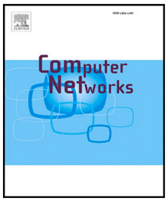
DeX: Deep learning-based throughput prediction for real-time communications with emphasis on traffic eXtremes

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



DeX: Deep learning-based throughput prediction for real-time communications with emphasis on traffic eXtremes

Tailai Song [*], Paolo Garza, Michela Meo, Maurizio M. Munafò

Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 TO, Torino, Italy

ARTICLE INFO

Keywords:
Real-time communications
RTP
Throughput prediction
Packet level
Machine learning
Deep learning

ABSTRACT

Recent years have witnessed a remarkable upsurge in the global proliferation of Real-Time Communications (RTC) applications, a trend propelled by the flourishing advancement of network technologies and further amplified by the COVID-19 pandemic. Within this context, there is a burgeoning interest in the innovation of sophisticated and intelligent network infrastructures and technologies. Positioned as a promising candidate for this purpose, real-time throughput prediction emerges as a key enabler to foster network observability and offer proactive functions, upholding advanced system management, including but not limited to, bandwidth allocation and adaptive streaming. Nonetheless, existing methodologies struggle with predicting extreme conditions of throughput, notably peaks, valleys, and abrupt changes, that are critical in RTC traffic. To surmount these obstacles, we introduce *DeX*, a Deep Learning (DL)-based framework, designed to predict short-term throughput, with a *dex*terous proficiency and dedicated focus on navigating the complexities of traffic *eX*tremes.

In particular, *DeX* leverages solely packet-level information as features and is composed of three integral components: a packet selection module that opts for an optimal subset of input features, a feature extraction block that partially incorporates the Transformer architecture, and a multi-task learning pipeline that improves the proficiency in handling traffic extremes. Moreover, our work is anchored in extensive traffic traces garnered during actual video-teleconferencing calls, and we formulate a time-series regression problem, rigorously evaluating a spectrum of technologies ranging from an adaptive filter to diverse Machine Learning (ML) and DL approaches. Initially, we aim at predicting throughput within 500-ms time windows using historical 1024 packets out of 2048, and consequently, our methodology exhibits exceptional efficacy, especially in forecasting traffic extremities. Conclusively, we conduct a series of ablation experiments and thorough analyses to showcase the enhanced performance of various scenarios, further validating the effectiveness and robustness of *DeX*.

1. Introduction

In the modern era, Real-Time Transport Protocol (RTP) [1]-based Real-Time Communications (RTC) have entrenched themselves as indispensable instruments across both professional and recreational domains, ushering in a suite of applications such as video-teleconferencing, online gaming, streaming, etc. The unprecedented popularity of RTC applications throughout recent years can be attributable to the heightened demand for entertainment and enhanced lifestyles in the post-pandemic period, in tandem with the global shift towards remote work practices [2]. Presently, consumers are confronted with an abundance of rival applications [3] driven by perpetual enrichment of RTC services, which can be ascribed to the augmented accessibility of bandwidth, the extensive growth of network infrastructures, and the advent of cutting-edge 5G technologies. This expansion leads to a diversification of user preferences and expectations, necessitating the implementation of advanced and efficacious optimizations in RTC systems to deliver an unparalleled user experience in various contexts. Contemporary users seek not only high-quality audio and video but also a seamless and fluid overall communication experience. Catering to these intensified expectations requires multifaceted and innovative approaches that transcend the scope of traditional solutions.

To this end, there is a compelling necessity to develop and refine robust, intelligent, and scalable technologies aimed at augmenting network performance and Quality of Experience (QoE). Notably, bandwidth management assumes a pivotal role in RTC, incorporating essential functionalities such as throughput measurement, bandwidth

[*] Corresponding author.
E-mail addresses: tailai.song@polito.it (T. Song), paolo.garza@polito.it (P. Garza), michela.meo@polito.it (M. Meo), maurizio.munafo@polito.it (M.M. Munafò).

https://doi.org/10.1016/j.comnet.2024.110507
Received 29 January 2024; Received in revised form 23 March 2024; Accepted 12 May 2024
Available online 16 May 2024

allocation, dynamic transmission adjustments, and traffic prioritization [4–7]. In light of this, the prediction of traffic throughput holds immense potential, proffering a preemptive mechanism that confers manifold advantages: (i) Optimized bandwidth allocation and utilization are attainable through accurate throughput estimations, thereby avoiding both underutilization and over-provisioning of network resources; (ii) Adaptive streaming and transcoding can increase QoE by dynamically modifying media quality, resolution, or encoding settings in alignment with the anticipated bitrate, ensuring optimal content dissemination; (iii) Network congestion management can be effectively executed through the forecasting of bandwidth requirements, facilitating proactive countermeasures such as traffic shaping, prioritization, or rerouting to alleviate or avert congestion issues; (iv) Resource planning becomes more efficient as service providers and network operators utilize predicted throughput information to assess and strategically allocate the required network resources, guaranteeing scalability and consistent levels of service quality to accommodate the anticipated communication demands. Nevertheless, throughput prediction poses formidable challenges, particularly within the context of RTC, owing to dynamic, ever-changing, and heterogeneous nature of networks, constrained computational capacity, and possible temporal limitations. Compounding this, existing solutions for common time-series problems often struggle with the prediction of extreme conditions, which are of critical significance in RTC traffic.

In this paper, we present *DeX*, an innovative *D*eep Learning (DL) Neural Network (NN), meticulously tailored to predict the throughput of RTC traffic, with emphasis on traffic *eX*tremes, namely **peak values**, **valley values**, and **abrupt changes**. *DeX* strategically and exclusively capitalizes on packet-level information, providing the benefit of minimal extraction efforts while endeavoring to address inherent challenges in the problem. Specifically, *DeX* is characterized by a tripartite structure, comprising three multifunctional and synergistic components: a packet selection module that autonomously and intelligently filters an optimal subset of input packets, aiming to curtail feature quantity and model complexity; a Transformer [8]-based feature extraction block that employs the multi-head attention mechanism to discern the dynamic and intrinsic network traffic patterns; and a multi-task learning pipeline with various weights that enhances the regression problem by integrating two supplementary tasks to effectively adapt to traffic extremities.

Our work is underpinned by a substantial collection of real video-conferencing traffic, collected from client sides across various network environments, and the dataset is curated with both historical throughput time series and packet-level features. We articulate a regression problem and benchmark our model against a simple baseline and an array of prevalent techniques, spanning from an adaptive filter to conventional ML and DL approaches. Initially, we select 1024 packets from a set of 2048 to predict short-term throughput in forthcoming time windows of 500 ms, and subsequently, we undertake a series of ablation studies to ascertain the importance of different components, accompanied by comprehensive analyses to explore diverse scenarios and expatiate the operational mechanics of *DeX*. Moreover, our proposed solution with a streamlined architecture is envisioned to achieve computational efficiency and reduce processing time consumption, and to be integrated as a software module for end-users or network equipment such as media servers, establishing an AI-based, RTC-aware, comprehensive, and proactive system for traffic monitoring and management. It facilitates application-level observability within the network control plane, thus empowering efficient and informed decision-making processes, and incorporates a feedback mechanism to rapidly respond to fluctuating network conditions. To summarize, our contribution is characterized by the following key aspects:

- A three-component DL framework, named *DeX*, optimized for accurate throughput forecasting, especially suited for handling traffic extremes.
- A dataset encompassing packet-level details of RTP-based traffic traces, collected from real video calls utilizing common RTC applications.
- A series of analyses coupled with model interpretation for our proposed solution that justifies the model's effectiveness and generalizability.

The remainder of this paper is organized as follows. Section 2 provides the motivation and formulates the problem, while Section 3 describes the dataset employed in our work. Afterwards, we delineate the architecture and functionalities of *DeX* in Section 4, and present the experimental outcomes in Section 5. Moreover, we perform five ablation tests in Section 6, followed by experimenting various parameter configurations and elucidating the model's operational logic in Section 7. Finally, Section 8 discusses relevant literature, and Section 9 offers concluding remarks. Additionally, in light of the research reproducibility, we make both the dataset and the model publicly available.[1]

2. Problem statement

This section starts with a brief introduction to the necessary background, followed by a detailed exposition of the motivation driving our work, and culminates in an expression of the problem formulation.

2.1. Background

RTC applications are predominantly categorized into two types: HTTP (Hypertext Transfer Protocol) [9]-based and RTP-based. The former paradigm, exemplified by commercial video-streaming services like Netflix, is favored for its stateless and reliable attributes, making it well-suited for scenarios where delay tolerance is permissible. In contrast, RTP over User Datagram Protocol (UDP) [10] forms the cornerstone for a myriad of RTC applications, especially for occasions requiring nearly instantaneous responsiveness with minimal latency, such as video-teleconferencing and online gaming. Moreover, a wide range of IoT applications also employ RTP [11], and web browsers as well as mobile devices (e.g., Android) universally hinge on the acclaimed standard, WebRTC [12],[2] an open-source framework built atop RTP. Traditionally, Real-Time Transport Control Protocol (RTCP) is implemented alongside RTP as an integral framework for monitoring, reporting, and managing the quality and delivery of multimedia data. Nowadays, plenty of innovative technologies are developed to optimize the performance of RTP-related applications, such as QoE metrics estimation [13], adaptive bitrate control [14,15], traffic identification [16,17], congestion control [18], etc. In many cases of modern optimization approaches, information retrieved from end-users including traffic throughput, are frequently used as vital feedback of endpoints to further enhance performance.

In this context, the predicted throughput transcends the observed value, assuming an instrumental role in preemptively informing network units and enabling proactive reactions, particularly under resource-constrained circumstances [19,20]. Generally, various scenarios are envisaged to benefit from the integration and utilization of *DeX*. For instance, dynamic adaptive streaming/encoding becomes increasingly effective by incorporating the anticipated throughput as feedback [21,22], or as a state for the decision-making agent [23]. Google congestion control (GCC) [24] could leverage the prediction to supplant the recorded measurement, intervening beforehand to mitigate the network congestion. The emerging software-defined networking (SDN) paradigm is able to effectively manage networks by employing the prediction as a key factor for bandwidth allocation [25,26]. Given the diverse latent usages and the computational requirement of *DeX*, we

[1] https://mplanestore.polito.it:5001/sharing/XTiiXJOPM

[2] https://webrtc.org/

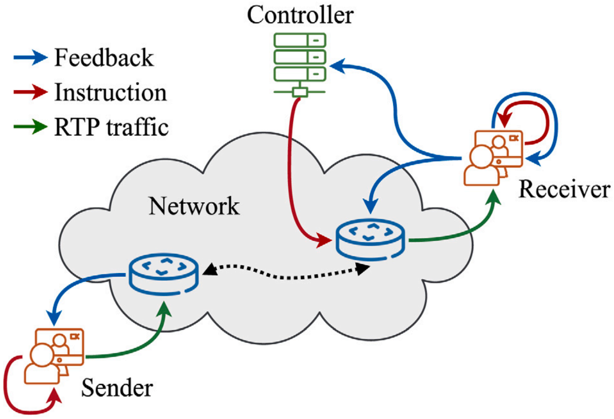


Fig. 1. Deployment scenario benefiting from *DeX*.

postulate a scenario involving multiple entities wherein our model can contribute, as depicted in Fig. 1. *DeX* could be deployed in end-users' devices (e.g., PC), with swift access to and processing of network packets through existing technologies [27–30]. The predicted throughput information serves as supportive feedback for the traffic sender, the network controller (or orchestrator), and the receiver itself, facilitating system management in an independent or cooperative manner. For example, the controller could accurately and promptly allocate more bandwidth in advance upon notification of predicted peaks or abrupt increases, potentially ensuring the consistent QoE for content delivery. Additionally, apart from functions provided by external actors, the receiver per se could also utilize the prediction to manage local resources through integrated tools [31,32].

2.2. Underlying motivations

In alignment with the three components of *DeX*, we hereby elucidate the motivation behind our work.

2.2.1. Why packet-level information

The packet selection module operates on packet-level information as its features, with the primary objective of optimally selecting a subset from the entire pool of considered packets.

The rationale underpinning the utilization of packet-level information is threefold: (i) Packets constitute the most fundamental and granular entities within networks, encapsulating the rapidly changing dynamics and inherent characteristics of network traffic [33]. Models sculpted around such meticulous features are intrinsically poised to effectively discern underlying traffic patterns, leading to enhanced prediction accuracy. (ii) The acquisition of packet-level data requires minimal effort in terms of feature extraction, an aspect particularly advantageous in the realm of RTC, where temporal and computational constraints are common. Importantly, packet encryption is becoming prevalent [34,35], rendering the acquirement and computation of intricate features not only arduous but also, at times, entirely unfeasible. Our model exclusively depends on elementary and unencrypted IP/UDP header attributes, circumventing potential complexities associated with packet encryption, and thus facilitating a more streamlined workflow with expeditious access to pertinent information. (iii) Packets are ubiquitously available across the network, extending beyond the confines of client sides, and thus affording a more holistic network observability. This broader vantage point enables the prospect of performing throughput prediction within the network, contributing to the improvement of overall network performance.

Crucially, RTC services often encounter computational and temporal limitations, and the demand for low-latency communication necessitates a delicate equilibrium between real-time response and consumed computational resources. On the one hand, RTC applications need swift as well as recurrent processing of media data and execution of various algorithms. This intensifies the computational demand, especially on devices with constrained processing capabilities, such as mobile phones and embedded systems. On the other hand, the natural pursuit of low latency in RTC inherently calls for minimal time consumption of any intermediate process. Absent this, elevated communication delays and synchronization discrepancies may emerge, impacting the overall QoE. Hence, we endeavor to decrease the total volume of input packets to downsize the model complexity, ultimately improving memory efficiency and fostering a more computationally and temporally effective paradigm.

