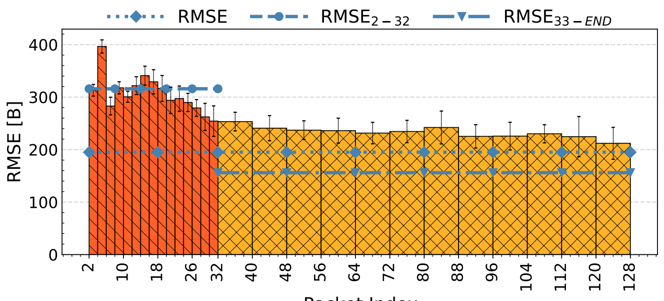
The present analysis highlights the need for adding additional information for the first 32 packets in order to fill the performance gap we find in RMSE of PL and IAT. To investigate this need, in Fig. 17, we illustrate the per-packet index performance referred to the GRU model trained with the addition of the PAY exogenous input, which we recall consists of the transport-layer payload of the first 32 packets. We underline that the nature of this input, as described in Sec. 3.2, mainly impacts the packets which fall within the interval of indexes $2 - 62$, because out of this interval the contribution of PAY input is of sole padding.[15] We focus on the same two video apps of previous per-packet index analyses, namely Snapchat (on the left) and TikTok (on the right). Firstly, we can observe only a negative impact on the DIR G-mean of the packets beyond the 32nd for TikTok,

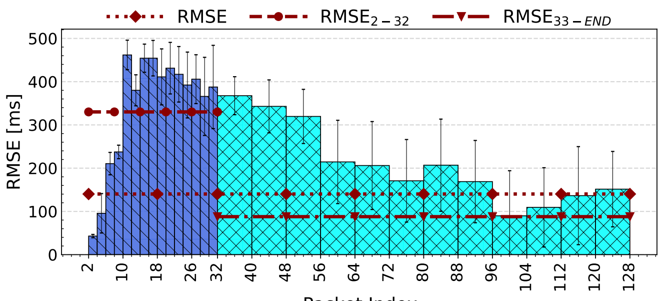
[15]Given a window-memory size of $W = 30$, the packets after the 62nd have a PAY exogenous input of sole padding.

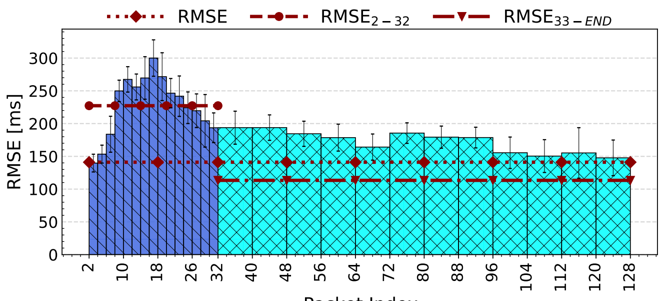

(a) `DIR` Dropbox.

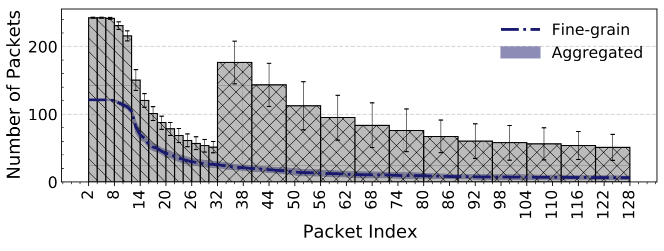

(b) `DIR` Flipboard.


(c) PL Dropbox.


(d) PL Flipboard.


(e) `IAT` Dropbox.


(f) `IAT` Flipboard.


(g) Per-index # packets Dropbox.

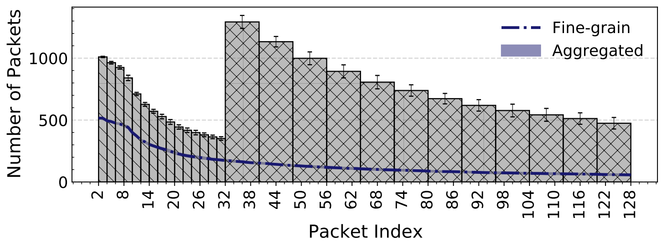
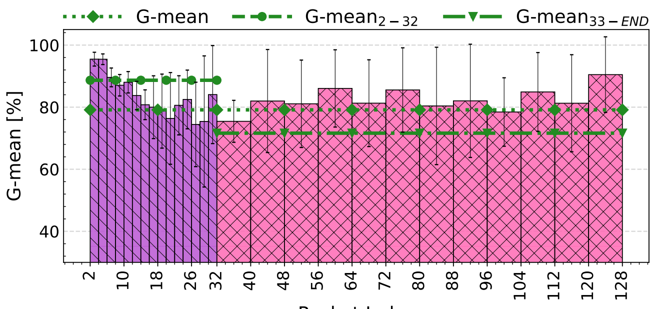
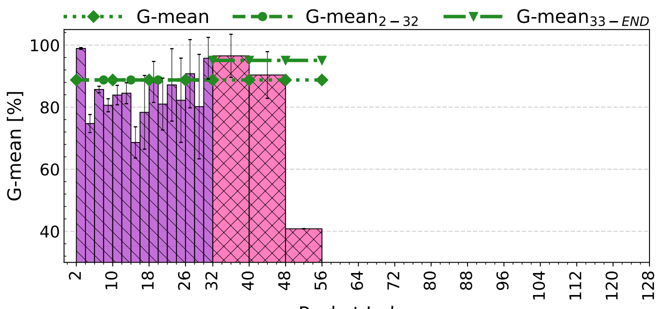
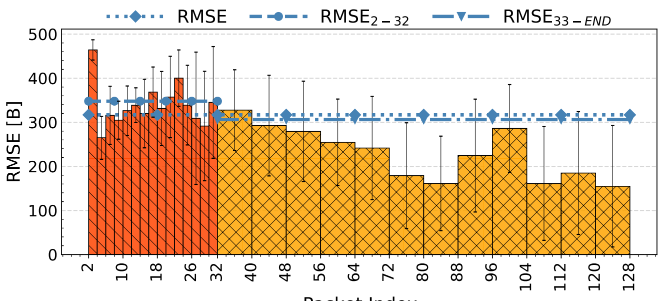
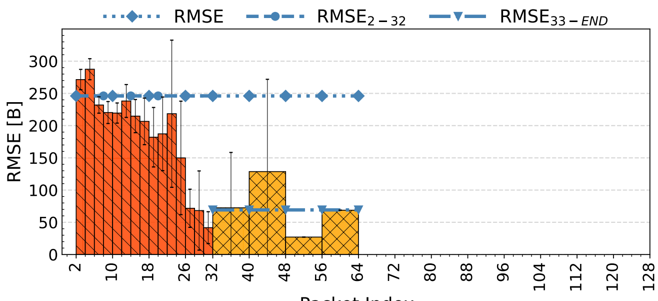

