Prediction of Mobile-App Network-Video-Traffic Aggregates using Multi-task Deep Learning

L. Pappone* F. Cerasuolo† V. Persico† D. Ciuonzo† A. Pescapé† F. Esposito*

*Saint Louis University †University of Napoli Federico II

***Abstract*—Traffic prediction has proven to be useful for several network management domains and represents one of the main enablers for instilling intelligence within future networks. Recent solutions have focused on predicting the behavior of traffic aggregates. Nonetheless, minimal attempts have tackled the prediction of mobile network traffic generated by different video application categories. To this end, in this work we apply Multi-task Deep Learning to predict network traffic aggregates generated by mobile video applications over short-term time scales. We investigate our approach leveraging state-of-art prediction models such as Convolutional Neural Networks, Gated Recurrent Unit, and Random Forest Regressor, showing some surprising results (e.g. $\mathrm{NRMSE} < 0.075$ for upstream packet count prediction while $\mathrm{NRMSE} < 0.15$ for the downstream counterpart), including some variability in prediction performance among the examined video application categories. Furthermore, we show that using smaller time intervals when predicting traffic aggregates may achieve better performances for specific traffic profiles.**

I. Introduction

Modeling network traffic is of utmost relevance for edge-cloud applications supporting softwarized edge infrastructures. Understanding traffic peculiarities helps manage different tasks, such as traffic engineering, risk minimization in network planning and provisioning, management of the Quality of Services/Experience, activity detection of profile users, anomaly detection, and even real traffic emulation for testing purposes. Nonetheless, this process is hindered by the increase of complexity and variability of the traffic that currently traverses networks. In particular, the global spread and growing usage of mobile devices have profoundly changed network traffic. For instance, according to a recent Ericsson Mobility Report, during 2019–2020 5G subscriptions increased to 220 million covering approximately 15% of world population and are expected to reach 8.8 billion by the end of 2026. The 2020 Mobile Internet Phenomena instead reported how video streaming applications are going to be the principal mobile traffic component, constituting the 65% of all downstream traffic. Traffic prediction techniques can be deployed to guarantee more reliable and efficient services, to optimize bandwidth resource allocation and to identify traffic anomalies. Predicting mobile network traffic has become even more challenging, given the high variability of traffic and network conditions, and the stringent QoS requirements of novel applications. Other challenges include traffic encryption, with the broad adoption of TLS or other protocols such as QUIC, mobile apps weekly updates, potential device/operating system/app-version diversity, and a lack of public datasets. To solve these challenges, several authors have recently proposed mobile traffic prediction techniques, with or without using machine learning, but mostly focusing on traffic aggregated in time, using several aggregation intervals, typically larger than one minute [1, 2, 3, 4]. As we detail in Sec. II, prior work has demonstrated the lack of a "panacea" method to deal with traffic prediction. This fact, along with the very large set of variables affecting learning models, makes the problem of mobile traffic aggregate prediction an interesting and important problem worth exploring further.

Our contribution. To this aim, in this paper we investigate the suitability of state-of-art Machine Learning (ML) and Deep Learning (DL) models to predict mobile video network traffic aggregates. In particular, we design a set of multi-task DL models to predict mobile-app network traffic aggregates and analyze their behavior when using different short-time aggregation windows with a time scale of the order of milliseconds. Given the mobile applications generating network traffic, we then report how the corresponding traffic profiles affects models performances. We found a strong dependency between ML/DL models and the specific mobile traffic profile, nevertheless showing promising performances when considering a finer-grain aggregation granularity.

The rest of the paper is organized as follows. In Sec. II we provide a background of related literature on (mobile-app) network traffic aggregates prediction. In Secs. III and IV we describe the proposed aggregated-traffic prediction methodology. We detail the considered experimental approach in Sec. V and discuss our conclusion in Sec. VI.