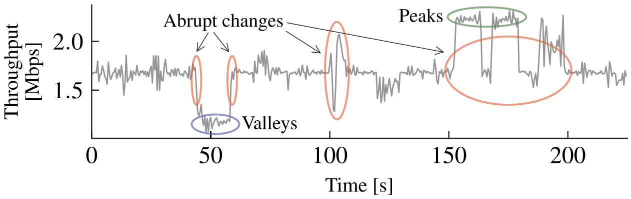


Fig. 2. Throughput time series of sample traffic.

2.2.2. Why Transformer

The feature extraction block partially incorporates a Transformer architecture to systematically condense packet series. Particularly, the sequential composition of packet flows shares affinities with problems in the domain of Natural Language Processing (NLP), which has been revolutionized by the groundbreaking game changer — Transformer. It equips our proposed model with the potential to demonstrate robust proficiency in learning the nuanced and inherent network patterns. Our objective is to harness the innate capabilities of the multi-head attention mechanism, to autonomously and intuitively discern the endogenous correlations interlacing the packet-level features and target traffic throughput.

2.2.3. Why traffic extremes

We implement the last component, a multi-task learning pipeline, aimed to improve the performance concerning traffic extremes, which constitute critical facets in RTC traffic, represent the intricate nuances of network dynamics, and exert a profound influence on prediction accuracy. To provide context, the time series throughput of an example traffic is presented in Fig. 2, where extremities are highlighted. More specifically, we underscore the prediction of extreme values for the sake of several reasons: (i) Peak values of transmission rates often coincide with network bottlenecks, providing invaluable insights into the prospective bandwidth availability. The precise prediction of peaks facilitate optimal resource allocation, which in turn averts potential pitfalls such as packet loss, compromised audio/video quality, diminished QoE, and more. (ii) Valley values denote periods of relative network idleness, where network resources remain underutilized, unveiling opportunities for energy-conservation strategies, resource redistribution, and load balancing. Moreover, certain unexpected valleys might herald network irregularities, bolstering the detection of traffic anomalies. (iii) The ability to anticipate abrupt changes, which signal sudden and transient network fluctuations, could significantly enhance adaptive streaming agility and expedited bandwidth allocation, ensuring a prompt adaptation to rapid transitions of network environment.

2.3. Problem formulation

The objective entails the prediction of traffic throughput in an upcoming time window with a duration of Δt. In order to comprehensively evaluate the performance, we approach the problem in a dual manner with distinct features but a shared target: (i) a conventional univariate

Table 1
Summary of the collected traffic.

Total number of *pcap* files	71
Total duration of traffic [h]	69.13
Average duration per *pcap* [min]	58.42
Period of collection	Apr, 2020–Jan, 2021
Total number of collected packets	66,327,753
Total number of throughput samples (500 ms)	482,905

time series problem with historical samples as features, and (ii) an irregular multivariate one with prior packet-level features. Assuming a given time instant t, we formulate a regression problem as follows:

- Problem *i* — univariate time series prediction:

$$\begin{aligned} &\hat{R}_t = f(X) \\ \text{with } &X = [r_{t-\Delta t}, r_{t-2\Delta t}, \dots, r_{t-m\Delta t}, \dots], \\ &m \in [1, M], \end{aligned} \tag{1}$$

- Problem *ii* — multivariate packet-level prediction:

$$\begin{aligned} &\hat{R}_t = f(X) \\ \text{with } &X = \underbrace{[\dots, \overline{x}_{t,n}, \dots,]}_{W}, \\ &n \in [1, N], \end{aligned} \tag{2}$$

where $\hat{R}_t$ is the predicted throughput in the ensuing time window spanning from time t to $t + \Delta t$, and the input feature matrix X varies between the two problems. For the conventional time series problem *i*, M historical samples are considered, and $r_{t-m\times\Delta t}$ denotes the previous throughput within the time window of duration Δt that commences at $t - m \times \Delta t$. In the case of the multivariate packet-level problem *ii*, we factor in the past records of N packets in total, while selecting a subset of $W < N$ packets as features. Notably, only the W chosen packets are features fed to the following prediction components, and they do not need to be contiguous. The remaining $N - W$ non-selected packets are simply unused and discarded. Should the nth preceding packet antecedent to time t be designated as one of the selected packets, $\overline{x}_{t,n}$ represents its corresponding feature vector, which is constituted by a tuple of the packet attributes (explained in the next section). The model learns a function $f(\cdot)$, undertaking the regression task and mapping the input feature matrix X into the estimated throughput that converges closely with the actual value, R. Additionally, we define the three extreme conditions for all throughput samples observed during each video-teleconferencing session as follows:

- Peak values: the throughput samples associated to the uppermost α_p (percentage) values;
- Valley values: the throughput samples associated to the lowest α_v (percentage) values;
- Abrupt changes: the throughput samples with inter-variations compared to their respective preceding samples exceeding a specific threshold (percentage) β:

$$\frac{|R_t - R_{t-\Delta t}|}{R_{t-\Delta t}} > \beta. \tag{3}$$

3. Dataset

Herein, we introduce the details of the dataset employed in our work, presenting the data source, dataset construction, feature selection, and related characteristics.

Our work is founded on abundant traffic traces collected during multiple real video-teleconferencing calls, each involving 2 to 6 participants, connected via WiFi, mobile networks, or Ethernet cables. We employ two RTC applications, *Jitsi Meet*[3] and *Webex*,[4] and gather traffic from client sides, dumping packet captures and archiving data in *pcap* format. By laying focus on incoming streams, we aim at forecasting the traffic throughput of RTP packet flows during a session, sourcing from all senders and traversing through the network. A summary of the [traffic] characteristics is provided in Table 1. To streamline the dataset construction process, we parse each *pcap* file, extracting the raw traffic data to create a datasheet, in which each entry represents the corresponding packet information, such as timestamp and packet size. Fig. 3 graphically depicts the entire procedure.

In accordance with the problem formulation, we construct the dataset by generating the time series of throughput samples for each individual video-call. Starting from the initial N packets in the traffic from a certain call, we establish a following Δt-long time window adjacent to the last packet. Afterwards, we progress forward by Δt, assimilating new packets while discarding old ones to maintain a consistent count of N packets in the past, and we iterate this procedure until consuming the entire traffic. On top of that, the traffic throughput are calculated in successive time windows following chronological order, by aggregating the frame length of all packets contained within each window. Simply put, the throughput is basically the amount of traffic per unit time, whose value is computed as $R_t = \sum l_i / \Delta t$, where l_i is the frame length of the ith packet in the time window starting from time t. In this context, we select 3 elements (i.e., $\overline{x}_{t,n} = [x^{\mathrm{FL}}_{t,n}, x^{\mathrm{IAT}}_{t,n}, x^{\mathrm{TS}}_{t,n}]$) of the RTP packet to serve as features:

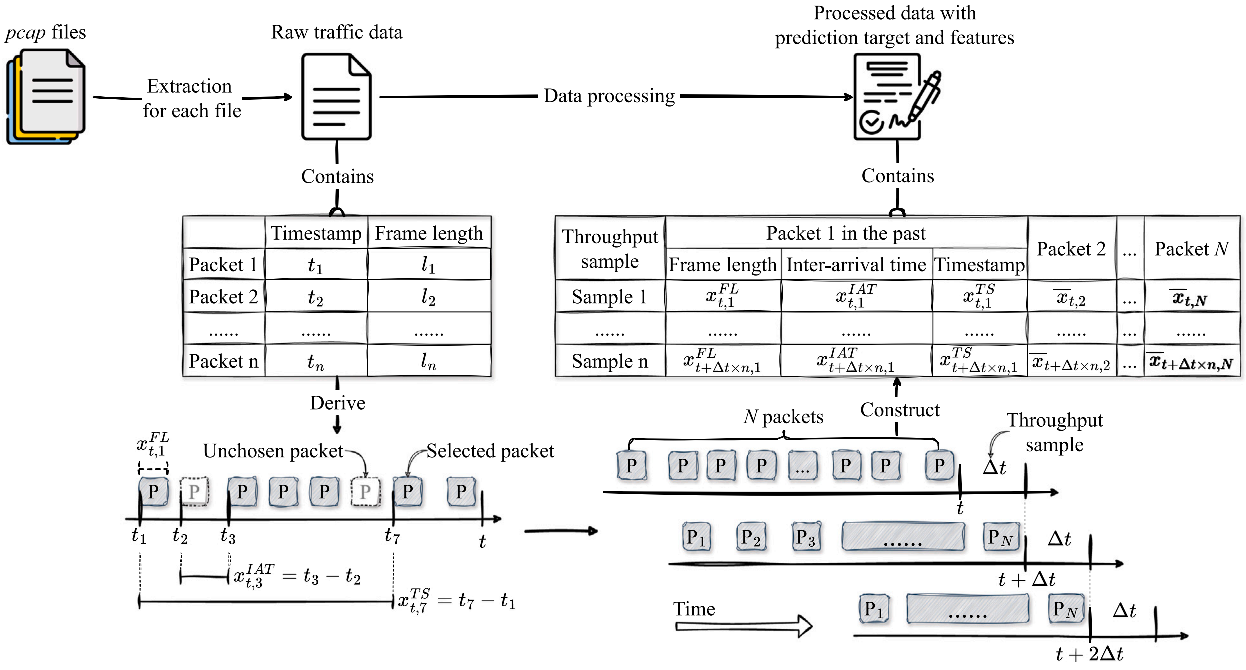
- **Frame length** ($x^{\mathrm{FL}}_{t,n}$) is the packet total length including both its header and data, which directly represents the impact of packet size and transmitted bits in the past, endowing the model with the capability to operate in an autoregressive manner.
- **Inter-arrival time** ($x^{\mathrm{IAT}}_{t,n}$) is the temporal gap between the arrival of the current packet and its previous packet, and it serves as a local granularity indicator for assessing the frequency of packet flows, even when the preceding packet in the consecutive pair is not selected.
- **Timestamp** ($x^{\mathrm{TS}}_{t,n}$) denotes the relative timestamp at which the packet is received by the end-user. It is the absolute timestamp of the current packet subtracted by the timestamp of the start of the session and it introduces global timing patterns from previously considered packets.

With these features, we intend to encompass potential influence arising from both spatial and temporal patterns, and we extract information directly from RTP packets, thereby obviating the necessity for resource-intensive feature engineering. As a consequence, each time window, i.e., a data sample of target throughput, is accompanied with the historical throughput samples from the previous M time windows for problem *i*, and the aforementioned packet-level features of each of the preceding N packets for problem *ii*.

In our initial setting, we compute and predict the throughput within time windows of $\Delta t = 500$ ms, and resort to the previous $N = 2048$ packets, with the goal of selecting half of them, i.e., $W = 1024$ packets, as features for problem *ii*. In our dataset, 2048 packets correspond to an average duration of roughly 7.6 s, and thus, a comparable time span of 8 s, equivalent to $M = 16$ (16×500 ms) prior windows is considered for problem *i*. Furthermore, we devise that $\alpha_p = 10\%$, $\alpha_v = 10\%$, and $\beta = 20\%$, i.e., the highest and lowest 10% throughput samples during a session are specified as peaks and valleys, respectively, and a sample with an inter-variation surpassing 20% when compared to its previous and neighboring sample is regarded as an abrupt change. Notice that all these selections are modifiable parameters, and we delve into alternative scenarios in Section 7.

[3] An open source platform, https://meet.jit.si/.

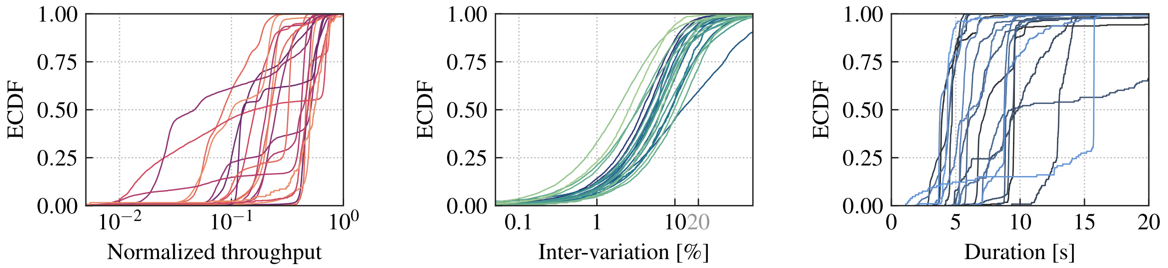
[4] A commercial application, https://www.webex.com/.



	Timestamp	Frame length
Packet 1	t_1	l_1
Packet 2	t_2	l_2
......	......	......
Packet n	t_n	l_n

Throughput sample	Packet 1 in the past			Packet 2	...	Packet N
	Frame length	Inter-arrival time	Timestamp			
Sample 1	$x^{FL}_{t,1}$	$x^{IAT}_{t,1}$	$x^{TS}_{t,1}$	$\overline{x}_{t,2}$	...	$\boldsymbol{\overline{x}_{t,N}}$
......	......	......	......	......	...	......
Sample n	$x^{FL}_{t+\Delta t\times n,1}$	$x^{IAT}_{t+\Delta t\times n,1}$	$x^{TS}_{t+\Delta t\times n,1}$	$\overline{x}_{t+\Delta t\times n,2}$	...	$\boldsymbol{\overline{x}_{t+\Delta t\times n,N}}$

Fig. 3. The process of data preprocessing.