(h) Per-index # packets Flipboard.

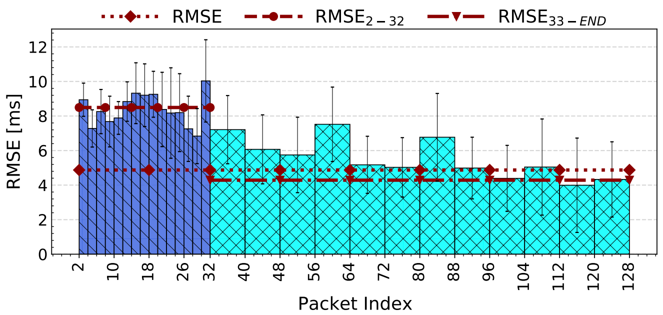
Figure 13: Per-packet index performance analysis for G-mean of `DIR` and RMSE of `PL` and `IAT` of the first 128 packets using a GRU model. The first 32 packets are aggregated with step 2, the remaining until the 128th with step 8. Horizontal lines report the overall RMSE, the RMSE of the first 32 packets (RMSE$_{2-32}$), and the RMSE of the remaining packets until the *end* of each biflow (RMSE$_{33-END}$). Analysis is conducted for two generic applications, namely `Dropbox` (on the left) and `Flipboard` (on the right). The last row shows the packet count over aggregates. Results are shown in the format *avg* ± *std* obtained over a 10-fold cross-validation procedure.

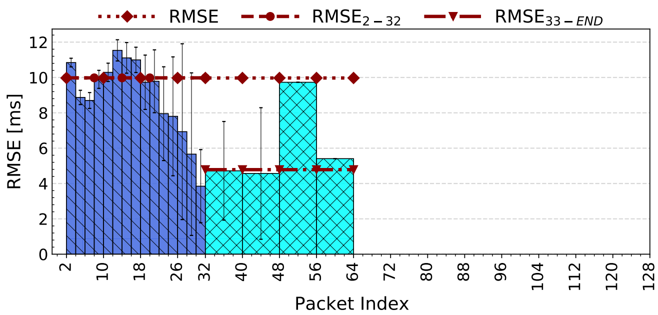

(a) Snapchat DIR.

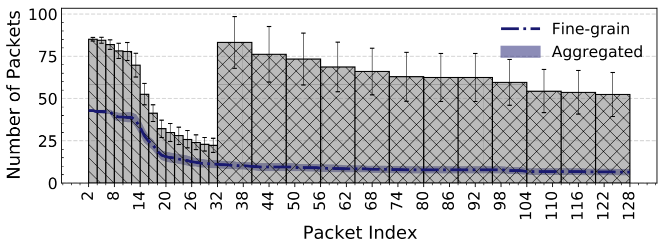

(b) TikTok DIR.


(c) Snapchat PL.


(d) TikTok PL.


(e) Snapchat IAT.


(f) TikTok IAT.


(g) Snapchat per-index # packets.

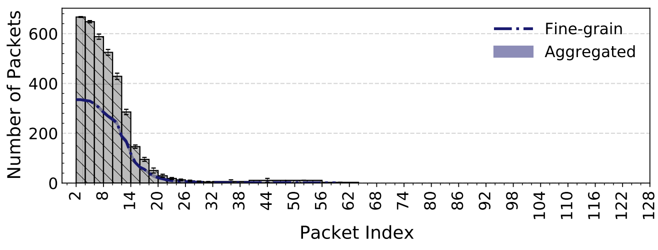
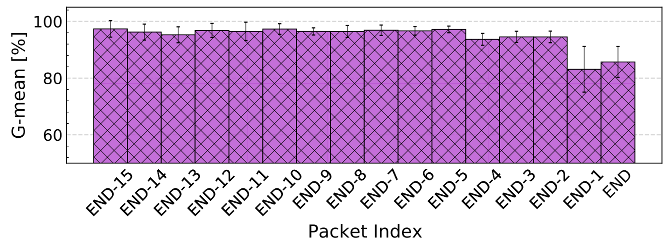
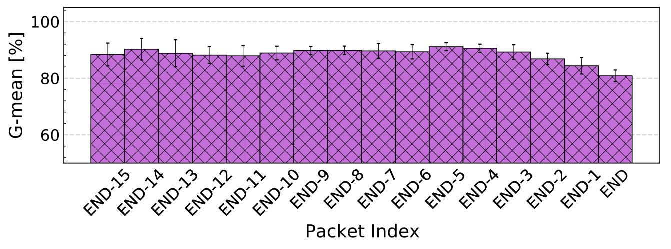

(h) TikTok per-index # packets Flipboard.