II. Related Work

Recent work aimed at predicting aggregated traffic features e.g., volume or network packet rates over time. Such aggregate prediction considered a diverse set of time-resolutions, ranging from time intervals smaller than one second in [5], to a few seconds in [6, 7], to minutes [4, 8, 9], or even hours and days in [4, 10, 11]. The majority of such existing prediction models has been designed for fairly large aggregation time windows [4, 8, 9, 11]. This is due to the challenges represented by the volatile nature of network traffic in small time scales and in part due to the lack of suitable datasets. Only a few recent authors have attempted to predict aggregated network traffic using a short-term prediction [5, 6, 7, 12], and only a small group predicted mobile network traffic [12], as we

ISBN 978-3-903176-48-5 ©2022 IFIP

do. For example, Lazaris and Prasanna [7] proposed an interesting approach tackling traffic aggregations using *jumping windows* [13] i.e., non-overlapping observation time intervals, showing, however, how the prediction is more accurate with larger time windows as the traffic variance decreases. Recently authors also proposed different aggregation techniques that deal with online traffic processing, optimizing algorithmic cost and complexity [14, 15, 16]. Such predictions utilized different features, spanning a variety of applications, e.g., traffic matrix estimation [6, 7], bandwidth estimation [5], and users demand prediction [12]. In this work, we analyze the behavior of DL models concerning the prediction of mobile network traffic aggregates. Differently to prior work in this area, our goal is to dissect the impact that the aggregation parameters have on the prediction of well-known aggregated features such as volume and count of packets. The use of recurrent neural networks to deal with traffic aggregate prediction problem has been also investigated, e.g., using LSTMs [7, 12, 17, 18], and GRUs [6], showing how each of them outperform statistical time series approaches such as ARIMA models. For example, Labonne et al. [5] compared performance of Deep Neural Networks (DNNs) and Random Forest Regression (RFR) algorithms in traffic prediction problem using aggregated CAIDA datasets and showing that RFR outperforms DL approaches in short-term bandwidth estimation, considering very short aggregation windows (less than 1s). As these papers, we also compare several state-of-art ML and DL approaches and their behavior when predicting mobile network traffic aggregates, but over short-time aggregation windows, and by analyzing how the traffic nature impacts on model performances. Based on real traffic traces generated from known mobile applications, we then identify the impact that specific traffic profiles have on models prediction performances.

III. Methodology

Short-term Traffic Prediction Problem. We consider the prediction of short-term network traffic aggregates generated by mobile applications as the objective of this work. We consider a bidirectional flow (*biflow*) as the elementary unit for the prediction task, which is defined by the classical 5-tuple (source IP, source port, destination IP, destination port, and transport-level protocol) including both directions of communication. Formally, an aggregate of network traffic consists of all the (biflow) packets whose arrival time τ_p (i.e., the timestamp of packet *p*) falls within an aggregation time interval of fixed duration $\Delta_M = (t_{n+1} - t_n), \forall n$, i.e., the interval $[t_n, t_{n+1}]$, where $t_n \triangleq n\Delta_M$ and Δ_M is in the order of milliseconds. For each biflow, all the packets are processed to compute aggregates according to the specific aggregation operation (e.g., number of bytes, packet rate, etc.). The aggregation process produces, for each feature, a time-series where each element is a traffic aggregate extracted from the n^{th} interval, denoted with $\boldsymbol{x}^n$. The predictions are performed based on the observation of a set of F_{in} features evaluated on the W most recent aggregation intervals, namely $[t_{n+1-W}, t_{n+2-W}], \ldots, [t_n, t_{n+1}]$. Accordingly, this choice implies a *memory window* of size W and a *memory time* equal to $T_M = W\Delta_M$, namely the overall amount of past time the model uses as input to predict the next aggregate. We consider T_M as an hyperparameter we set for our experiments. The features obtained by aggregation during the aforementioned intervals, denoted with $\boldsymbol{x}^n, ..., \boldsymbol{x}^{n-(W-1)}$, represent the input of the prediction problem. The aim of this work is to *predict* a set of F_{out} traffic features associated to a *future* aggregation interval $[t_{n+1}, t_{n+1} + T_P]$, where T_P denotes an application-specific *prediction horizon*. The features calculated in the prediction horizon (to be predicted) are grouped in the vector $\boldsymbol{y}^{n+1}$, which represents the (desired) outcome of the prediction problem. Accordingly, the prediction model is specified as

$$\boldsymbol{y}^{n+1} = \mathcal{M}(\boldsymbol{x}^n, \boldsymbol{x}^{n-1}, ..., \boldsymbol{x}^{n-(W-1)}) \tag{1}$$

We use a single (shared) multitask architecture that allows to solve multiple (and diversified) inference tasks at the same time. Our algorithm predicts the F_{out} aggregated traffic features via a *single* DL architecture. This approach constitutes a significant difference in terms of computational complexity with respect to a multiple single-task learning approach.