(a) Throughput values for individual sessions.

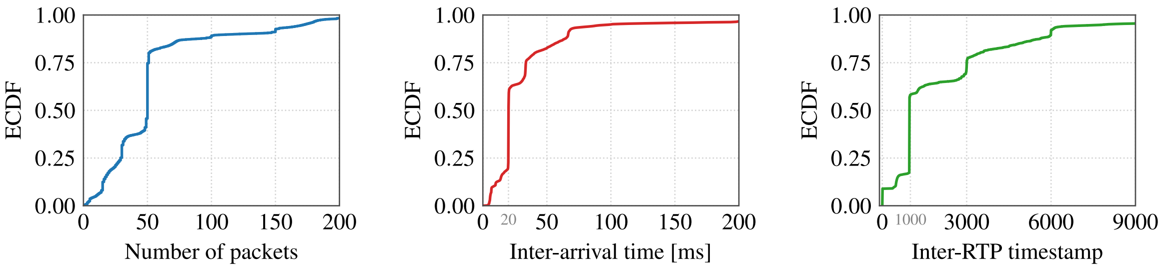
(b) Percentage change of a throughput sample comparing to its preceding sample.

(c) Time span of 2048 packets.

Fig. 4. Traffic patterns of 20 randomly selected video-teleconferencing sessions.

To provide contextual insight, we illustrate traffic patterns from 20 randomly selected sessions in Fig. 4, which depicts 3 sets of Empirical Cumulative Distribution Function (ECDF) plots.[5] On the one hand, the leftmost Fig. 4(a), which shows the ECDF of throughput values, demonstrates that nearly all values exhibit a steep ascent in the middle, gradually tapering into narrower tails for both ultra-low and high values, despite quantitative differences among traffic. On the other hand, the middle Fig. 4(b) showcases the percentage variations (inter-variation) between successive throughput samples, revealing that the majority of inter-variations remain below 20%. In fact, 64.9% of inter-variations are smaller than 10%, and 84.2% are smaller than 20% for all traffic. Both of the previous observations suggest that the traffic throughput generally undergoes a globally stationary evolution, which underlines the significance of comprehending and forecasting traffic extremes, further rendering their prediction an intriguing and substantial endeavor. Furthermore, we also investigate the duration of each set of 2048 packets for separate sessions in the rightmost Fig. 4(c). Although the average elapse is 7.6 s as mentioned earlier, different traffic exhibit various characteristics, where multiple traffic sessions share a similar pattern with a majority of cases covering a duration ranging from 3 to 10 s, as indicated by the rapid ascent in the ECDF, and several sessions manifest peculiar patterns, characterized by either relatively uniform or irregular distributions of elapsed duration. Such phenomena are actually common in reality, given that a video-call could be either active with frequent data packets exchanged or quiescent with a few transmitted packets. The aforementioned aspects provide rationality for choosing $N = 2048$ packets to encompass the multifarious facets of traffic transmitted in RTC, regardless of the degree of activity. This leads to a judicious trade-off between including

[5] Notes: (i) The display of only 20 traffic samples is for the sake of a relatively lucid visualization, and we can confirm a similar pattern across the dataset. (ii) Rather than amalgamating all the traffic to generate a ECDF for the entire dataset, we choose to construct individual ECDFs, in order to illustrate and compare the pattern of each individual traffic trace separately.



(a) Total quantity of packets per second.

(b) Average inter-arrival time of all packets per second.

(c) Average difference between RTP timestamps of all packets per second.

Fig. 5. Traffic patterns of individual RTP flows (this is not the pattern of the entire dataset with mixed flows).

extraneous and excessive information beyond the target and having insufficient packets.

Moreover, we also provide insights into the actual multimedia data being transmitted by analyzing the statistics of individual RTP traffic flows. A single RTP flow, responsible for conveying particular content, such as audio or video, is uniquely determined by a tuple composed of $(IP_{src},\ IP_{dst},\ port_{src},\ port_{dst},\ SSRC,\ type_{payload})$.[6] For the traffic of each session, we segregate and extract data on a per-flow basis, and then for each flow, we aggregate packets into consecutive 1-s time bins, calculating 3 types of statistics for each bin, i.e., the total number of packets, the average inter-arrival time, and the average inter-difference between RTP timestamps, whose ECDFs are presented separately in Fig. 5. Importantly, all 3 plots indicate a notable degree of statistical similarity, featuring a pronounced increase to above 50% in the ECDFs around either 50 packets/s (Fig. 5(a)), an inter-arrival time of 20 ms (Fig. 5(b)), or an inter-RTP timestamp of 1000 (Fig. 5(c)). Such a trend is due to the presence of audio flows that are typically packetized adhering to the protocol RTP RFC 3550 [36]. For example, given that the most common audio packetization implementation employs a 20-ms interval, and the widely-used codecs like Opus operates at a frequency of 48 kHz [37,38], the difference in RTP timestamp between two consecutive audio packets is 960 ($48\ \text{kHz} \times 20\ \text{ms} \approx 1000$). Building upon these observations, coupled with other distinct sudden rises in the ECDFs that potentially indicate various payload types, we can reasonably infer that our dataset is composed of abundant and different media data rather than monotonous content. This diversity ensures the datasets' relevance and representativeness in capturing the numerous aspects of RTC traffic, complicating somewhat the problem because of the mixture of different patterns, but simultaneously consolidating the versatility and comprehensiveness of our model.

4. Methodology

In this section, we describe the architecture of our proposed model, *DeX*. Subsequently, other considered approaches for comparison and the model development as well as evaluation process are outlined.

4.1. Introduction of the proposed model

Our novel DL framework *DeX* leverages historical packet-level features to predict traffic throughput in future time windows. The architecture of *DeX* comprises three components, namely, a packet selection module, a feature extraction block, and a multi-task learning pipeline, as elucidated in Fig. 6. In general, we perform a moving window prediction during the training phase, taking into account all of the preceding $N = 2048$ packets with three RTP elements, as raw input. Along with the training, *DeX* learns an optimal subset out of all available packets, selecting a portion as actual input features. Following the feature extraction process, we adopt a multi-task learning paradigm that integrates various loss functions to optimize the performance regarding traffic extremities. Particularly, each component is detailed in the following.

4.1.1. Packet selection module

This component reduces the input quantity while maintaining the performance by selecting an optimal subset out of all considered packets. To choose which packets to select, we employ a straightforward logic, by randomly initializing $N = 2048$ trainable parameters $(\mathcal{P}_i)$, each being assigned to the corresponding packet (P_i). Subsequently, we pass all of these parameters through a Softmax function to derive 2048 probabilities $(\overline{p}_{\text{softmax}})$ to opt for corresponding packets based on their values, i.e., we select the $W = 1024$ packets associated to the highest 1024 probabilities.[7] It is important to note that the outputs of Softmax are called probabilities simply because of the naming convention, and thus do not imply a stochastic process but are treated as deterministic scores for packet selection. We expect the model to learn and refine the trainable parameters in a way such that the derived probabilities (scores) are optimized to select the most suitable packets for the regression task.

However, the aforementioned procedure possesses a fundamental flaw of not being attached to the computational graph, given that such a selection, which triggers no computations in the last step, is not involved in the gradient flow. In order to tackle the issue, we introduce a predefined distribution of probabilities $(\overline{p}_{\text{predefined}})$ to compare the derived probabilities output by Softmax, thereby bringing in computational process. Meanwhile, we postulate that the potentially optimal selections are the packets closest to the target throughput sample, i.e., the most recent 1024 packets in the packet sequence. This hypothesis stems from the nature of time series problem, where temporally proximate samples ought to encompass the most salient features, reflecting the latest trend in evolution. However, unlike conventional time series problem, the domain-specific irregularity in packet sequence could potentially provide critical information to various packets other than those proximate to targets. Therefore, we conceive the preset probabilities (scores) as a guideline to steer the learning process as well as the selection towards the hypothetically optimal (closest) packets, while still permitting a certain degree of freedom to unearth potentially valuable packets located further away from the targets.

[6] IP_{src} and IP_{dst} are the IP addresses of the traffic source and destination, respectively, $port_{src}$ and $port_{dst}$ are the ports of source and destination, $SSRC$ is the synchronization source identifier that uniquely identifies the source of a stream, and $type_{payload}$ is dynamically assigned to represent the format of the payload.

[7] $\overline{P}_{\text{selected}} = \{P_i \mid \sum_{i \neq j} \prod_{i \neq j} (p_{\text{softmax},i} > p_{\text{softmax},j}) > 1024\}, \forall i, j \in [1, 2048]$.

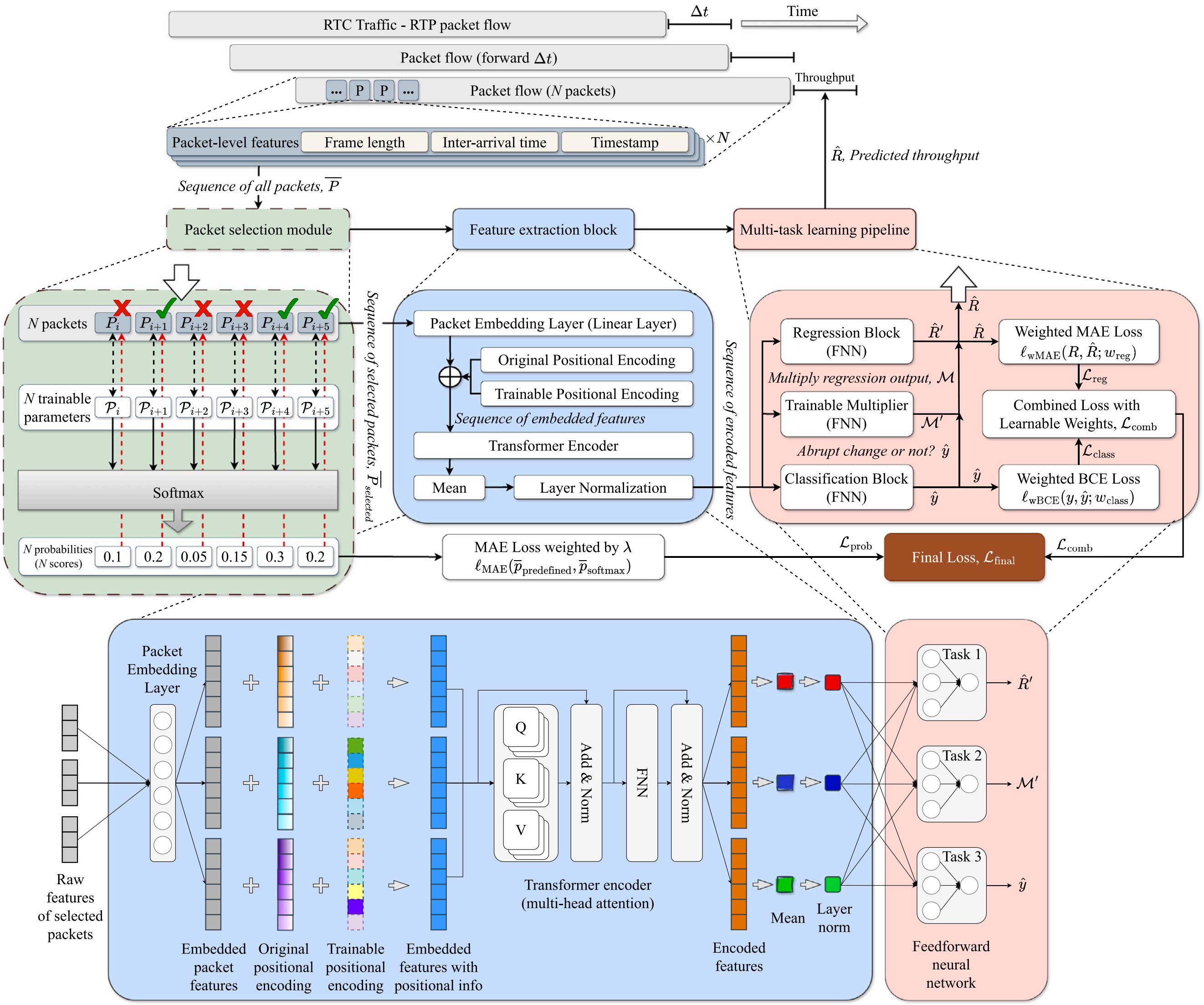

RTC Traffic - RTP packet flow
Δt
Time
Packet flow (forward Δt)
... P P ...
Packet flow (N packets)
Throughput
Packet-level features
Frame length
Inter-arrival time
Timestamp
×N
R̂, Predicted throughput
Sequence of all packets, P̄
Packet selection module
Feature extraction block
Multi-task learning pipeline
N packets
P_i P_{i+1} P_{i+2} P_{i+3} P_{i+4} P_{i+5}
N trainable parameters
$\mathcal{P}_i$ $\mathcal{P}_{i+1}$ $\mathcal{P}_{i+2}$ $\mathcal{P}_{i+3}$ $\mathcal{P}_{i+4}$ $\mathcal{P}_{i+5}$
Softmax
N probabilities (N scores)
0.1 0.2 0.05 0.15 0.3 0.2
Sequence of selected packets, $\overline{P}_{\text{selected}}$
Packet Embedding Layer (Linear Layer)
Original Positional Encoding
Trainable Positional Encoding
Sequence of embedded features
Transformer Encoder
Mean
Layer Normalization
Sequence of encoded features
Regression Block (FNN)
Multiply regression output, $\mathcal{M}$
Trainable Multiplier (FNN)
Abrupt change or not? $\hat{y}$
Classification Block (FNN)
$\hat{R}'$ $\hat{R}$ $\mathcal{M}'$ $\hat{y}$
Weighted MAE Loss $\ell_{\text{wMAE}}(R, \hat{R}; w_{\text{reg}})$
$\mathcal{L}_{\text{reg}}$
Combined Loss with Learnable Weights, $\mathcal{L}_{\text{comb}}$
$\mathcal{L}_{\text{class}}$
Weighted BCE Loss $\ell_{\text{wBCE}}(y, \hat{y}; w_{\text{class}})$
MAE Loss weighted by λ $\ell_{\text{MAE}}(\overline{p}_{\text{predefined}}, \overline{p}_{\text{softmax}})$
$\mathcal{L}_{\text{prob}}$
Final Loss, $\mathcal{L}_{\text{final}}$
$\mathcal{L}_{\text{comb}}$
Packet Embedding Layer
Raw features of selected packets
Embedded packet features
Original positional encoding
Trainable positional encoding
Embedded features with positional info
Q K V
Add & Norm
FNN
Add & Norm
Transformer encoder (multi-head attention)
Encoded features
Mean
Layer norm
Task 1 Task 2 Task 3
$\hat{R}'$ $\mathcal{M}'$ $\hat{y}$
Feedforward neural network

Fig. 6. Workflow, model architecture, and training strategy of *DeX*.