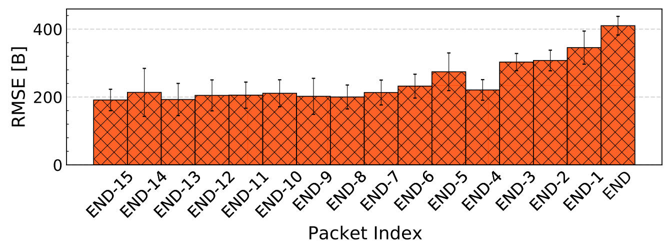
Figure 14: Per-packet-index performance analysis for G-mean of DIR and RMSE of PL and IAT of the first 128 packets using a GRU model. The first 32 packets are aggregated with step 2, the remaining until the 128th with step 8. Horizontal lines report the overall RMSE, the RMSE of the first 32 packets (RMSE_{2-32}), and the RMSE of the remaining packets until the *end* of each biflow (RMSE_{33-END}). Analysis is conducted for two video applications, namely `Snapchat` (on the left) and `TikTok` (on the right). The last row shows the packet count over aggregates. Results are shown in the format *avg* ± *std* obtained over a 10-fold cross-validation procedure.



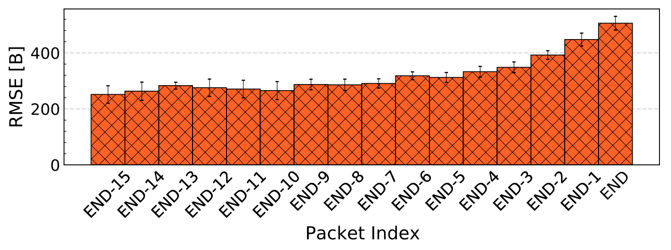
(a) DIR Dropbox.



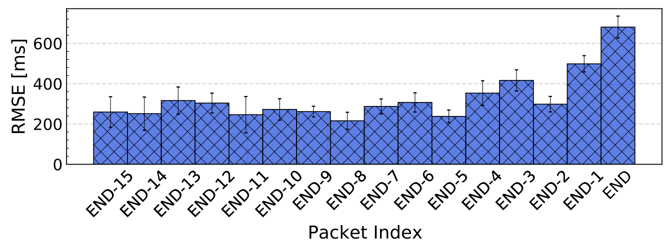
(b) DIR Flipboard.



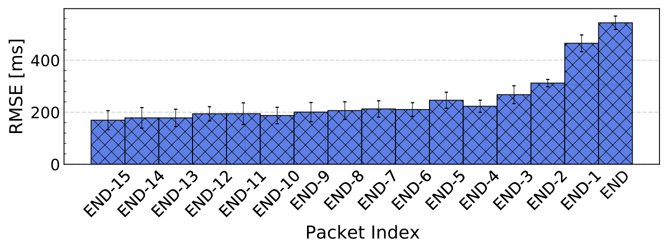
(c) PL Dropbox.



(d) PL Flipboard.



(e) IAT Dropbox.



(f) IAT Flipboard.

Figure 15: Per-packet-index performance analysis for G-mean of DIR and RMSE of PL and IAT of the last 16 packets without aggregation using a GRU model. Analysis is conducted for two generic applications, namely Dropbox (on the left) and Flipboard (on the right). Results are shown in the format *avg* ± *std* obtained over a 10-fold cross-validation procedure.

with an absolute drop < 5%. Conversely, the multiple positive effects of the exogenous input translate in: (*i*) ≈ 2% absolute gain on DIR G-mean of the first 32 packets when considering both Snapchat and TikTok; (*ii*) ≈ 10B and ≈ 35B absolute reduction of PL RMSE of the first 32 packets when considering Snapchat and TikTok, respectively; (*iii*) ≈ 1.5ms absolute reduction of IAT RMSE of the packets following the 32^{nd} when considering TikTok.

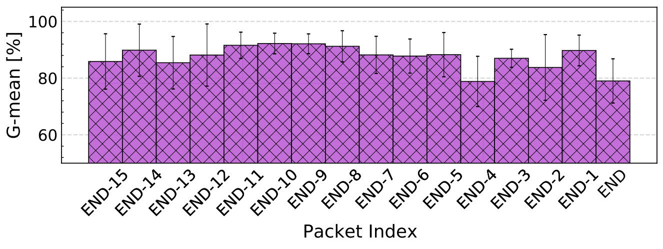
Main Remarks: Based on the investigated traffic, prediction performance typically improves after the 32^{nd} *packet, when predicting PL or IAT. Differently, concerning DIR prediction, app-dependent performance behaviors have been observed. On the other hand, focusing on the last* 16 *packets of each biflow, a decreasing performance trend is typically observed in correspondence of the very last packets, for all the parameters.*

5.7. Distillation Analysis

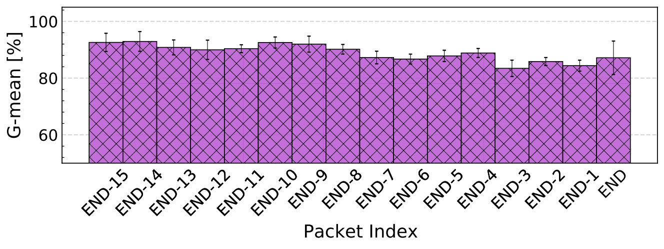
Finally, we carry out a *differential Markovian analysis* to directly compare the best multitask DL models against the best baselines belonging to ML and Markovian families, highlighting the differences in their behaviors, focusing on Dropbox (Fig. 18) and TikTok (Fig. 19) for brevity. To this aim we *distill* small-order Markov Chain models of the best baselines (i.e. high-order MC and RFR), compare them with that of the GRU (resp. SeriesNet), and analyze the discrepancies in their predictive behaviors on Dropbox (resp. TikTok).

In these figures we report the *difference of the distilled transition distributions* obtained for the two methods. Accordingly, in such a representation the values close to zero witness similar distributions, whereas positive (resp. negative) values report higher occurrences for the best baselines (resp. GRU) in Fig. 18. A similar reasoning applies to the distillation results in Fig. 19 when replacing the GRU with the SeriesNet.