Traffic Aggregation and Preprocessing. Packet aggregation is the core step of our pre-processing phase. The sets of downstream and upstream packets of the generic biflow are denoted with P_{dw} and P_{up}, respectively. Biflow traffic packets are submitted to the aggregation process and the resulting input-output pairs will be used to feed the learning model. The aggregation is implemented using a window which encloses packets and produces the actual aggregated value. We implemented the aggregation process using a temporal *jumping window* of length Δ_M (i.e., the sliding unit of the window is equal to the window size). In this case, consecutive windows always include non-overlapping sets of packets. Therefore, Δ_M is defined as the aggregation *granularity*: each packet of each biflow that occurs within this time is properly aggregated according to the specific feature of interest. As a result of the process, for each Δ_M a single aggregate value is generated for each feature, so the resulting time-series will be generated for the single biflow. We aim at predicting $F_{out} = 4$ traffic aggregated features based on F_{in} traffic features evaluated on the previous W aggregation intervals. For simplicity, the nature and set of the input and output aggregated features coincide (thus $F_{in} = F_{out} = F$), and they differ only in the aggregation granularity. Specifically, for each biflow, the following input features are considered for the n^{th} interval:

- $b_i^n \triangleq \left\{ \sum_p b_p : \; t_n < \tau_p \leq t_{n+1} \; \wedge p \in P_i \right\}$
  $i \in \{dw, up\}$
  denotes the *downstream/upstream traffic volume*, i.e., the sum of application-layer bytes of downstream/upstream packets P_{dw} arrived during the aggregation time-interval. In the above formula, b_p denotes the number of application-layer bytes of packet p.
- $c_i^n \triangleq \left\{ \sum_p : \; t_n < \tau_p \leq t_{n+1} \; \wedge p \in P_i \right\}$
  $i \in \{dw, up\}$

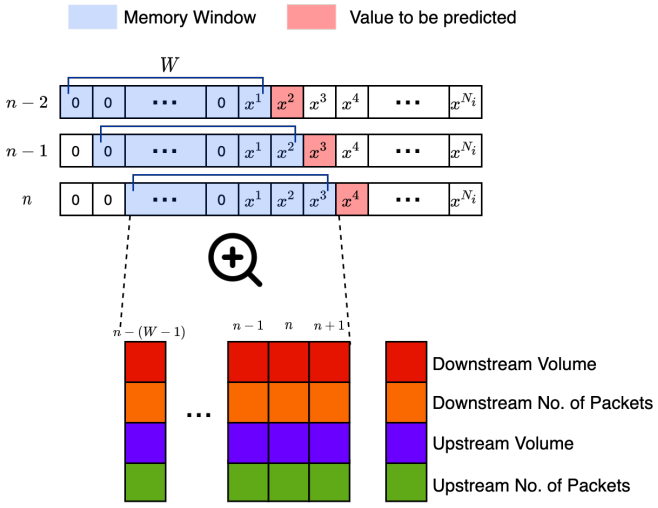


Fig. 1: Incremental windowing: $\boldsymbol{x}^n$ is the n^{th} vector element of the multivariate aggregated time series of length N_i, extracted from the biflow B_i; W = size of the memory window (input), $n+1$ = index of the vector value to be predicted (output).

denotes the *count of downstream/upstream packets* P_{dw} arrived during the aggregation time-interval.