To this end, we devise a simple predefined and monotonically increased linear distribution[8] as follows:

$$\begin{aligned} &\overline{p}_{\text{predefined}} = [p_1, \ldots, p_i, \ldots, p_{2048}] \\ &\qquad \text{s.t. } p_i < p_{i+1}, \\ &\qquad\qquad \sum p_i = 1, \\ &\qquad\qquad p_i = a \cdot i + b, \\ &\quad \text{with } i \in [1, 2048], \end{aligned} \tag{4}$$

where p_i represents the predefined probability (score) assigned to the ith packet (the larger indexes are closer to the target), and the summation of $\overline{p}_{\text{predefined}}$ equals 1, as the output of Softmax sums to 1. The term $a \cdot i + b$ indicates a linear relationship.[9] A visual example is illustrated in Fig. 7, and the parameters, $\mathcal{P}$, are trained to produce Softmax probabilities that converge to the predefined ones, accomplished by computing the Mean Absolute Error (MAE loss function, $\ell_{\text{MAE}}(\cdot)$) between them:

$$\begin{aligned} &\mathcal{L}_{\text{prob}} = \ell_{\text{MAE}}(\overline{p}_{\text{predefined}}, \overline{p}_{\text{softmax}}) \\ &\text{with } \overline{p}_{\text{softmax}} = \text{Softmax}(\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_{2048}), \end{aligned} \tag{5}$$

in which, $\mathcal{L}_{\text{prob}}$, namely the loss of probability, is one of the optimized targets subjected to minimization. As a result, the module operates in a manner that diligently pushes the selection of packets towards the target. Nevertheless, the procedure encounters another problem — it always ends up with the proximate packets due to the consistent gravitation towards the hypothetically optimal selection,[10] deviating from our original goal of choosing informative packets in distance. To

[8] Superficially, the choice of such a distribution may appear somewhat arbitrary, but, by fine-tuning the hyperparameters, it is possible for the NN to automatically cherry-pick beneficial packets, reaching optimal performance, irrespective of the initial distribution chosen. Therefore, the specific distribution adopted is inconsequential, as long as it satisfies our requirement, for example, an exponentially increased distribution of probabilities (scores) could also serve the purpose, as we elaborate on in Section 6. In other words, the predetermined probabilities function as a reference, and different predefinitions, i.e., different references, will unquestionably influence the learning process, engendering distinct alterations in the trainable parameters, but the ultimate optimized objective, that is always the regression task, remains invariant, impelling the model to eventually ascertain the advantageous packet selection, regardless of the trajectory it follows in relation to the convergence towards various references.

[9] Details regarding their derivation are in Section 4.2.

[10] The solution to minimization of MAE exists, i.e., $\mathcal{L}_{\text{prob}} = 0$, when the trainable parameters yield Softmax probabilities identical to the predefined ones.

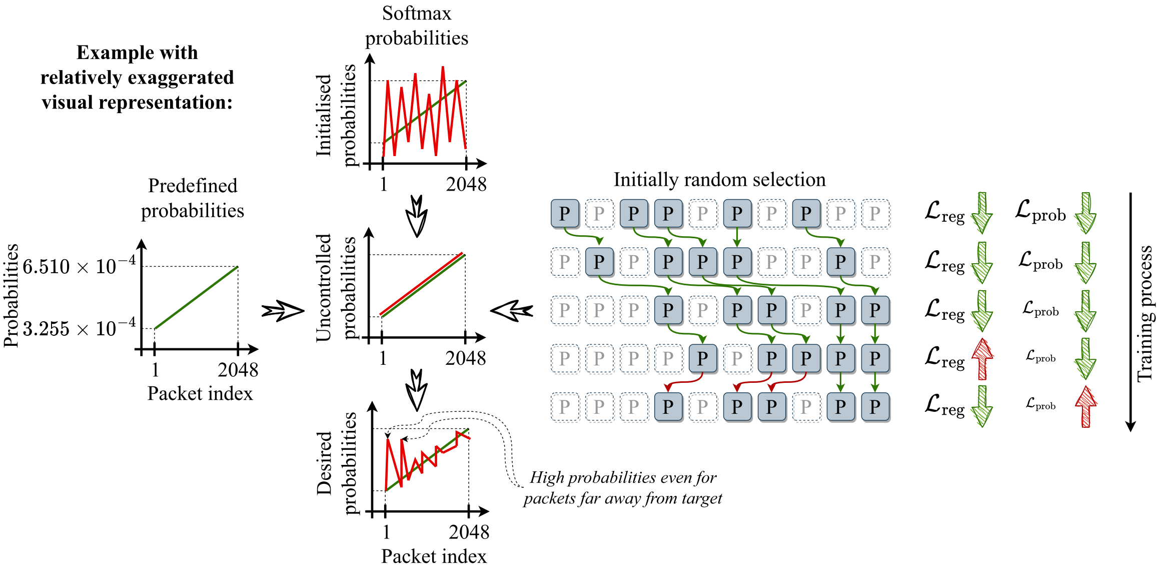

Example with relatively exaggerated visual representation:
Softmax probabilities
Initialised probabilities
1
2048
Predefined probabilities
Probabilities
6.510×10^{-4}
3.255×10^{-4}
1
2048
Packet index
Uncontrolled probabilities
1
2048
Desired probabilities
1
2048
Packet index
High probabilities even for packets far away from target
Initially random selection
P
$\mathcal{L}_{\text{reg}}$
$\mathcal{L}_{\text{prob}}$
Training process

Fig. 7. How packet selection module works (note that $\mathcal{L}_{reg}$ in the figure is just a nominal representation instead of the actual regression loss in Eq. (6)).

alleviate such an aggressive outcome, we mitigate the MAE loss, $\mathcal{L}_{\text{prob}}$, by introducing a weighting factor, λ, to reduce its impact, preventing from permanently choosing the packets closest to the target. An illustrative depiction of this concept is presented in the right part of Fig. 7. As the training process unfolds, both losses of regression and probability decrease synchronously, until reaching a certain point, where the packet selection is deemed non-optimal, leading to an upswing in the regression loss. Simultaneously, the MAE loss of probability continues to decrease, further abating its influence thanks to the weight, λ, and when the regression loss escalates because of certain selection, the selection revisits a prior relatively optimal position, given that the loss of probability exerts mere impact on the final loss. In consequence, the selection process stabilizes or hovers around the optimal locations, primarily due to the dominance of the regression loss.

Consequently, we optimize the packet selection throughout the training process, significantly reducing the amount of input for downstream components. Notice that such a selection module will be rendered superfluous and discarded post-training without continuing to increase the model complexity, since the optimal position in the packet sequence is already derived, and from then on, we can directly channel the selected packets into the next components.

4.1.2. Feature extraction block

We employ a Transformer-based NN with customized structures to extract features. Firstly, we inject the sequence of selected packets, i.e., each set of the 3 features (a 1×3 vector), into a packet embedding layer (linear layer) to create embedded features. The NN is anticipated to learn an apt mapping from the primordial attributes of a packet to its latent embedding, thereby enriching the traffic features and transforming the dimension of the feature vector to $1 \times N_{\text{embedding}}$. Secondly, while each packet has the timestamp indicative of its order, we still lack of positional information for other packet entities. To address this, we implement *sine/cosine* positional encoding, following the original Transformer model. On top of that, we augment the architecture by introducing an additional trainable positional encoding constituting of learnable parameters for two main purposes: (i) learning automatically optimal positional and potentially domain-related patterns during training to improve the task-specific adaptability, considering that the original one is fixed and particularly designed for NLP, and (ii) supplementing probably absent insights caused by inconsistent packet selection, since the original one operates on continuous sequence without any interstitial gap in between. With the same dimension of $W \times N_{\text{embedding}}$, both positional encodings are superimposed to embedded features, wherein each vector ($1 \times N_{\text{embedding}}$) at the second dimension of the encoding is added to its corresponding packet embedding. Thirdly, the resultant sequence of embedded features is fed into a single Transformer encoder, a component frequently employed for sequence representation [39,40], to generate encoded features. Our objective is to leverage the multi-head attention mechanism, the core of the encoder, to unveil latent patterns and apprehend network fate. Moreover, we opt not to implement the Transformer decoder or additional stacks of encoders to avoid increasing the model complexity. Finally, the output encoded features comprise a total of $W \times N_{\text{embedding}}$ entries. While a straightforward solution would be to connect all values to each subsequent task (a feedforward neural network (FNN)), we refrain from doing so to prevent potential excessive noise and model over-complication. Instead, we calculate the mean value for each set of encoded features in the output sequence, distilling feature quintessence and deriving the ultimate feature vector with a dimension of $1 \times W$. Additionally, we apply layer normalization to standardize the condensed features dependently for an individual sample, aiming to stabilize the pattern within each input and consummate the feature extraction phase.

4.1.3. Multi-task learning pipeline

We elaborately craft a multi-task learning strategy, enriched with multifunctional weights, designed to further incentivize the model to discern traffic extremes. Besides the primary regression task, we incorporate two auxiliary learning blocks: a binary classification component and a trainable multiplier. The former block aims to predict and identify whether the target throughput signifies an abrupt change with respect to the preceding sample, a feat deemed attainable due to the granular and domain-specific packet-level features that are often absent in conventional time-series scenarios. The latter block operates as a calibrator that either amplifies or attenuates the regression output based on the classification outcome, adjusting the final predictions to better accommodate dramatic variations. Each block (task) consists of a 2-layer FNN, which takes as input the encoded features generated by the feature extraction block and produces a scalar value. For the classification block, a Sigmoid function is adopted to convert the output into class probabilities. Consequently, the NN undergoes meticulous training, ensuring that it consistently satisfies normal value expectations, while also methodically compensating for abrupt changes. Meanwhile, we implement learnable weights [41] that autonomously determine the

Table 2
Implementation detail of *DeX*.

Parameter	Value
Learning rate[a], η	10^{-3}
Size of feature embedding, $N_{\text{embedding}}$	32
Size of positional encoding	1024 × 32
Number of heads	8
Number of encoder	1
Number of neurons for FNN in encoder	512
Activation function in encoder	*ReLU* [42]
Number of layers for multi-task learning pipeline	2
Number of neurons of the 1st layer for a task in pipeline	512
Number of neurons of the 2nd layer for a task in pipeline	1
Activation in multi-task learning pipeline	*ReLU*
Training optimizer	Adam [43]
Batch size	16
Weight for peaks and valleys, w_{reg}	2.0
Weight for abrupt changes, w_{class}	6.0
Weight for loss of probability, λ	4×10^{-4}
Parameters for the predefined probability, $\overline{p}_{\text{predefined}}$	$a = 1.59 \times 10^{-7}$ $b = 3.25 \times 10^{-4}$

[a] We adopt a decay of 1 order of magnitude for every 2 epochs.

Table 3
Model summary.

Category	Model
Naive baseline[a]	Moving Average (MA)[b] [44]
Adaptive filter	Recursive Least Squares (RLS)[b] [45]
ML method	Random Forest (RF)[b] regressor [46] XGBoost (XGB)[b] regressor [47]
DL method	Multi Layer Perceptron (MLP)[b] [48] Long- and Short-term Time-series network (LSTNet)[c] [49] Long Short-Term Memory (LSTM)[b],[c] [50] N-BEATS network[b] [51]

[a] It calculates the average value of past throughput samples as the prediction.

[b] Problem *i*, univariate time series prediction.