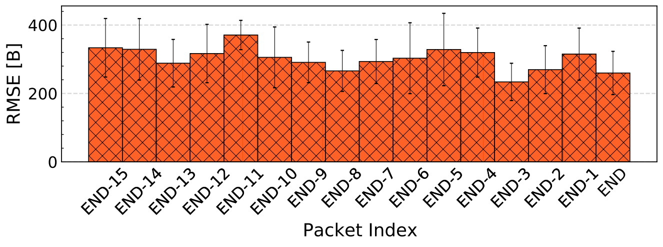
Considering DIR prediction on Dropbox, Fig. 18a shows that the strongest departure of GRU from the MC is for the “UDD” case (one upstream packet and two consecutive downstream packets), where the multitask GRU predicts an upstream packet with a probability 47% higher than MC. Quite significant are also the cases associated to “DDU” and “DUD”, where the multitask GRU predicts an upstream packet with a probability 33% and 31%, respectively, lower than MC. Conversely, the difference in the DIR predictive behavior between the multitask GRU and the RFR (shown in Fig. 18d) is more contained, with the highest difference also pertaining to “UDD” (as in the MC case), but with an upstream packet predicted with a probability



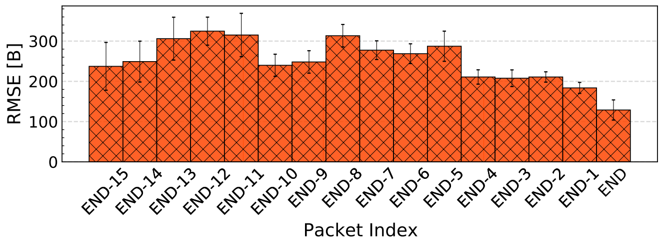
(a) Snapchat DIR.



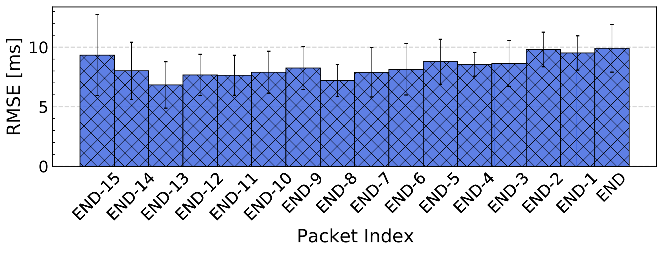
(b) TikTok DIR.



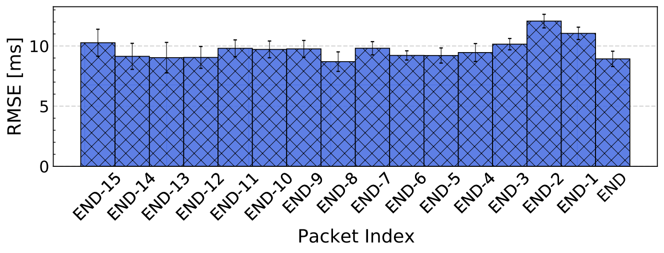
(c) Snapchat PL.



(d) TikTok PL.



(e) Snapchat IAT.



(f) TikTok IAT.

Figure 16: Per-packet index performance analysis for G-mean of DIR and RMSE of PL and IAT of the last 16 packets without aggregation, using a GRU model. Analysis is conducted for two video applications, namely Snapchat (on the left) and TikTok (on the right). Results are shown in the format *avg* ± *std* obtained over a 10-fold cross-validation procedure.

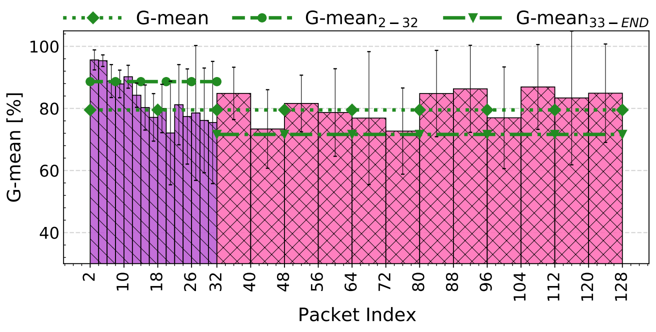
6.8% lower than the RFR. A similar situation can be observed in Figs. 19a and 19d, referring to TikTok. Indeed, the multitask SeriesNet has a predictive DIR behavior much more different than MC with respect to the RFR. More specifically, the strongest departure of SeriesNet from the MC is for the "UDU" case (alternating sequence ending with an upstream packet), where the multitask DL approach predicts an upstream packet with a probability 35.8% lower than the MC. Differently, when comparing SeriesNet behavior with RFR, the strongest departure corresponds to the "DDD" case, where the multitask DL approach predicts an upstream packet with a probability 11.9% higher than the RFR.

Concerning PL and IAT, when the specific multitask DL approach is compared with the RFR, there is an appreciable difference in predictive behavior of both traffic parameters. Indeed, the latter tends to predict next PL and IAT more frequently as the previous one, as highlighted by the main diagonal line in Figs. 18e, 18f, 19e, and 19f. This observation indeed applies to both GRU and SeriesNet on Dropbox and TikTok, respectively. On the other hand, for what it concerns the comparison of the multitask DL approach with MC, different behaviors can be observed. Specifically referring to PL (Fig. 18b), it is apparent on Dropbox that MC tends to predict more frequently close-to-zero values and a specific value corresponding to ≈ 700B, whereas for TikTok, MC PL (Fig. 19b) predictions are more clustered toward a cloud of low PL values. A similar behavior applies to IAT (Figs. 18c and 19c), where either low or very high values are more frequently predicted by MC, whereas the GRU/SeriesNet tend to predict consecutive similar IAT values.

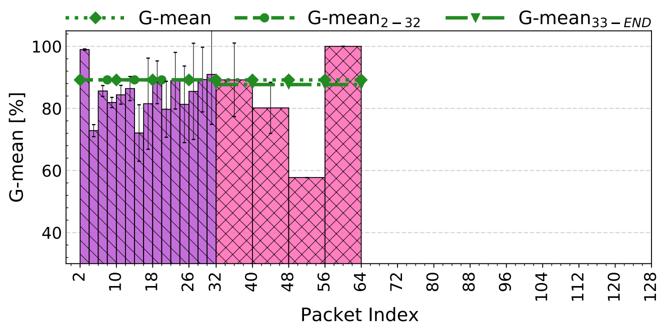
Main Remarks: The outcome of the distillation analysis—aimed at comparing the best DL model against RFR and high-order MC in terms of transitions distribution—can be summarized as follows. Concerning DIR, the best DL model is more similar to RFR than to MC. On the other hand, concerning PL and IAT, the behavior of the best DL model exposes remarkable differences w.r.t. both RFR and MC, whose peculiarities depend on the specific app.