The model is then trained to predict the value of the F_{out} features associated to the prediction horizon. Formally, the desired output vector is $\boldsymbol{y}^{n+1} \triangleq \{\bar{b}_{dw}^{n+1}, \bar{b}_{up}^{n+1}, \bar{c}_{dw}^{n+1}, \bar{c}_{up}^{n+1}\}$, where:

$$\bar{b}_i^{n+1} \triangleq \left\{ \sum_p b_p : \; t_{n+1} < \tau_p \leq (t_{n+1} + T_P) \; \wedge p \in P_i \right\},$$

$$\bar{c}_i^{n+1} \triangleq \left\{ \sum_p : \; t_{n+1} < \tau_p \leq (t_{n+1} + T_P) \; \wedge p \in P_i \right\},$$

$$i \in \{dw, up\}.$$

Such formulation includes the following *two different* cases:

- $T_P = \Delta_M$, i.e., the aggregation time-interval to predict next aggregate values is the same as the one used in memory. In such a case, it simply holds $\boldsymbol{y}^{n+1} = \boldsymbol{x}^{n+1}$;
- $T_P \neq \Delta_M$ i.e., the aggregation time-interval is different from the one used during the pre-processing. This corresponds to the general case $\boldsymbol{y}^{n+1} \neq \boldsymbol{x}^{n+1}$.

Prediction Strategy. Our approach is based on a jumping window of size W and unit stride, which is grown *incrementally* [19] (Fig. 1). Specifically, incrementally-sized sets of samples are grouped together until reaching the prescribed maximum size W of the prediction window (i.e., predictions can be made as soon as the first aggregate sample is available). That leads to model the *border effects*, (i.e., the early behaviours), which are achieved through a left *zero-padding* up to W samples. That is in contrast with a *fixed windowing* approach which considers predicting only the values that come after W actual samples. Therefore, for the generic time instant n, this procedure results in an input matrix $\boldsymbol{I}^n$ (of size $F \times W$, F being the number of input features with granularity Δ_m) and a prediction vector $\boldsymbol{y}^{n+1}$ (of size $F \times 1$, representing the desired output), constructed applying the windowing approach described above to each of the aggregated time-series within the considered set (e.g., those that are generated by the same app biflows) (Fig. 1). Accordingly, in the latter case, for the i^{th} aggregated time series, with corresponding length N_i, this procedure produces $(N_i - 1)$ prediction samples.

IV. Mobile-Network Traffic Modeling

Multitask DL approaches. *Convolutional Neural Networks (CNNs)* are a relevant example of DL architectures deriving from NNs and inspired by visual mechanism of living organism. CNNs were firstly devised to solve image recognition tasks, then successfully employed in natural language processing and recently in network traffic analysis.

Recurrent Neural Networks (RNNs) are a further deep NN family that have been widely leveraged in the field of traffic prediction. The reason of their usage is their suitability to handle time-series whose elements have temporal correlation. The main difference with respect to CNNs, is the presence of backward and forward connection between layers, also denoted as loops. The most common variants are *Long Short Term Memory* (LSTM) and *Gated Recurrent Unit* (GRU). These 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. In this work, we used a GRU optimized implementation [20] because it has a simpler structure than LSTM and therefore fewer parameters to train. The architecture considered are a CNN composed of two convolutional layers with 32 and 64 filters, respectively, and a GRU with 200 units, set based on state-of-art works [21, 22].

Multitask Loss Specification and Training Procedure. The training phase of DL models is performed following an iterative procedure when executing the stochastic gradient descent (first-order) optimization algorithm for minimizing a loss function. Training is performed leveraging a subset of N_B biflows associated with the considered app, which constitutes the training set $\mathcal{T}$. The latter is formally defined as

$$\boldsymbol{\mathcal{T}} = \bigcup_{i=1}^{N_B} \{\boldsymbol{I}^n(A_i), \boldsymbol{y}^{n+1}(A_i)\}_{n=1}^{N_i} \tag{2}$$

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 i^{th} aggregate time-series A_i (of length N_i). In what follows we denote with N the total number of samples within $\mathcal{T}$. We remark that the considered prediction strategy differs from those commonly employed in time-series forecasting, which leverage past observations to update the model or learn a specific parameter. Indeed the latter philosophy is practically infeasible in real-world (especially online) scenarios due to the complexity related to the fine-grained (aggregate) prediction task and the sophisticated prediction model considered. On the basis of these considerations, we have chosen the biflow-based cross-validation granularity. Furthermore, since the applied architectures leverage the

multitask learning approach, the minimization of the loss function depends on the specific parameter to predict (viz. the prediction task to address). We focus on the prediction of F parameters (viz. aggregated traffic features) of the next time-series aggregate collected in the vector $\boldsymbol{y}^{n+1}$ which represent the outputs of the DL architecture. Consequently, we intend to minimize a weighted sum of the losses of the F prediction tasks considered, namely:

$$\mathcal{L}\left(\boldsymbol{\theta}_{\text{shared}}, \{\boldsymbol{\theta}_f\}_{f=1}^{F}\right) \triangleq \sum_{f=1}^{F} \lambda_f \, \mathcal{L}_f\left(\boldsymbol{\theta}_{\text{shared}}, \boldsymbol{\theta}_f\right) \tag{3}$$