[c] Problem *ii*, multivariate packet level prediction.

importance of different tasks to systematically and optimally combine losses generated by different blocks, as depicted below:

$$
\begin{aligned}
\mathcal{L}_{\text{comb}} &= e^{-w_1} \cdot \mathcal{L}_{\text{class}} + w_1 + e^{-w_2} \cdot \mathcal{L}_{\text{reg}} + w_2 \\
\text{with } \mathcal{L}_{\text{class}} &= \ell_{\text{wBCE}}(y, \hat{y}; w_{\text{class}}), \\
\mathcal{L}_{\text{reg}} &= \ell_{\text{wMAE}}(R, \hat{R}; w_{\text{reg}}), \\
\hat{R} &= \hat{R}' \cdot \mathcal{M}, \\
\mathcal{M} &= \begin{cases} \mathcal{M}', & \text{if } \hat{y} = 1 \text{ (abrupt change)} \\ 1, & \text{if } \hat{y} = 0 \text{ (normal transition)}, \end{cases}
\end{aligned}
\tag{6}
$$

where $\mathcal{L}_{\text{comb}}$ stands as the combined loss, calculated by blending the classification ($\mathcal{L}_{\text{class}}$) and regression ($\mathcal{L}_{\text{reg}}$) losses via learnable weights, w_1 and w_2. Moreover, both regression and classification blocks leverage weighted losses during training phase. On the one hand, in order to tackle the imbalance between classes (only around 16% of throughput samples are abrupt changes), the classification loss is calculated by the weighted Binary Cross Entropy (BCE) loss function $\ell_{\text{wBCE}}(\cdot)$, with elevated weights w_{class} granted to the minority samples of abrupt changes. On the other hand, the weighted Mean Absolute Error (MAE) loss function $\ell_{\text{wMAE}}(\cdot)$ is employed for regression, with larger weights w_{reg} assigned to peaks and valleys to accentuate the model's sensitivity to such scenarios. Both y and R represent the ground truths of classification and regression tasks, and $\hat{y}$ denotes the label predicted by classification block, while $\hat{R}$ symbolizes the final forecasted throughput, ascertained by modulating the regression output ($\hat{R}'$) with the intervention of the trainable multiplier ($\mathcal{M} = \mathcal{M}'$). Notably, when the classification indicates a normal transition ($\hat{y} = 0$), the multiplier remains neutral ($\mathcal{M} = 1$), thus leaving the regression output unaltered.

Finally, by considering the entire model, the final loss is computed as:

$$
\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{comb}} + \lambda \cdot \mathcal{L}_{\text{prob}}, \tag{7}
$$

in which $\mathcal{L}_{\text{comb}}$ is the combined loss yielded by the multi-task learning pipeline in Eq. (6), and $\mathcal{L}_{\text{prob}}$ corresponds to the loss of probability tweaked by the hyperparameter, λ, and generated by packet selection module in Eq. (5). The second term can be regarded as a regularization component, imposing constraints on the learning process, that are instrumental in preventing the model from becoming overly dependent on proximal packets, and thereby nudge the model towards solutions that are not only effective on the primary task but also exhibit a level of flexibility and adaptability when faced with varying packet selections.

4.2. Model development, comparison, and evaluation process

DeX is developed using the *Pytorch* [52] framework and is trained on a single GPU of NVIDIA Tesla V100-16 GB. The implementation details are enumerated in Table 2. Notably, the parameters for the predefined probabilities, a and b, are not initialized randomly but derived based on the following procedure:

- Step 1 – define a line space of $N = 2048$ elements with uniform increment:

$$
\begin{aligned}
&\Theta = [\theta_1, \dots, \theta_i, \dots, \theta_{2048}] \\
&\text{s.t.} \quad i \in [1, 2048], \\
&\theta_i = \theta_{i-1} + \frac{1}{2048}, \\
&\theta_1 = 1, \ \theta_{2048} = 2.
\end{aligned}
$$

- Step 2 – normalize the line space to sum to 1:

$$
\begin{aligned}
&\overline{p}_{\text{predefined}} = \Theta \leftarrow \frac{\Theta}{\sum \Theta}, \text{with} \sum \Theta = 3071.5 \\
&\qquad\qquad \Downarrow \\
&\sum \overline{p}_{\text{predefined}} = 1, \\
&p_1 = \theta_1 = 1/3071.5 \approx 3.256 \times 10^{-4}, \\
&p_{2048} = \theta_{2048} = 2/3071.5 \approx 6.511 \times 10^{-4}.
\end{aligned}
$$

As a result, the parameters a and b are computed accordingly. In fact, the actual controllers governing the linearity of the distribution are the values of the head and tail of Θ, i.e., θ_1 and θ_{2048}, for which we simply endow with a rudimentary initialization of 1 and 2.

Furthermore, we deliberately and randomly partition the corpus of 71 *pcap* files (video-calls) into 3 independent groups (50, 10, 11) to construct training (355,651 samples of throughput), validation (62,193), and test (65,061) datasets. For features and throughput samples, we calculate based on the training set the statistics of mean value and standard deviation, which are then used to standardize validation and test sets. Consequently, the model is trained based on traffic collected under unique conditions different from other datasets in terms of location, connectivity, and time, aiming to derive a generalized solution and preclude data cross-contamination among traffic. Additionally, akin to the domain of Computer Vision, each throughput sample (target) is stored in a single file (*npy* format in our case). Each file is composed of a matrix with dimensions 2048 × 3, that represents the 3 chosen features of the preceding 2048 packets, while the file name contains the throughput value, along with the corresponding percentile and intervariation, derived within a single session to respect our initial setting for the thresholds of traffic extremes.

We also extend our examination to a wide range of domains for the purpose of comparison, incorporating multiple other technologies appeared in the literature, as listed in Table 3. Notably, a total of 7 models are implemented for problem *i*, while 3 models including *DeX*

Table 4
Experimental result of all models regarding overall traffic, peaks, valleys, and abrupt changes.

Problem					Problem *i*				Problem *ii*		
Feature					Historical time series samples				Packet-level information		
Model		MA	RLS	RF	XGB	MLP	LSTM-*i*	N-BEATS	LSTM-*ii*	LSTNet	*DeX*
Overall values [Mbps]	MSE ↓	0.0984	0.0777	0.0499	0.0522	0.0524	0.0530	0.0516	0.0577	0.0510	**0.0474**
	MAE ↓	0.1620	0.1307	0.1175	0.1187	0.1205	0.1208	0.1160	0.1245	0.1171	**0.1147**
	MAPE ↓	15.726%	12.299%	11.844%	10.827%	12.453%	12.354%	10.600%	11.286%	11.457%	**10.597%**
	R^2 ↑	0.8461	0.8785	0.9220	0.9184	0.9180	0.9171	0.9193	0.9101	0.9204	**0.9261**
Peak values [Mbps]	MSE ↓	0.2623	0.2398	0.1798	0.1895	0.1759	0.1762	0.1799	0.1990	0.1821	**0.1688**
	MAE ↓	0.3290	0.2685	0.2662	0.2706	0.2624	0.2645	0.2610	0.2767	0.2581	**0.2460**
	MAPE ↓	15.807%	13.167%	12.852%	13.142%	12.349%	12.610%	12.795%	12.871%	12.298%	**12.263%**
	R^2 ↑	0.7286	0.7520	0.8140	0.8040	0.8181	0.8177	0.8139	0.7953	0.8127	**0.8263**
Valley values [Mbps]	MSE ↓	0.1196	0.0570	0.0335	0.0314	0.0424	0.0421	0.0314	0.0340	0.0306	**0.0245**
	MAE ↓	0.1619	0.1021	0.0913	0.0819	0.1037	0.1017	0.0814	0.0805	0.0834	**0.0777**
	MAPE ↓	33.553%	23.158%	23.133%	18.954%	27.271%	27.107%	19.366%	19.825%	23.178%	**17.669%**
	R^2 ↑	−0.4927	0.2888	0.5819	0.6078	0.4711	0.4749	0.6080	0.5774	0.6189	**0.6948**
Abrupt changes [Mbps]	MSE ↓	0.3137	0.3671	0.2805	0.2898	0.2866	0.2960	0.2922	0.3022	0.2825	**0.2645**
	MAE ↓	0.3740	0.3992	0.3737	0.3768	0.3800	0.3884	0.3819	0.3983	0.3750	**0.3574**
	MAPE ↓	43.283%	42.834%	40.305%	39.055%	41.900%	42.080%	39.105%	43.018%	38.595%	**36.057%**
	R^2 ↑	0.5861	0.5156	0.6299	0.6177	0.6219	0.6095	0.6145	0.6009	0.6269	**0.6507**

are developed for problem *ii*.[11] Moreover, we evaluate the performance of each model across various dimensions, including overall traffic, peaks, valleys, and abrupt changes, by gauging 4 metrics between the ground truth and prediction: Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the coefficient of determination (R^2 score).

5. Experimental result

In this section, we compare *DeX* against baselines, presenting the experimental outcomes derived from all models based on the initialized configuration. Then, two visual instances are displayed to further endorse our findings.

Table 4 showcases the assessment metrics independently for overall traffic, peak values, valley values, and abrupt changes. We embark upon the overall performance outlined in the first part. Evidently, *DeX* outshines other solutions across all quantitative measures. Although certain models, such as RF, XGB, N-BEATS, and LSTNet, yield results that are ostensibly on par, e.g. N-BEATS achieves a nearly identical MAPE of 10.6%, they invariably falter in other metrics, like RF's declined MAPE of 11.844%. Additionally, the MA baseline and RLS filter manifest the poorest performance, marked by the most substantial errors (e.g., MSE > 0.07) and inferior R^2 scores (<0.9), indicating the inadequacy of simple statistical tools.

Proceeding to the plateaus as well as troughs depicted by the second and third parts in the table, *DeX* significantly outperforms its counterparts. On the one hand, our model stands out in terms of peak values, boasting the most minimal errors (e.g., the only MSE beneath 0.17 and the sole MAE lower than 0.25) and a preeminent R^2 score (the only one exceeding 0.82), affirming its capacity for precise forecasting rather than mere overestimation. On the other hand, the superiority becomes even more conspicuous when examining valley values. *DeX* delivers markedly diminished errors and remarkable coherence, as evidenced by a MAPE that is lowered by 1.285% and an R^2 score augmented by 0.0759 in comparison to their respective second-best values.

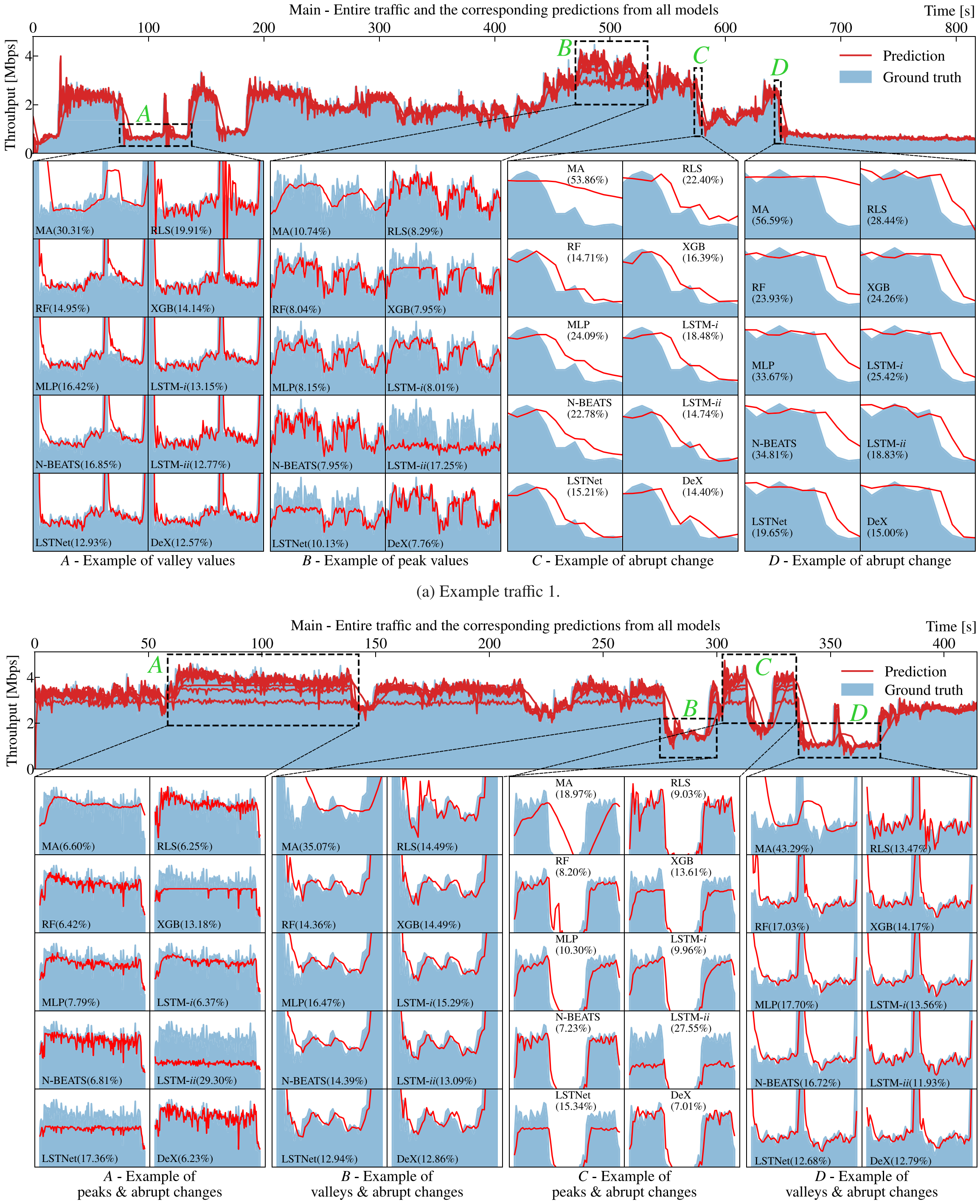
As for abrupt changes, *DeX* consistently excels other models with optimal performance across the board, exemplified by being the only model with an MSE below 0.27. However, it remains intrinsically arduous to precisely predict such rapid and sudden transitions, given the relatively subpar performance regardless of the models. The non-ideal result originates from the inherent complexities entwined within the problem per se. Instantaneous fluctuations of throughput in the context of RTC could be induced by a multitude of factors, like emergent traffic surges or network disruptions, elements which may not manifest prominently in the packet flows received by end-users, thus rendering them elusive and challenging to be detected by the models. Yet, amidst this backdrop of hurdles, *DeX* skillfully harnesses granular packet-level insights in conjunction with a meticulously designed model architecture, accommodating abrupt changes to a commendable extent.