6. Conclusions and Future Directions

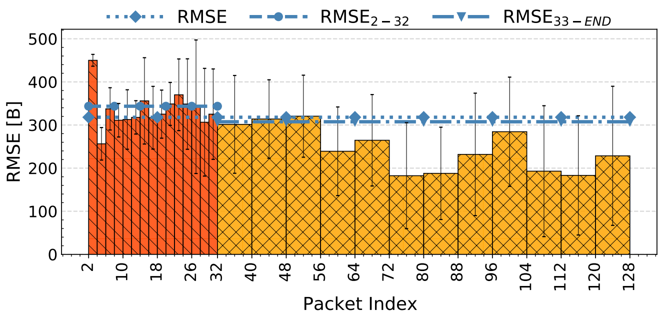
In this paper, we address the challenging problem of predicting the highly complex, dynamic, encrypted network traffic generated by mobile devices. In detail, we focus on packet-level prediction, leveraging two human-generated and recent datasets



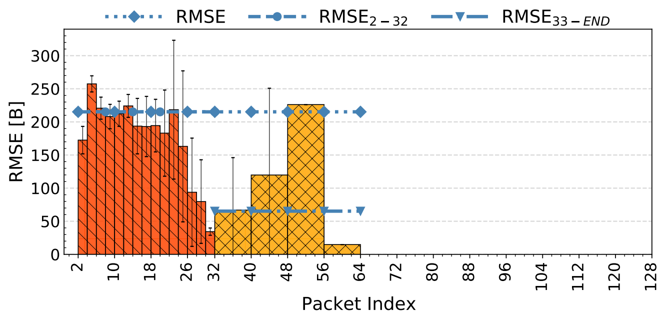
(a) Snapchat `DIR`.



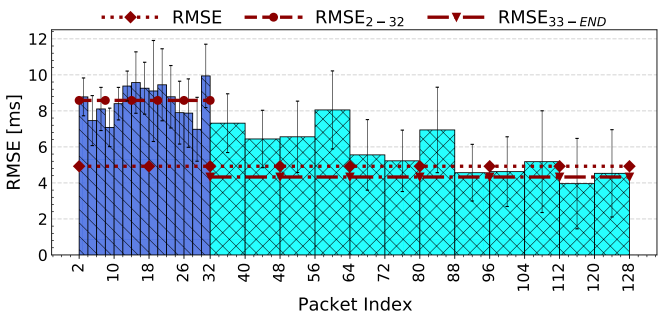
(b) TikTok `DIR`.



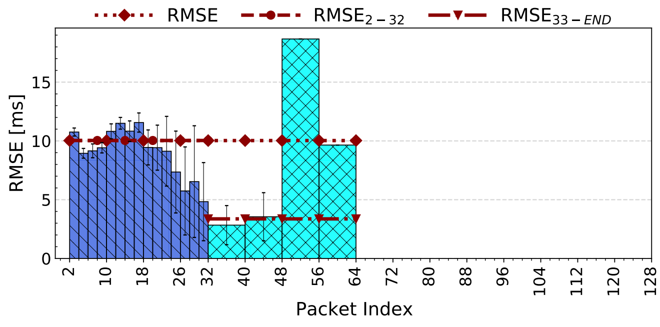
(c) Snapchat `PL`.



(d) TikTok `PL`.

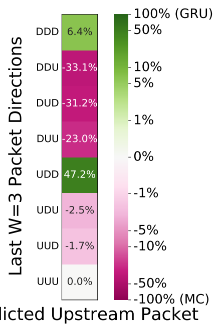


(e) Snapchat `IAT`.

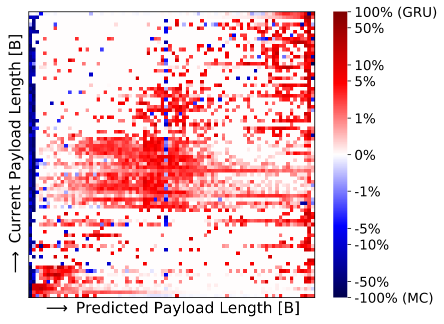


(f) TikTok `IAT`.

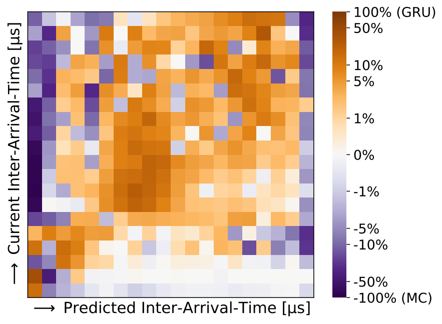
Figure 17: Per-packet index performance analysis for G-mean of `DIR` and RMSE of `PL` and `IAT` of the first 128 packets, when adding the *PAY* exogenous input to the GRU model. The first 32 packets are aggregated with step 2, the remaining until the 128*th* with step 8. Horizontal lines report the overall RMSE, the RMSE of the first 32 packets (RMSE$_{2-32}$), and the RMSE of the remaining packets until the *end* of each biflow (RMSE$_{33-END}$). Analysis is conducted for two video applications, namely `Snapchat` (on the left) and `TikTok` (on the right). The last row shows the packet count over aggregates. Results are shown in the format *avg* ± *std* obtained over a 10-fold cross-validation procedure.



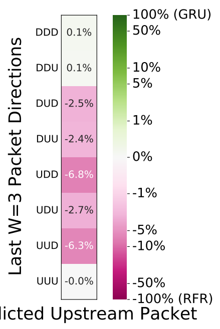
(a) 3rd-order MC distilled from `DIR` predictions.