The weight λ_f represents the importance level of the f^{th} task in the multitask objective function to be optimized, for simplicity, we select the uniform weighting $\lambda_f = 1/F$, i.e., no specific preference is given to upstream/downstream volume/packet count. In the above equation $\boldsymbol{\theta}_{\text{shared}}$ collects the set of parameters associated with the layers shared by the different tasks, whereas $\boldsymbol{\theta}_f$ collects the parameters associated to the task-specific layers of the fth task. In detail, we aim to minimize the *MSE* loss for all the F prediction tasks, namely:

$$\boldsymbol{\mathcal{L}}_f^{\text{mse}}(\cdot) \triangleq \frac{1}{N} \sum_{i=1}^{N_B} \sum_{n=1}^{N_i - 1} (\hat{y}_f^{n+1}(A_i) - y_f^{n+1}(A_i))^2 \tag{4}$$

In the above equation, we recall that N denotes the overall number of training samples, and N_i the number of samples (viz. aggregates) of the i^{th} time-series A_i. Additionally, $y_f^{n+1}(A_i)$ denotes the f^{th} traffic parameter associated to the $(n+1)^{th}$ time window of the time-series A_i, whereas $\hat{y}_f^{n+1}(A_i)$ denotes the corresponding prediction.

Single-task ML baselines. Random Forest Regressor (RFR) represents a more complex implementation of the well-known decision trees-based ML algorithm. The RFR employs as estimators several decision trees. The number of these trees is a configurable model hyperparameter. As a result, trees constitute a forest, namely an ensemble of B decision trees. The forest is built at training time leveraging the so called "bootstrap aggregating" (bagging) and random-feature selection to mitigate over-fitting. Each tree is trained in order to minimize the MSE between the predictions and the actual values. RFR is employed for each of the $F = 4$ aggregated features to be predicted.

Implementation Aspects. We perform the optimization of the multitask DL architectures by employing the Adam optimizer 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. Each DL architecture is trained for 150 epochs. 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. Concerning the RFR algorithm, we consider $B = 100$ tree estimators in the forest, leaving the maximum depth of each tree as default.

V. Experimental evaluation

Dataset Preprocessing. We leveraged the *public dataset* MIRAGE-VIDEO which includes Wi-Fi traffic, collected by 280+ experimenters and generated by *eight* mobile video apps belonging to four custom video categories: cloud VR (`DiscoveryVR`, `FulldriveVR`), short video (`Instagram`, `TikTok`), video chat (`Messenger`, `Zoom`) and video on-demand (`Netflix`, `PrimeVideo`). Ranging from 15 to 80 minutes, the duration of each capture session depends on the type of the specific video app activity e.g., a sequence of one-minute-long short videos, or a two-hour-long movie on-demand.

We processed the dataset to aggregate raw packets constituting the biflows for each examined application. The related biflow data are then processed by the aggregation module which implements an incremental-window approach and provides the sequence of aggregate values for the four features considered. Subsequently, data are reshaped for the training phase and grouped in samples. Each sample constitutes the memory of the model of size W i.e., W traffic aggregates that will be learned, along with the related ground truth, represented by the next aggregate value. Data is then submitted to scaling operation to be ranged within $(0, 1)$. We evaluate prediction performances via a *ten-fold cross-validation* performing summary statistics i.e., average and standard deviation of the evaluation metrics considered.

In Figs. 2a and 2b we report the distributions of the packet count and packet rate per biflow for each application. Applications can be grouped according to similar distributions (not matching the corresponding video categories): `Netflix` and `DiscoveryVR` show a more intense downstream activity with respect to upstream (Fig. 2a) and also share a highly similar distribution of packet rates (Fig. 2b). Similarly, `Instagram`, `Messenger` and `Tiktok` have few packets per biflow in both directions, along with a very short biflow duration, resulting in low packet rates.