Two traffic examples from the test dataset are outlined in Fig. 8, which portrays the time series of throughput regarding ground truth alongside the corresponding predictions, and accentuates traffic extremities in four sub-figures with MAE underscored in parenthesis.[12] Generally, all models exhibit an aptitude for tracking the fundamental traffic evolution, albeit to varying degrees of proficiency. Upon closer analysis, *DeX* exhibits superiority characterized by the lowest error in the majority of cases, while the MA baseline falls short in adapting the variations and RLS filter produces aggressive and volatile predictions. In particular:

- The first example in Fig. 8(a) presents individual instances of critical traffic scenarios. For valley values (sub-figure *A*), *DeX* excels the others by deftly following the localized fluctuations, although performance disparities are relatively subtle with respect to certain models, e.g., LSTM-*ii* and LSTNet. Notably, certain models, like RF and MLP, suffer from the sudden drop, resulting in unsatisfactory performance at the onset of troughs. Concerning peak values (sub-figure *B*), *DeX* is good in capturing and adapting to summits, whereas the majority of other models tend to generate underestimated predictions. On top of that, *DeX* also outperforms its peers in terms of abrupt changes (sub-figures *C* & *D*), by swiftly and precisely accommodating the instantaneous declines. Remarkably, *DeX* can even anticipate the precipitate rise after a serial descents in example *C*, while yielding an optimal error, 3.83% lower the penultimate one in example *D*. However, each model exhibits a latency in response to the initial abrupt transformation, reaffirming the notion that it is barely possible to predict drastic transitions.
- The second example in Fig. 8(b) depicts mixtures of traffic extremes. Regarding peaks coupled with abrupt changes (sub-figures *A* & *C*), *DeX* shows the smallest errors, whereas the tendency of underestimation from the most of other models reappears. More importantly, *DeX* also demonstrates how swift and effective it

[11] It is noteworthy that the other two models leverage the nearest 1024 packets as features without packet selection.

[12] The presence of MAE facilitates performance comparison in cases where predictions are visually hard to differentiate.



(a) Example traffic 1.

(b) Example traffic 2 (white spaces are present in the zoomed-in figures for a clear visualization at the edges).

Fig. 8. Ground truth & predictions: traffic examples of throughput time series with highlights on peaks, valleys, and abrupt changes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is to transition from a prior precise prediction of critical values to another subsequent critical point in the opposite direction. Meanwhile, similar capability is also applied to valleys with abrupt changes (sub-figures *B* & *D*). An evident instance can be observed at the initiation and the culmination of the example *D*, where *DeX* responses rapidly to the sudden transitions, while all the others exhibit a lingering effect of either a tail for the descent from high values to valleys or a delay for the ascent from valleys to high values. Although *DeX* only ranks the third place in terms of MAE with a slight performance dip, all of the aforementioned irreplaceable merits still serve to validate its uniqueness and overall distinction.

Indeed, the prediction of traffic extremes ends up with underwhelming performance in contrast to the totality, which could emanate from the prevalence of relatively stable throughput samples, that limits the model's ability to effectively learn patterns associated with critical values, echoing the dilemma in imbalanced ML scenarios [53].

Table 5
Result of ablation study.

Scenario		Ablation test 1[a]	Ablation test 2[b]	Ablation test 3[c]	Ablation test 4[d]	Ablation test 5[e]	Original *DeX*[f]
Overall values [Mbps]	MSE ↓	0.0458	0.0489	0.0478	0.0492	0.0530	0.0474
	MAE ↓	0.1108	0.1149	0.1149	0.1147	0.1230	0.1147
	MAPE ↓	10.710%	10.709%	10.437%	10.599%	11.998%	10.597%
	R^2 ↑	0.9285	0.9238	0.9254	0.9232	0.9174	0.9261
Peak values [Mbps]	MSE ↓	0.1567	0.1646	0.1646	0.1806	0.1529	0.1688
	MAE ↓	0.2327	0.2392	0.2417	0.2582	0.2322	0.2460
	MAPE ↓	12.075%	12.019%	12.292%	12.235%	11.526%	12.263%
	R^2 ↑	0.8388	0.8306	0.8307	0.8142	0.8428	0.8263
Valley values [Mbps]	MSE ↓	0.0239	0.0276	0.0242	0.0258	0.0413	0.0245
	MAE ↓	0.0754	0.0815	0.0780	0.0747	0.0989	0.0777
	MAPE ↓	19.001%	18.526%	17.605%	18.187%	22.557%	17.669%
	R^2 ↑	0.7030	0.6574	0.6996	0.6790	0.4864	0.6948
Abrupt changes [Mbps]	MSE ↓	0.2646	0.2774	0.2670	0.2768	0.2890	0.2645
	MAE ↓	0.3621	0.3707	0.3601	0.3662	0.3771	0.3574
	MAPE ↓	36.548%	37.216%	35.509%	37.217%	39.419%	36.057%
	R^2 ↑	0.6505	0.6337	0.6473	0.6344	0.6184	0.6507

[a] Refer to the entire 2048 packets that precede the targets as features without packet selection.

[b] Train the model with the 1024 packets closest to targets, devoid of packet selection.

[c] Train the model with 1024 packets subject to a different distribution of predefined probabilities for packet selection.

[d] Replace the Transformer encoder with an LSTM neural network for feature extraction.

[e] Remove the multi-task learning pipeline, and only reserve the regression block.

[f] Indicate the identical result retrieved from Table 4 to facilitate immediate comparisons.

Additionally, given that certain discrepancies among the reported metrics are subtle, we also conduct pair-wise t-tests with a confidence level of 95% between the errors generated by comparative models and those produced by *DeX*, in order to statistically and rigorously examine the disparities among predictions. In fact, the resulting p-values are exceedingly negligible with a magnitude lower than 10^{-4}, decisively rejecting the null hypothesis and unequivocally indicating significant differences with respect to the predictions of *DeX*.

6. Ablation study

In order to comprehend the impact of individual components and specific designs, we perform a series of ablation tests to substantiate their respective contributions. We first focus on packet selection module to conduct 3 sets of experiments, eliminating the module to assess the performance with either the complete ensemble of 2048 packets or the hithermost 1024 packets, and retaining the module but to train the intact model with an alternative predetermined distribution for packet selection. Following the architectural sequence of *DeX*, we then substitute the Transformer-based feature extraction block with an LSTM-based one, and at last, we retrain the model without multi-task learning pipeline. All results are shown in Table 5, alongside the original performance metrics derived from the unaltered architecture. In general, *DeX* surpasses the outcomes from ablation tests 2, 4, and 5, where the defective model structure with the equivalent amount of features is considered, achieving the superior overall performance notwithstanding occasional minor exceptions.

6.1. Ablation test 1 - Training with the entire 2048 packets

Firstly, we consider all of the 2048 ($N = W = 2048$) packets preceding target samples to feed the following prediction components so as to probe the upper limit of *DeX*'s capability and shed light on the potential performance degradation associated with fewer packets. As corroborated by the result (1st column in the table), *DeX* successfully reaches a prominent performance with 2048 packets at our disposal, exemplified by the only model boasting R^2 scores greater than 0.92, 0.83, 0.70, and 0.65, simultaneously, for the entirety, peaks, valleys, and abrupt changes, respectively. Concurrently, this supremacy also reinforces the pivotal role played by the packet selection mechanism, whose indispensability becomes evident, as it would be unreasonable to contemplate a reduction in packets, starting from the initial quantity of 2048, if we could not attain peak performance when endowed with the entire historical features. Indeed, *DeX* does encounter effectiveness decline as we curtail the amount of features, but the performance degradation is almost negligible with respect to the original 1024 packets, verifying the exceptional ability of *DeX* to utilize the packet selection module to adeptly grasp long-term knowledge while preserving salient short-term insights.

6.2. Ablation test 2 - Training with the nearest 1024 packets

To scrutinize the efficacy of the packet selection module, we opt for its omission, relying instead on an equivalent subset of 1024 packets that closely proximate the target samples (i.e., the hypothetically optimal selection). According to the 2nd column in the result, *DeX* registers a marginal performance decrement due to the absence of packet selection, especially for valley values and abrupt changes, e.g., a degradation of 0.0374 in R^2 score, in spite of the slight improvement for peaks. The reason behind lies in the fact that the hypothetically optimal (closest) packets primarily account for the immediate, short-term influence exerted by recently transpired packets, neglecting the incorporation of long-term dependencies. Conversely, the packet selection module is engineered to take into account both aspects, introducing the impact of relatively distant packets as well as preserving a substantial degree of adjacency (which is elaborated upon in Section 7.2.1). Although the performance enhancement thanks to packet selection may not be substantial, we nonetheless perceive its significance. This is because the module is excised after model training, thereby engendering an equivalence in the practical complexity between models featuring the selected or the nearest 1024 packets.

6.3. Ablation test 3 - Training with a different distribution of predefined probabilities

As heretofore mentioned, the predefined probability (score) of the hypothetically optimal selection merely serves as an intermediary tool to introduce a computational process so that the neural network can dynamically adjust trainable parameters and learn an efficacious subset of packets. In other words, the choice of the predefined distribution holds no paramount significance, so long as it aligns with our prerequisite of assigning higher probabilities (scores) to packets in proximity

Table 6
Result of parametric analysis 1: less input packets.

Model		*DeX*			
Scenario (number of packets)		2048[a]	1024[a]	512	256
Overall values [Mbps]	MSE ↓	0.0458	0.0474	0.0486	0.0557
	MAE ↓	0.1108	0.1147	0.1155	0.1294
	MAPE ↓	10.710%	10.597%	10.575%	11.902%
	R^2 ↑	0.9285	0.9261	0.9242	0.9131
Peak values [Mbps]	MSE ↓	0.1567	0.1688	0.1682	0.1993
	MAE ↓	0.2327	0.2460	0.2474	0.2865
	MAPE ↓	12.075%	12.263%	12.459%	15.211%
	R^2 ↑	0.8388	0.8263	0.8270	0.7950
Valley values [Mbps]	MSE ↓	0.0239	0.0245	0.0241	0.0207
	MAE ↓	0.0754	0.0777	0.0792	0.0694
	MAPE ↓	19.001%	17.669%	17.947%	17.185%
	R^2 ↑	0.7030	0.6948	0.6997	0.7420
Abrupt changes [Mbps]	MSE ↓	0.2646	0.2645	0.2756	0.2968
	MAE ↓	0.3621	0.3574	0.3686	0.3865
	MAPE ↓	36.548%	36.057%	36.273%	37.104%
	R^2 ↑	0.6505	0.6507	0.6360	0.6080

[a] The results are obtained from ablation test 1 with the entire set of 2048 packets without packet selection in Table 5, and the original *DeX* with the selection of 1024 packets in Table 4, for a straightforward comparison.

to targets. Thus, we persist in the strategy of choosing 1024 packets out of 2048, but adopt a different distribution of predefined probabilities for packet selection, aiming to reproduce the outcome and substantiate the underlying concept. Explicitly, we adhere to the same procedure in Section 4.2, but instead of creating a linear distribution, we envision an exponential one by appending an additional procedure at the end of step 1 to transform the linear space into a curvilinear one, i.e., $\theta_i \leftarrow e^{\theta_i}$. As a consequence (3rd column in the table), *DeX* with the predefined exponential distribution yields outcome akin to the original, despite trivial variations, indicating that the model is compatible with different configurations of predefined probabilities.

More importantly, the performance produced by the trio of aforementioned ablation tests, wherein the Transformer-based feature extraction and the multi-task learning pipeline always discharge their functions, remains decent, outstripping other comparative models in Table 4 as well, which in turn illustrates the prowess of other components of *DeX*.

6.4. Ablation test 4 - Training with LSTM-based feature extraction block

Hereafter, we maintain the bipolar components, yet implement an LSTM neural network to substitute the intermediate feature extraction block, aiming to verify the competence of Transformer. We employ 3 layers of LSTM unit and connect the hidden states of the last layer to the multi-task learning pipeline. Apparently, extracting features with LSTM falls short in both of traffic entirety and extremities (4th column in the table) comparing to the Transformer-based one in most cases. Interestingly, the performance is found to substantially transcend that of the vanilla LSTM with packet-level features (Table 4, LSTM-*ii*), demonstrating the *DeX*'s applicability as well as transportability to a certain extent, and further consolidating the proficiency embodied in other components.

6.5. Ablation test 5 - Training without multi-task learning

Finally, we explore the scenario where the last component is excised, leaving behind solely the regression task, while discarding the other two. As attested by the 5th column in the table, the basic model without multi-task learning experiences a substantial performance drop, which particularly reflects in valley values with an R^2 score of merely 0.4864, thus reinstating the model's performance to a level on par with other comparative methods. It is worth noting that we inadvertently reach the best performance for peak values up to now, implying that the deployment of Transformer encoder in concert with packet-level features intrinsically advocates for traffic peaks, and the employment of multi-task learning comes with a concomitant expense, sacrificing somewhat such performance to accommodate and compensate other categories of critical values.