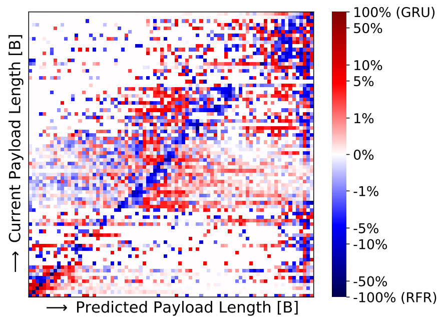
(b) 1st-order MC distilled from `PL` predictions.



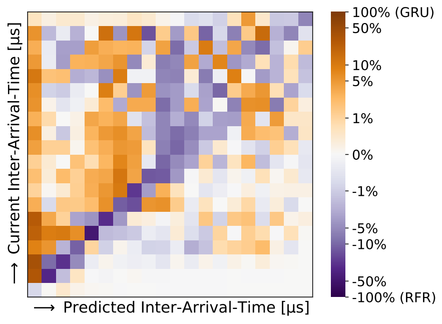
(c) 1st-order MC distilled from `IAT` predictions.



(d) 3rd-order MC distilled from `DIR` predictions.



(e) 1st-order MC distilled from `PL` predictions.

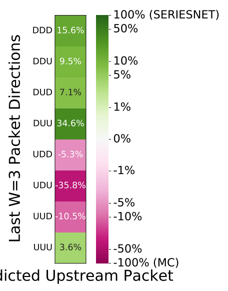


(f) 1st-order MC distilled from `IAT` predictions.

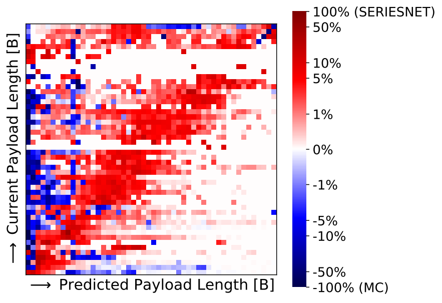
Figure 18: Differential Markov analysis of GRU predictive behavior against MC (top row) and RFR (bottom row) for the `Dropbox` application. The discrepancy is expressed through the *difference of the distilled* 3rd*-order transition distributions* for `DIR` (a), and 1st*-order transition distributions* for `PL` (b) and `IAT` (c). Positive values represent probabilities higher for SeriesNet when compared to MC or RFR ones, negative values vice-versa.

of mobile traffic, and using the methodological toolset of Deep Learning. One dataset, publicly released, includes generic mobile apps, while a newly-collected one (to be released) focuses on *video* apps belonging to different categories. Predicting the *finest-grain* features of network traffic (direction, payload length, and inter-arrival time of single packets) allows for the widest applicability, but is specifically challenging, also in relation to the considered traffic nature. Therefore, we resorted to a variety of DL approaches, exploring also different architectures (multitask and multimodal, besides single-task and single-modal ones). The experimental results are assessed varying the architecture (leveraging Markov Models, classic ML, and DL), the input representation (memory size and *exogenous* inputs), and considering different viewpoints (per-app and per-packet-position prediction performance, as well as model and training complexity). Using two differently-composed dataset, we assess the performance for both generic and video apps. The outcomes reveal notable variability in prediction performance among different apps and app categories (no silver bullet for this difficult problem). On average, (multitask) DL architectures outperformed the other ones in predicting packet direction (approximately +3% G-mean), and reported marginal performance degradation w.r.t. RFR when predicting payload length and inter-arrival time (+5 B and +8.7 ms RMSE, respectively). Per-packet analysis witnessed that the performance of the prediction of payload-length and inter-arrival time typically improved after the 32nd packet. Distillation analysis was helpful in interpreting and relating the behaviors of the models. The provided in-depth evaluation, besides confirming and quantifying in a solid methodological framework the heterogeneous behavior of mobile apps, provides valuable analysis tools to compare different predictors and strike the best balance among the different performance measures. The tested architectures proved to benefit from multi-modality, with multimodal-multitask architectures being specifically appealing when also considering complexity. In more detail, 30-sample memory allows to achieve the best performance in most of the cases. In general, higher complexity does not imply better performance. Last, it was observed that exogenous inputs are worth to be considered (with the addition of TCP window size and TCP flags to non-exogenous inputs being the most promising configuration in the considered examples) only for specific scenarios as their application does not remarkably enhance prediction performance, on average.

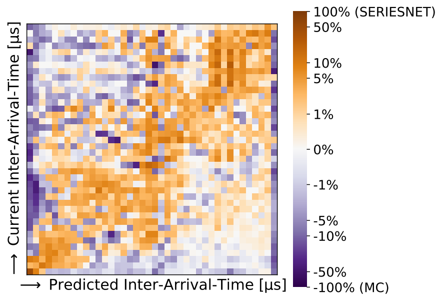
In future works, the effectiveness of the considered predictors will be assessed on other datasets and their underlying models exploited in the context of synthetic traffic generation. Additionally, we plan to investigate more sophisticated DL architectures—optimized via automatic tuning of hyperparameters—which can jointly exploit multitask learning and the natural multi-modality of network traffic data, possibly attempting to solve prediction and classification tasks simul-



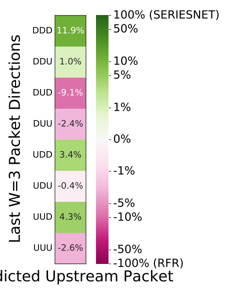
(a) 3^{rd}-order MC distilled from DIR predictions.



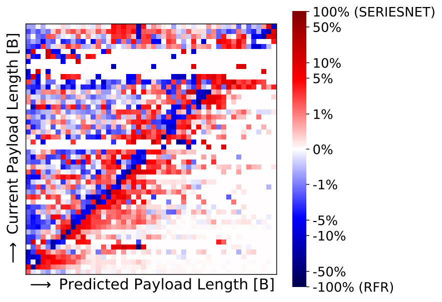
(b) 1^{st}-order MC distilled from PL predictions.