Evaluation Metrics. Model performance is evaluated using two variants of *Root Mean Squared Error (RMSE)*.

Per-biflow RMSE. RMSE is computed for each biflow, hence it is measured over the biflow sequence and averaged over the total number of biflow observations. The final *RMSE* will be the average of the errors resulting from each biflow.

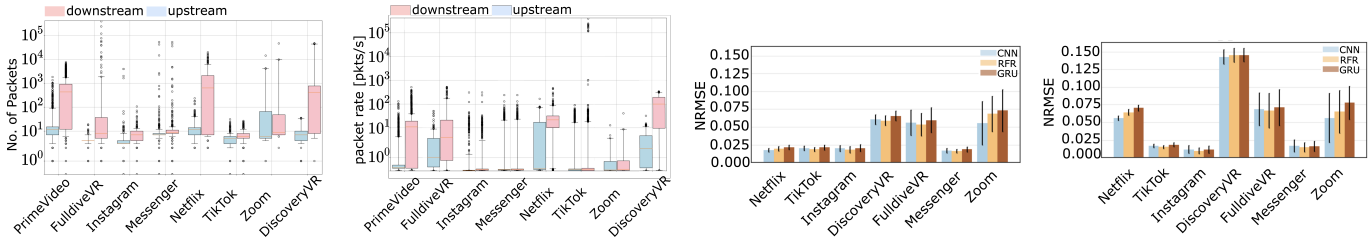
$$\text{RMSE}_m \triangleq \frac{\sum_{t=0}^{T} \sqrt{\frac{1}{N_t} \sum_{n=0}^{N_t - 1} [\hat{x}_m^t(n+1) - x_m^t(n+1)]^2}}{T} \tag{5}$$

where N_t denotes the number of predictions from the t^{th} biflow, $x_m^t(n)$ the sequence of values of the m^{th} feature observed from the t^{th} biflow, $\hat{x}_m^t(n)$ the corresponding sequence of values provided by the prediction model and T the total number of biflows.

Normalized RMSE (NRMSE): it is a useful variant that normalizes the *RMSE* by the ground truth range of values.

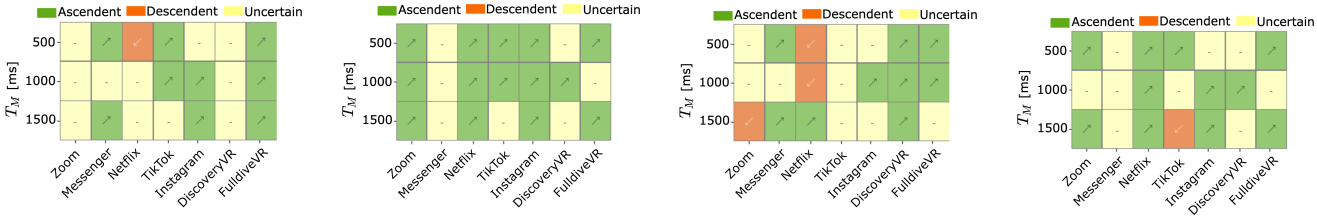
$$\text{NRMSE}_m \triangleq \text{RMSE}_m \,/\, (\max(x_m) - \min(x_m)) \tag{6}$$

where RMSE_m denotes the *RMSE* computed for the m^{th} feature and x_m the ground truth sequence observed (i.e., the entire training set).



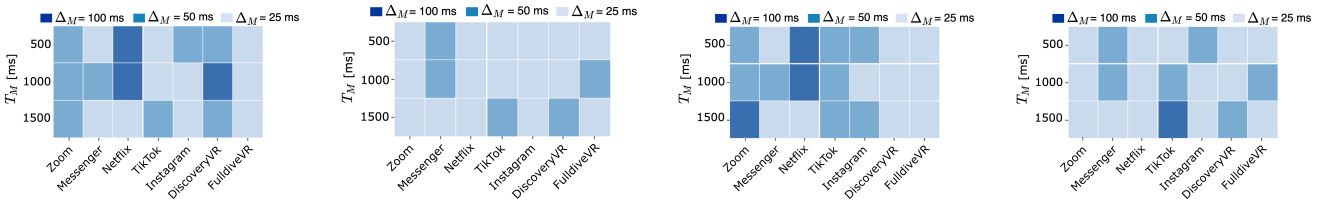
(a) Number of packets (b) Packet rate (c) Upstream Packet Count. (d) Downstream Packet Count.