7. In-depth analysis

This section is composed of three parts, a sequence of parametric analyses to investigate the universality and versatility of *DeX*, an attempt of model explainability that unravels the working logic, and an exploration of model practicability that elucidates the model overhead and the potential for further optimization.

7.1. Parametric analysis

We undertake 3 distinct sets of parametric analyses to comprehensively evaluate the performance of *DeX* and establish that the previously obtained outcomes are not a mere happenstance due to a specific model configuration. Notably, for the latter two sets of analyses, where the comparison against other baselines is needed, we confine our consideration to only three models (RF, XGB, N-BEATS) with comparative performance as elucidated by the initial experimental findings (Table 4).

7.1.1. Number of selected packets

We embark on an exploration of two additional scenarios for different numbers of input packets to reveal the possibility of further reducing the quantity of features. By halving the amount at a time, we consider the scenarios with $W = 512$ or 256 packets. Due to the substantial reduction in packet quantity, we opt to conduct the selection out of the 1024 nearest packets preceding targets, differing from the original case with 1024 out of 2048 packets to avoid excessively sparse selection, that may lead to a loss of crucial contextual information. According to the result in Table 6, selecting 512 packets still delivers comparable outcome without fundamentally compromising the performance even in contrast to 2048 packets, but 256 packets starts to exhibit a decline except for valley values (briefly explained in Section 7.2.1). Indeed, it is conceivable and inevitable to encounter performance setback with diminishing features. Nevertheless, the selection of only 512 packets constitutes merely a quarter of the initial quantity, yet still manages to attain results that consistently secure the highest rank among those in Table 4, further consolidating the efficacy of *DeX*, and opening up the possibility to reduce even more memory consumption, computational overhead, and model complexity.

7.1.2. Duration of predicted time window

In the following, we delve into the performance of predicting traffic throughput across distinct future time horizons, 300 and 1000 ms. In particular, we construct the datasets in a consistent way, computing throughput values within consecutive windows of the requisite time span, and for each time window (target throughput sample), we adhere to the usual practice of considering 2048 packets in the past for *DeX*, while referring to the historical throughput samples of the preceding 27 and 8 time windows[13] in the 300- and 1000-ms scenarios respectively for time series models. Additionally, all models are subjected to retraining from scratch due to the modification of dataset.

Table 7 presents all the results regarding different predicted time windows for *DeX* and other models. Overall, *DeX* still dominates the board with three minor exceptions in terms of MAPE, irrespective of the duration, justifying its versatility and robustness. Comparing all

[13] The choice of such amount of time windows mirrors our previous approach, wherein 2048 packets translate to a duration of approximately 8 s, i.e., 27×300 ms ≈ 8 s and 8×1000 ms $= 8$ s.

Table 7
Result of parametric analysis 2: different duration (Δt) for predicted time window.

Scenario (duration of predicted window, Δt)		300 ms				1000 ms			
Model		RF	XGB	N-BEATS	*DeX*	RF	XGB	N-BEATS	*DeX*
Overall values [Mbps]	MSE ↓	0.0589	0.0580	0.0567	**0.0538**	0.0490	0.0505	0.0512	**0.0457**
	MAE ↓	0.1347	0.1335	0.1275	**0.1260**	0.1084	0.1126	0.1080	**0.1057**
	MAPE ↓	13.794%	13.579%	**11.453%**	11.547%	12.206%	13.260%	**10.291%**	10.872%
	R^2 ↑	0.9097	0.9111	0.9131	**0.9177**	0.9216	0.9191	0.9180	**0.9269**
Peak values [Mbps]	MSE ↓	0.2291	0.2223	0.2153	**0.2010**	0.1335	0.1378	0.1432	**0.1172**
	MAE ↓	0.3192	0.3134	0.3023	**0.2881**	0.2154	0.2227	0.2224	**0.1927**
	MAPE ↓	14.659%	14.480%	14.439%	**14.022%**	10.459%	11.262%	10.888%	**9.897%**
	R^2 ↑	0.7713	0.7780	0.7850	**0.8003**	0.8478	0.8429	0.8367	**0.8667**
Valley values [Mbps]	MSE ↓	0.0531	0.0540	0.0478	**0.0410**	0.0359	0.0372	0.0293	**0.0233**
	MAE ↓	0.1209	0.1234	0.1071	**0.1015**	0.0865	0.0910	0.0707	**0.0688**
	MAPE ↓	31.547%	31.732%	24.342%	**23.133%**	29.078%	32.553%	**19.609%**	24.854%
	R^2 ↑	0.3791	0.3684	0.4407	**0.5270**	0.5545	0.5385	0.6359	**0.7174**
Abrupt changes [Mbps]	MSE ↓	0.2190	0.2175	0.2185	**0.2119**	0.3259	0.3297	0.3553	**0.3019**
	MAE ↓	0.3159	0.3135	0.3093	**0.3085**	0.4275	0.4332	0.4517	**0.4041**
	MAPE ↓	34.411%	34.072%	30.806%	**30.417%**	57.281%	58.422%	50.060%	**49.754%**
	R^2 ↑	0.7167	0.7187	0.7174	**0.7263**	0.5144	0.5087	0.4706	**0.5518**

Table 8
Result of parametric analysis 3: different thresholds ($\alpha_p, \alpha_v, \beta$) for defining traffic extremes.

Model		RF	XGB	N-BEATS	*DeX*	RF	XGB	N-BEATS	*DeX*
Different thresholds for peak & valley		Top and lowest 15% throughput values ($\alpha_p = 15\%, \alpha_v = 15\%$)				Top and lowest 20% throughput values ($\alpha_p = 20\%, \alpha_v = 20\%$)			
Peak values [Mbps]	MSE ↓	0.1381	0.1455	0.1380	**0.1287**	0.1138	0.1200	0.1141	**0.1065**
	MAE ↓	0.2271	0.2305	0.2222	**0.2093**	0.1994	0.2036	0.1959	**0.1856**
	MAPE ↓	11.387%	11.649%	11.337%	**10.946%**	10.347%	10.666%	10.347%	**10.114%**
	R2 ↑	0.8478	0.8397	0.8479	**0.8592**	0.8685	0.8613	0.8682	**0.8776**
Valley values [Mbps]	MSE ↓	0.0293	0.0277	0.0277	**0.0221**	0.0311	0.0303	0.0308	**0.0245**
	MAE ↓	0.0832	0.0753	0.0744	**0.0723**	0.0842	0.0781	0.0776	**0.0747**
	MAPE ↓	19.375%	15.633%	16.041%	**14.808%**	17.622%	14.420%	14.823%	**13.703%**
	R2 ↑	0.6955	0.7116	0.7115	**0.7702**	0.7407	0.7473	0.7430	**0.7958**
Different thresholds for abrupt change		Inter-variation ≥ 15% ($\beta = 15\%$)				Inter-variation ≥ 10% ($\beta = 10\%$)			
Abrupt changes [Mbps]	MSE ↓	0.1988	0.2060	0.2074	**0.1875**	0.1285	0.1335	0.1340	**0.1213**
	MAE ↓	0.3020	0.3042	0.3087	**0.2894**	0.2267	0.2285	0.2311	**0.2173**
	MAPE ↓	30.940%	29.648%	29.839%	**27.683%**	22.701%	21.411%	21.627%	**20.186%**
	R2 ↑	0.7356	0.7260	0.7242	**0.7508**	0.8221	0.8151	0.8145	**0.8322**

conditions including the original 500 ms, several insightful observations emerge: (i) The 300-ms case fails in predicting peaks and valleys, e.g., *DeX* produces an R^2 score of only 0.5270 for valleys, and RF and XGB cannot even reach 0.4. Conversely, there is a notable improvement in performance concerning abrupt changes, with all models exhibiting MAPEs lower than 35% and R^2 scores exceeding 0.7 for the first time. This could be rooted in the more frequent updates of throughput values, which promptly and timely inform the dynamic traffic fluctuations. Yet, it could be for the same reason that traffic aggregated into shorter windows exhibits higher volatility, leading to elusive patterns and thus overall diminished performance. (ii) In the 1000-ms scenario, *DeX* maintains its preeminence with further enhancement for peaks and valleys (e.g., R^2 scores rise from 0.8263 and 0.6948 to 0.8667 and 0.7174), but noticeable degradation for abrupt changes (e.g., an escalation in MAPE from 36.057% to 49.754%). Both observations can originate from the same reason of a longer predicted time window, which smooths out traffic variations and stabilizes throughput evolution. Such an effect facilitates the forecasting of peaks and valleys, thanks to the reduction of localized fluctuations, but renders the abrupt changes even more sudden and unexpected, as the gradually escalated patterns of traffic throughput vanish, significantly hampering the prediction. Nonetheless, *DeX* still yields the best outcome (e.g., an improvement of 0.0374 in R^2 score comparing to the second place), not to mention the overall decent performance and the longer time horizon, which affords a higher degree of freedom for system management to implement optimized policies.

7.1.3. Different thresholds for defining extremes

Previously, traffic extremities are defined by specific thresholds to optimize the model performance, and thus, we now elaborate on 4 extra conditions, varying the thresholds from the highest and lowest 10% values to 15% or 20% ($\alpha_{p,v} = 10\% \Rightarrow \alpha_{p,v} = 15\%$ or 20%) for peaks and valleys, and from inter-variations greater than 20% to 15% or 10% ($\beta = 20\% \Rightarrow \beta = 15\%$ or 10%) for abrupt changes. Simply put, we broaden our experiment to encompass more samples as traffic extremes to also examine the performance for sub-critical values. As expected, *DeX* still excels the others, regardless of variations in thresholds according to Table 8. The performance excess generated by *DeX* remains relatively stable across different thresholds for peaks and abrupt changes (e.g., the advantage in R^2 score for peaks with respect to RF is 0.0123 when $\beta = 20\%$, and is 0.0091 when $\beta = 10\%$), while it declines for valleys to an acceptable extend (e.g., the difference in R^2 score comparing to N-BEATS is 0.0862 for $\alpha_v = 10\%$, but decreases to 0.0528 for $\alpha_v = 20\%$), underscoring the consistent versatility and comprehensiveness of *DeX*, which is not only reliant on the predominance of performance regarding the original predefined thresholds.

7.2. Model explainability

Herein, we showcase several outcomes of the packet selection module and the multi-task learning pipeline, elucidating their operational mechanisms.

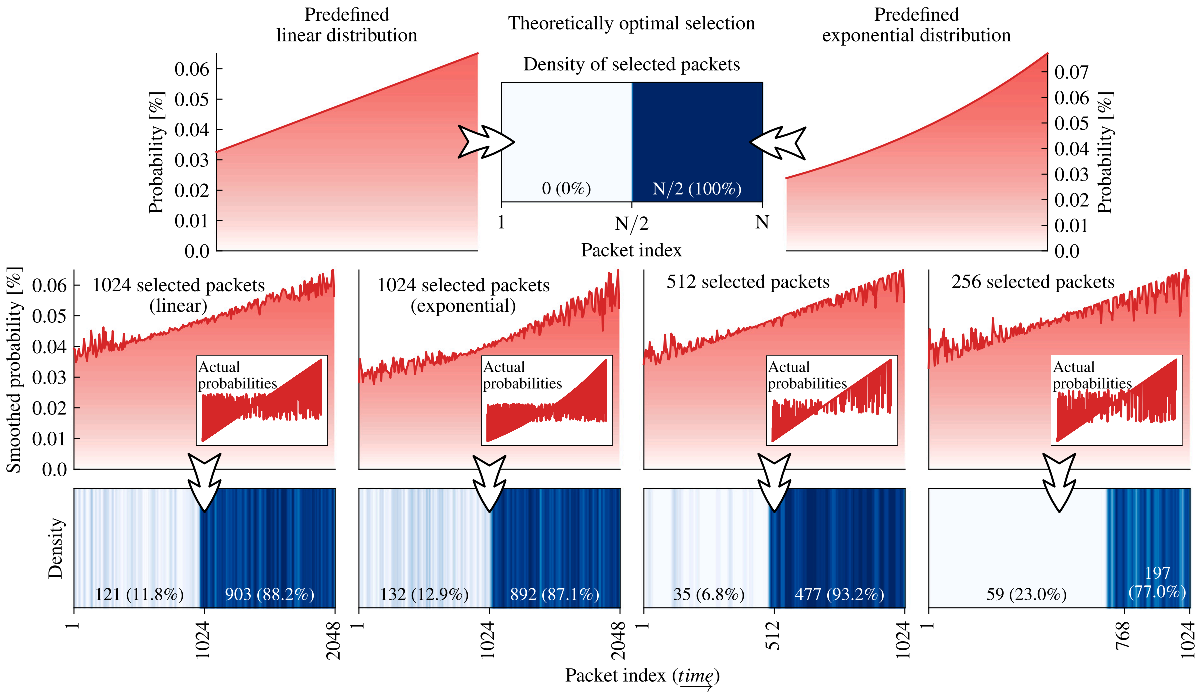


Fig. 9. Outcomes of packet selection.