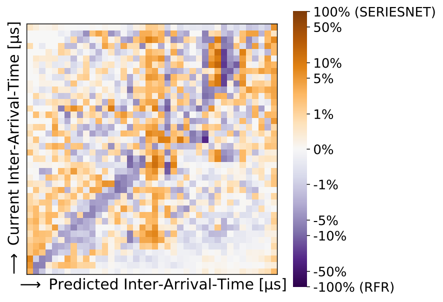
(c) 1^{st}-order MC distilled from IAT predictions.



(d) 3^{rd}-order MC distilled from DIR predictions.



(e) 1^{st}-order MC distilled from PL predictions.



(f) 1^{st}-order MC distilled from IAT predictions.

Figure 19: Differential Markov analysis of SeriesNet predictive behavior against MC (top row) and RFR (bottom row) for the TikTok application. The discrepancy is expressed through the *difference of the distilled* 3^{rd}*-order transition distributions* for DIR (a), and 1^{st}*-order transition distributions* for PL (b) and IAT (c). Positive values represent probabilities higher for SeriesNet when compared to MC or RFR ones, negative values vice-versa.

taneously. The analysis of biflow-level prediction at different time granularities is also of interest, as well as the investigation of predictors for longer future horizons. As the proposed approaches are designed and applied at the finest granularity (packet level), they are relevant to a wide spectrum of applications (e.g., traffic engineering, routing): thus, further work along these specific lines is envisioned as well. Finally, although the present work puts some contributions toward the interpretability of the considered DL architectures, the aided-design of predictors based on sophisticated explainable AI techniques is also foreseen [52].

References

[1] G. Aceto, A. Pescapè, Internet censorship detection: A survey, Computer Networks 83 (2015) 381 – 421. doi:10.1016/j.comnet.2015.03.008.

[2] J. Zhang, I. C. Paschalidis, Statistical anomaly detection via composite hypothesis testing for Markov models, IEEE Transactions on Signal Processing 66 (2017) 589–602. doi:10.1109/TSP.2017.2771722.

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

[4] F. Jejdling, et al., Ericsson mobility report, Ericsson AB, Business Area Networks, Stockholm, Sweden, Tech. Rep. EAB-20 9174 (2020).

[5] C. Huang, C. Chiang, 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. doi:10.1109/PIMRC.2017.8292737.

[6] C. Zhang, X. Ouyang, P. Patras, ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network, in: Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies, 2017, pp. 363–375. doi:10.1145/3143361.3143393.

[7] C. Zhang, P. Patras, Long-term mobile traffic forecasting using deep spatio-temporal neural networks, in: Proceedings of the Eighteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2018, pp. 231–240. doi:10.1145/3209582.3209606.

[8] C. Zhang, M. Fiore, P. Patras, Multi-service mobile traffic forecasting via convolutional long short-term memories, in: 2019 IEEE International Symposium on Measurements & Networking (M&N), IEEE, 2019, pp. 1–6. doi:10.1109/IWMN.2019.8804984.

[9] A. Dainotti, A. Pescapé, P. Salvo Rossi, F. Palmieri, G. Ventre, Internet traffic modeling by means of hidden Markov models, Computer Networks 52 (2008) 2645–2662. doi:10.1016/j.comnet.2008.05.004.

[10] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou, A roadmap for traffic engineering in sdn-openflow networks, Computer Networks 71 (2014) 1–30. doi:10.1016/j.comnet.2014.06.002.

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

[12] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. van Steen, A. Peter, FlowPrint: Semi-supervised mobile-app fingerprinting on encrypted network traffic, in: Network and Distributed System Security Symposium (NDSS), 2020. doi:10.14722/ndss.2020.24412.

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

[14] G. Aceto, G. Bovenzi, D. Ciuonzo, A. Montieri, V. Persico, A. Pescapè, Characterization and prediction of mobile-app traffic using Markov mod-

eling, IEEE Transactions on Network and Service Management, in press (2020). doi:10.1109/TNSM.2019.2899085.

[15] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, J. Domingo-Pascual, Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition, in: IEEE 7th International Conference on Intelligent Computer Communication and Processing (ICCP), 2011, pp. 95–102. doi:10.1109/ICCP.2011.6047849.

[16] C. Katris, S. Daskalaki, Comparing forecasting approaches for internet traffic, Expert Systems with Applications 42 (2015) 8172–8183. doi:10.1016/j.eswa.2015.06.029.

[17] T. P. Oliveira, J. S. Barbar, A. S. Soares, Computer network traffic prediction: a comparison between traditional and deep learning neural networks, International Journal of Big Data Intelligence 3 (2016) 28–37. doi:10.1504/IJBDI.2016.073903.

[18] S. Tanwir, D. Nayak, H. Perros, Modeling 3D video traffic using a Markov modulated gamma process, in: International Conference on Computing, Networking and Communications (ICNC), 2016, pp. 1–6. doi:10.1109/ICCNC.2016.7440638.

[19] A. Biernacki, Analysis and modelling of traffic produced by adaptive HTTP-based video, Multimedia Tools and Applications 76 (2017) 12347–12368. doi:10.1007/s11042-016-3623-8.

[20] X. Cao, Y. Zhong, Y. Zhou, J. Wang, C. Zhu, W. Zhang, Interactive temporal recurrent convolution network for traffic prediction in data centers, IEEE Access 6 (2017) 5276–5289. doi:10.1109/ACCESS.2017.2787696.

[21] A. Azzouni, G. Pujolle, NeuTM: A neural network-based framework for traffic matrix prediction in SDN, in: IEEE/IFIP Network Operations and Management Symposium (NOMS), 2018, pp. 1–5. doi:10.1109/NOMS.2018.8406199.

[22] A. Bayati, K. Khoa Nguyen, M. Cheriet, Multiple-step-ahead traffic prediction in high-speed networks, IEEE Communications Letters 22 (2018) 2447–2450. doi:10.1109/LCOMM.2018.2875747.

[23] A. Kalampogia, P. Koutsakis, H.264 and H.265 video bandwidth prediction, IEEE Transactions on Multimedia 20 (2018) 171–182. doi:10.1109/TMM.2017.2713642.