Fig. 2: Distributions of the count of packets (a) and packet rate (b) per biflow in both down/up directions for each application, where whiskers denote the 5^{th} and 95^{th} percentile, whereas the box limits represent 25^{th} and 75^{th} percentile. Performance of models in terms of NRMSE for upstream (c) and downstream (d) count of packets, with $\Delta_M = 100$ ms and $T_M = 500$ ms.



(a) Upstream Packet Count. (b) Downstream Packet Count. (c) Upstream Volume. (d) Downstream Volume.

Fig. 3: Performance trend *when lowering* Δ_M for different memory time (T_M) values and $T_P = 100$ ms. Ascendent performance indicates lower (better) RMSE.



(a) Upstream Packet Count. (b) Downstream Packet Count. (c) Upstream Volume. (d) Downstream Volume.

Fig. 4: Aggregation interval (Δ_M) attaining lowest (optimum) RMSE for different memory time (T_M) values; $T_P = 100$ ms.

Prediction Results and Model Comparison. In Figs. 2c and 2d models performances are compared, fixing $\Delta_M = 100\,\text{ms}$ and using $T_M = 500\,\text{ms}$. For the applications[1] with a low packet rate and few packets per biflow (i.e., `TikTok`, `Instagram`, `Messenger`), the error is lower with respect to those with a high packet rate and a denser downstream activity (i.e., `Netflix`, `DiscoveryVR`). These results show that our considered models lead to higher accuracy when the traffic profile has a low packet rate. Each traffic inactivity period corresponds to a sequence of zero-values. The denser is the activity, the higher is the prediction error for each model. CNN outperforms the other models for `Netflix`, `DiscoveryVR` and `Zoom` and for the downstream features (e.g. $\approx +30\%$ w.r.t. GRU and RFR for `Zoom`), showing that the model is more accurate with a denser traffic activity. RFR outperforms the others for the applications with low packet rates (e.g. $\approx +2\%$ for `Tik Tok`), as GRU and CNN achieve similar overall performances.

In the following analysis, we examine the existence of trends occurring when considering smaller granularities in the traffic aggregation process. We leverage the CNN model for this analysis since it requires the least computational time during the training phase.We found that the reduction of Δ_M leads to an improvement of the prediction performance, namely a decrease of the error metric considered. We repeat the experiments for different values of the memory time T_M to understand how the number of aggregate samples in the model memory impacts the prediction accuracy. In Fig. 3 we show the heatmap obtained for different combinations of T_M and Δ_M with T_P fixed at $100\,\text{ms}$. The results show that, specifically for downstream traffic, lowering Δ_M leads to a decreasing error trend. For the packet count, almost all applications show a decreasing RMSE trend independently from the T_M value selected. Additionally, the downstream packet count shows the highest number of decreasing error trends, showing that its prediction benefits from adopting a

[1]In what follows, we omit performance for `PrimeVideo` due to large evaluation time required.

smaller Δ_M regardless of the application and the memory time T_M. For the other features, the memory time T_M affects the performances differently for each application and each feature. Overall, downstream traffic presents a more significant number of decreasing RMSE trends than upstream (+16.6%) and shows that a finer granularity allows for a more accurate pattern discovery in the presence of denser traffic activity. In contrast, the applications with less traffic activity are also less sensitive to the granularity value. Finally, we investigate the Δ_M value producing the lowest error for each value of the memory time T_M. Fig. 4 reports the granularity through which the best result is achieved. As a result, we found that for $T_P = 100\,\mathrm{ms}$ and downstream features, the results obtained from our experiments with $\Delta_M = 25\,\mathrm{ms}$ outperform those with larger aggregation granularities, showing that predicting aggregates in the order of milliseconds when traffic activity is relatively denser leads to better prediction accuracy. Further, we found that the memory time affects performance, see e.g., `TikTok` downstream volume and `DiscoveryVR` upstream packet count. Concerning the upstream features, we found that the best granularity strongly depends on the combination of application, T_M and Δ_M.