[24] N. Ramakrishnan, T. Soni, Network traffic prediction using recurrent neural networks, in: 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2018, pp. 187–193. doi:10.1109/ICMLA.2018.00035.

[25] J. Zhou, X. Yang, L. Sun, C. Han, F. Xiao, Network traffic prediction method based on improved echo state network, IEEE Access 6 (2018) 70625–70632. doi:10.1109/ACCESS.2018.2880272.

[26] D. Andreoletti, S. Troia, F. Musumeci, S. Giordano, G. Maier, M. Tornatore, Network traffic prediction based on diffusion convolutional recurrent neural networks, in: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2019, pp. 246–251. doi:10.1109/INFCOMW.2019.8845132.

[27] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, H. Zhang, Deep learning with long short-term memory for time series prediction, IEEE Communications Magazine 57 (2019) 114–119. doi:10.1109/MCOM.2019.1800155.

[28] Y. Huo, Y. Yan, D. Du, Z. Wang, Y. Zhang, 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), IEEE, 2019, pp. 1–4. doi:10.23919/APNOMS.2019.8892991.

[29] A. Lazaris, V. K. Prasanna, An LSTM framework for modeling network traffic, in: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE, 2019, pp. 19–24.

[30] V. A. Le, P. Le Nguyen, Y. Ji, Deep convolutional LSTM network-based traffic matrix prediction with partial information, in: IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2019, pp. 261–269.

[31] N. Segolene, K. Liao, W. Jiang, Improvement of the prediction-based energy efficient ethernet strategy, IEEE Access 7 (2019) 156420–156429. doi:10.1109/ACCESS.2019.2948840.

[32] Q. He, A. Moayyedi, G. Dán, G. P. Koudouridis, P. Tengkvist, A meta-learning scheme for adaptive short-term network traffic prediction, IEEE Journal on Selected Areas in Communications 38 (2020) 2271–2283. doi:10.1109/JSAC.2020.3000408.

[33] L. Nie, X. Wang, S. Wang, Z. Ning, M. Obaidat, B. Sadoun, S. Li, Network traffic prediction in industrial Internet of Things backbone networks: A multi-task learning mechanism, IEEE Transactions on Industrial Informatics (2021) 1–1. doi:10.1109/TII.2021.3050041.

[34] S. Waldmann, K. Miller, A. Wolisz, Traffic model for HTTP-based adaptive streaming, in: IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 683–688. doi:10.1109/INFCOMW.2017.8116459.

[35] S. Colonnese, P. Frossard, S. Rinauro, L. Rossi, G. Scarano, Joint source and sending rate modeling in adaptive video streaming, Signal Processing: Image Communication 28 (2013) 403–416. doi:10.1016/j.image.2013.01.007.

[36] S. Kang, S. Lee, Y. Won, B. Seong, On-line prediction of nonstationary variable-bit-rate video traffic, IEEE Transactions on Signal Processing 58 (2010) 1219–1237. doi:10.1109/TSP.2009.2035983.

[37] Yao Liang, Real-time VBR video traffic prediction for dynamic bandwidth allocation, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 34 (2004) 32–47. doi:10.1109/TSMCC.2003.818492.

[38] A. Rao, A. Legout, Y.-s. Lim, D. Towsley, C. Barakat, W. Dabbous, Network characteristics of video streaming traffic, in: 7th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT), 2011, pp. 1–12. doi:10.1145/2079296.2079321.

[39] D. Tsilimantos, T. Karagkioules, A. Nogales-Gómez, S. Valentin, Traffic profiling for mobile video streaming, in: IEEE International Conference on Communications (ICC), IEEE, 2017, pp. 1–7. doi:10.1109/ICC.2017.7996603.

[40] A. Biernacki, Improving quality of adaptive video by traffic prediction with (F)ARIMA models, Journal of Communications and Networks 19 (2017) 521–530. doi:10.1109/JCN.2017.000083.

[41] J. Du, C. Jiang, Y. Qian, Z. Han, Y. Ren, Resource allocation with video traffic prediction in cloud-based space systems, IEEE Transactions on Multimedia 18 (2016) 820–830. doi:10.1109/TMM.2016.2537781.

[42] R. Caruana, Multitask learning, Machine learning 28 (1997) 41–75. doi:10.1023/A:1007379606734.

[43] A. Dainotti, A. Pescapè, K. C. Claffy, Issues and future directions in traffic classification, IEEE Network 26 (2012) 35–40. doi:10.1109/MNET.2012.6135854.

[44] V. Losing, B. Hammer, H. Wersing, Incremental on-line learning: A review and comparison of state of the art algorithms, Neurocomputing 275 (2018) 1261 – 1274. doi:10.1016/j.neucom.2017.06.084.

[45] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapè, Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges, IEEE Transactions on Network and Service Management 16 (2019) 445–458. doi:10.1109/TNSM.2019.2899085.

[46] W. Wang, M. Zhu, J. Wang, X. Zeng, Z. Yang, End-to-end encrypted traffic classification with one-dimensional convolution neural networks, in: IEEE International Conference on Intelligence and Security Informatics (ISI), 2017, pp. 43–48. doi:10.1109/ISI.2017.8004872.

[47] S. Huang, D. Wang, X. Wu, A. Tang, DSANet: Dual self-attention network for multivariate time series forecasting, in: 28th ACM International Conference on Information and Knowledge Management (CIKM), 2019, pp. 2129–2132. doi:10.1145/3357384.3358132.

[48] Z. Shen, Y. Zhang, J. Lu, J. Xu, G. Xiao, SeriesNet: A generative time series forecasting model, in: IEEE International Joint Conference on Neural Networks (IJCNN), IEEE, 2018, pp. 1–8. doi:10.1109/IJCNN.2018.8489522.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: International Conference on Neural Information Processing Systems (NIPS), volume 30, Curran Associates, Inc., 2017, pp. 5998–6008.

[50] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).

[51] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL: http://arxiv.org/abs/1412.6980.

[52] H. Hagras, Toward human-understandable, explainable AI, IEEE Computer 51 (2018) 28–36. doi:10.1109/MC.2018.3620965.