VI. Conclusion

In this work, we have investigated modeling and prediction of mobile-app traffic. Multiple predicted features together with the short-term prediction horizon considered allow us to comprehend the finest variation of the traffic in relation to the specific application considered, thus representing a paramount tool in edge-cloud network management tasks. We have accurately examined the suitability of DL models to the prediction of traffic aggregates, using a Multi-Task Learning approach. Our experimental analyses—involving real user-generated datasets—have shown that mobile traffic is particularly irregular and unbalanced when considering upstream and downstream directions, hence putting to the test the state-of-art models that recently has been widely used in the context of traffic prediction and classification. We found that the specific input application notably impacts prediction performance, in relation to traffic profiles and users' activity. Nonetheless, regarding the employment of finer aggregation granularity, the prediction error can be lowered as the models can learn patterns that are unrevealed when larger aggregation is carried out. As future work, we want to extend our analysis on other public datasets for mobile traffic.

Acknowledgment

The work of Lorenzo Pappone and Flavio Esposito has been supported by NSF # 1836906 and # 1908574.

References

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

[2] C. Zhang and P. Patras, "Long-term mobile traffic forecasting using deep spatio-temporal neural networks," in *18th ACM Mobihoc*, 2018, pp. 231–240.

[3] C. Zhang, X. Ouyang, and P. Patras, "ZipNet-GAN: Inferring fine-grained mobile traffic patterns via a generative adversarial neural network," in *13th ACM CoNEXT*, 2017, pp. 363–375.

[4] C.-W. Huang, C.-T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *IEEE 28th PIMRC*, 2017, pp. 1–6.

[5] M. Labonne, J. López, C. Poletti, and J.-B. Munier, "Short-term flow-based bandwidth forecasting using machine learning," *arXiv preprint arXiv:2011.14421*, 2020.

[6] N. Ramakrishnan and T. Soni, "Network traffic prediction using recurrent neural networks," in *17th IEEE ICMLA*, 2018, pp. 187–193.

[7] A. Lazaris and V. K. Prasanna, "Deep learning models for aggregated network traffic prediction," in *15th IEEE CNSM*, 2019, pp. 1–5.

[8] A. Bayati, K. Khoa Nguyen, and M. Cheriet, "Multiple-step-ahead traffic prediction in high-speed networks," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2447–2450, 2018.

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

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

[11] Y. Huo, Y. Yan, D. Du, Z. Wang, Y. Zhang, and Y. Yang, "Long-term span traffic prediction model based on STL decomposition and LSTM," in *IEEE 20th APNOMS*, 2019, pp. 1–4.

[12] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using LSTM networks," in *IEEE 29th PIMRC*, 2018, pp. 1827–1832.

[13] J. Traub, P. M. Grulich, A. R. Cuéllar, S. Breß, A. Katsifodimos, T. Rabl, and V. Markl, "Efficient window aggregation with general stream slicing." in *EDBT*, 2019, pp. 97–108.

[14] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[15] E. Viegas, A. Santin, A. Bessani, and N. Neves, "Bigflow: Real-time and reliable anomaly-based intrusion detection for high-speed networks," *Future Generation Computer Systems*, vol. 93, pp. 473–485, 2019.

[16] W. Iqbal, J. L. Berral, D. Carrera *et al.*, "Adaptive sliding windows for improved estimation of data center resource utilization," *Future Generation Computer Systems*, vol. 104, pp. 212–224, 2020.

[17] A. Azzouni and G. Pujolle, "NeuTM: a neural network-based framework for traffic matrix prediction in SDN," in *IEEE/IFIP NOMS*, 2018, pp. 1–5.

[18] A. Lazaris and V. K. Prasanna, "An LSTM framework for modeling network traffic," in *IFIP/IEEE IM*, 2019, pp. 19–24.

[19] G. Aceto, G. Bovenzi, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapé, "Characterization and prediction of mobile-app traffic using Markov modeling," *IEEE Trans. Netw. Serv. Manag.*, pp. 1–1, 2021.

[20] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[21] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *IEEE ISI*, 2017, pp. 43–48.

[22] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 2, pp. 445–458, 2019.