7.2.1. Packet selection module

The derived packet selections across various scenarios involving different quantity of selected packets (initial 1024 packets in Section 5, modified 1024 packets with different predefined probabilities in Section 7.1.1, and 512 as well as 256 packets in Section 7.1) are presented in Fig. 9, in which the uppermost 3 graphs depict the 2 aforementioned hypothetically optimal selections as baselines, the quartet of reddish figures below illustrate the smoothed distribution of resulting selection probabilities along the packet indices, showing the average probability value for every 8 packets, and the bottom 4 bluish figures elucidate the density of the corresponding packet selections, with darker regions signifying heightened packet density.[14]

With regard to the probability of packet selection, it is discernible that convergence towards the hypothetically optimal probabilities persists consistently across the various scenarios, despite minor divergences in alignment. The integrated packet selection module enables *DeX* to meticulously strike a balance between optimally choosing input features and modulating the trainable parameters to generate Softmax probabilities towards the predefined ones, resulting in higher scores even for packets situated distally from the target. Indeed, the majority of selections are still allocated in temporal proximity to the prediction target, notwithstanding the broader distribution. Furthermore, the three cases of 1024 and 512 packets share a similar pattern of picking a portion of packets in the farside, which is likely attributed to the fact that they all occupy half of the original packet count. In contrast, no remote packets are elected in the case of 256 packets, and even the 59 packets absent in the target's immediate vicinity (256 nearest ones) are still positioned closed to the boundary. This could originate from the exiguous 256 packets per se, which lack of sufficient margin for potential packets in distance. Moreover, alongside the performance indicated in Section 7.1.1, where 256 packets output suboptimal performance with an exception for valleys, we can safely deduce that the long-term packet features indeed foster the overall prediction, albeit with some hindrance in identifying valleys. Of noteworthy concern, although both instances of 1024 packets produce analogous performance, the ensuing packet selections are different. The packet selection module does not seek to pinpoint the absolutely and deterministically optimal locations of packets, but aim to uncover the practically effective packets that are suitable for a specific model training. Therefore, while dissimilar packets are selected based on distinct predefined probabilities, they still make comparable short- as well as long-term contributions, due to their respective roles during the associated model development.

7.2.2. Multi-task learning pipeline

Fig. 10 showcases 4 time series of traffic throughput in terms of both ground truth and the corresponding predictions, with and without intervention of trainable multiplier, to exemplify the role of the multi-task learning pipeline in improving the regression outputs when it comes to abrupt changes.[15] Accordingly, trainable multipliers can either magnify or compress the primordial regression outcomes, bringing them into closer alignment with the ground truth, especially in example 1, where the adjusted prediction is almost identical to the true value. Beyond the specific points of focus within the figures, a more profound instance is demonstrated at the tail end of traffic example 4, where the throughput experiences a sudden increase, but the unaltered prediction veers in the opposite direction. Remarkably, the trainable

[14] Notes: (i) The two-sided figures on top present the hypothetically optimal selection probabilities (linear one in Eq. (4) and exponential one in Section 7.1.1), and consequently, the closest half packets ($N/2$) to the target are selected (top central figure); (ii) The smoothed probabilities on bottom are for a clearer visualization, while the original ones are portrayed in the sub-figures; (iii) The lowermost figures also indicate the specific number of selected packets and their occupation in different regions, which are marked based on the target number of packet selection, e.g., in the case of selecting 256 packets from a pool of 1024, 768 packets are allocated to the farside while 256 packets are in the nearside.

[15] Notes: (i) The presence of negative throughput samples is a consequence of the displayed figures showcasing the raw values emanating from the neural network, subsequently subjected to an inverse scaling procedure to replicate the actual throughput; (ii) The magnified sub-figures explicitly indicate the magnitude of the multiplier, that modulates the original regression output to align with the ground truth.

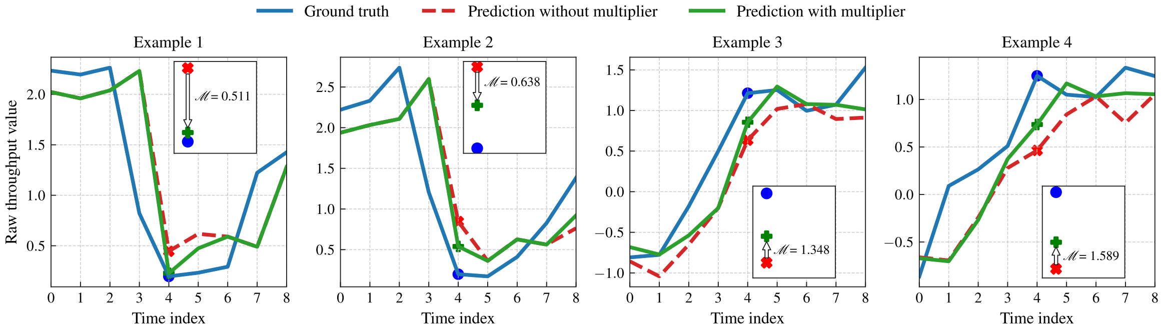


Fig. 10. Examples of how the trainable multiplier operates.

multiplier discerns such an erroneous behavior as well as the abrupt surge, calibrating the prediction to attain a performance level deemed acceptable. Although it is barely possible to response instantaneously to the very first abrupt change, the prompt and precise adaptations still demonstrate the effectiveness of *DeX*.

7.3. Model practicability

Given the real-time nature and the restricted time window, the time consumed by *DeX* for predictions stands as a critical consideration. While examining the implementation in reality remains challenging at present, we maintain confidence in the practicability and feasibility of the model. To provide context, we investigate the time needed for two versions of *DeX* to execute a single prediction across three different levels of CPU environments devoid of GPU acceleration. Based on the findings presented in Table 9, both models demonstrate acceptable consumption irrespective of the CPU, leaving at least four-fifths of the time available within the 500-ms window. Notably, *DeX* of 512 packets with a comparable performance merely requires 23 ms even in the worst case scenario. We envision sufficient temporal margin to employ possible optimization strategies, thereby validating the model practicability, let alone the moderately prolonged case of the 1000-ms time window.

Furthermore, we also provide insights into the model complexity regarding the number of parameters shown also in Table 9. At a first glance, both models exhibit a significant amount of parameters, but actually the majority (93.3% and 80.8%) are occupied by the FNNs in the multi-task learning pipeline, which can be readily customized to reduce the parameter quantity and enhance efficiency. In this regard, we curtail the size of the multi-task learning pipeline for *DeX* with 1024 packets, decreasing the number of neurons in the first layer of the FNNs in all tasks from 512 to 256, and consequently obtaining a reduction of roughly 47% in the total parameter count. We then evaluate the model on the test set and present the result in Table 10. Evidently, the trimmed model successfully achieves comparable performance to the original *DeX*, with only a slight decline for valley values, illustrating the potential and possibility for further improvement of model efficiency. Moreover, it is noteworthy that the aforementioned analyses do not factor in any existing technical optimizations, including well-established pruning technologies [54] and efficient Transformer architectures [55], which could further bolster the model efficacy and practicality.

8. Related work

The realm of computer networks and communications has witnessed an enrichment through the integration of ML and DL technologies in recent years [56–59], and the domain of RTC, encompassing traffic classification, adaptive management, performance improvement, etc., has not been overlooked [60–65]. Herein, we provide an overview of relevant literature pertaining to throughput and packet-level-based prediction.

Table 9
Time consumption for *DeX* to make a prediction in CPU environments.

Model	*DeX* 1024[a]	*DeX* 512[b]
Number of parameters	1.69M	488.65K
Server tier Intel Xeon Gold 6140	24 ms ± 307 μs	3.82 ms ± 195 μs
High-performance tier Apple M2	42.3 ms ± 280 μs	11.9 ms ±57.3μs
Mid-range consumer tier AMD Ryzen 7 PRO 4750U	90.6 ms ± 550 μs	23 ms ± 287 μs

[a] The original model with 1024 selected packets.
[b] The modified model with 512 selected packets.

Table 10
Experimental result of *DeX* with less parameters.

Scenario	MSE	MAE	MAPE	R^2
Overall values [Mbps]	0.0459	0.1118	10.615%	0.9284
Peak values [Mbps]	0.1684	0.2547	12.679%	0.8267
Valley values [Mbps]	0.0267	0.0818	19.300%	0.6674
Abrupt changes [Mbps]	0.2562	0.3515	36.359%	0.6616

Throughput prediction, akin to bandwidth or bitrate prediction, has gained considerable traction in academia. The work of [66] aimed to improve the selection of bitrate by predicting throughput based on data-driven approach. The authors analyzed millions of sessions, finding out similarities and stateful patterns, which were used to cluster sessions and develop a Hidden-Markov-Model (HMM) predictor. [67] payed attention to the average throughput prediction in cellular networks, exploiting a Random Forest (RF) [46] regressor based on a range of radio channel metrics and throughput measurements. They then extended the work in [68] by examining two more technologies, Support Vector Machine (SVM) [69] and Long Short-Term Memory (LSTM) [50] NN. Authors in [70] proposed a cross-layer solution to enhance transmission quality, particularly for video calls within cellular networks. Their approach hinged upon a linear adaptive filter, Recursive Least Squares (RLS) [45], to forecast forthcoming bandwidth based on historical capacities. In [71], a RF-based ML framework, namely LinkForest, was introduced to predict cellular link bandwidth in 4G Long Term Evolution (LTE) networks. In addition to historical throughput data, the authors also considered lower-layer metrics, e.g., Reference Signal Received Power (RSRP), as input features. The authors in [72] utilized public datasets of general Internet traffic to perform short-term bandwidth prediction. They adopted multiple ML algorithms, ranging from tree-based models to Deep Neural Network (DNN), and aggregated packets into time windows, in which various features, such as cumulative bitrate and number of packets, are computed. As a result, RF turned out to be a promising approach. Furthermore, the LSTM Recurrent Neural Networks (RNN) were implemented in [73,74] with customized designs. In the former, the authors investigate real-time mobile bandwidth prediction across diverse mobile network conditions. Their proposed LSTM model was pre-trained

Table 11
Related works for throughput prediction and their adopted technologies..

Paper	Methodologies[a]
Y. Sun *et al.,* 2016, *ACM SIGCOMM* [66]	HMM
D. Raca *et al.,* 2019, *ACM MMSys* [67]	RF
D. Raca *et al.,* 2020, *IEEE MCOM* [68]	RF, SVM, LSTM
E. Kurdoglu *et al.,* 2016, *ACM MMSys* [70]	RLS
C. Yue *et al.,* 2017, *IEEE TMC* [71]	RF
M. Labonne *et al.,* 2021, *IEEE WoWMoM* [72]	DT, RF, XGB, DNN
L. Mei *et al.,* 2020, *Elsevier ComNet* [73]	LSTM, HM, RLS
A. Lekharu *et al.,* 2020, *IEEE COMSNETS* [74]	LSTM, 1D-CNN
A. Bentaleb *et al.,* 2022, *IEEE TMM* [75]	DRL
G. Lv *et al.,* 2022, *IEEE INFOCOM* [76]	DT, MLR
B Wei *et al.,* 2019, *IEEE Access* [77]	LSTM, MA, HMM, AM HM, LS, Stochastic
J. Yin *et al.,* 2021, *arXiv* [78]	CNN+DNN
A. Sobhani *et al.,* 2017, *ACM TOMM* [79]	KAMA

[a] The unmentioned acronyms: DT – Decision Tree, HM – Harmonic Mean, MLR – Multiple Linear Regression, AM – Arithmetic Mean, LS – Last Sample.

and subsequently, they augmented the framework by employing model switching and Bayes model fusion for online deployment. In the latter, the work aimed at improving adaptive video streaming by predicting bitrates to optimize QoE metrics. The authors proposed a reinforcement learning (RL) paradigm, wherein an LSTM combined with a Convolutional Neural Network (CNN) assumed responsibility for bitrate prediction. Unlike the former work that relied on historical bandwidth alone, they introduced multiple features, such as chunk throughput and size. Differently, [75] directly applied RL technique, proposing a hybrid bandwidth prediction solution, namely BoB, which was trained offline based on network traces, and implemented online to support a controller on the receiver side. Meanwhile, the focus of [76–79] was centered on Adaptive Bitrate (ABR) for HTTP-based video streaming. The first two works adopted Tree-based or DL-based models for throughput prediction, integrating the algorithms into ABR decision-making process to optimize QoE. The penultimate work intended to use K-means to cluster network conditions, measuring the corresponding feature vectors and leveraging DNN to capture temporal dynamics, which were then integrated into renowned RL-based ABR decision engine, while the last one utilized a Kaufman's Adaptive Moving Average (KAMA) [80] method to predict bitrate for a fuzzy-logic controller to dynamically manage video rate and provide decision-making for video segment downloading. To summarize, we present the related works with considered methodologies in Table 11.

Regarding packet-level prediction, that is not confined to RTC, a rather limited corpus of research exists. Authors in [81] capitalized on multi-task DL approach to not only utilize packet-level information but also to predict packet-level characteristics, such as packet direction and payload length. They investigated multiple DL techniques as the backbone model, and compared the performance against Markov chain and Random Forest (RF) regressor. Specifically, packets featuring three predicted and three exogenous parameters (e.g., TCP window size) were arranged in a sequential manner to enable sliding window prediction. The study further provided intensive analyses of different types of traffic and ML models. Moreover, the Transformer architecture appeared in both [82,83]. The first work proposed FlowFormer to classify real-time network flow types (video, conference, and download). Rather than adhering strictly to the original Transformer architecture, the authors implemented multiple layers of attention-based encoder to extract features to then feed to a LSTM or a CNN model. Notably, packet-level attributes such as payload length were tracked and compared against predefined thresholds to be aggregated into corresponding chronological bins, whereby the quantities of packets in each bin were calculated as features. The second work sought to model and generalize network dynamics by exploiting the power of Transformer structure grounded in packet-level information, e.g., timestamps. The authors proposed the so-called Network Traffic Transformer (NTT) framework, in which the general architecture of Transformer was implemented except that they incorporated a hierarchical aggregation layer preceding the encoder to condense lengthy packet sequence, concatenating summarized older packets with the most recent ones. The model was initially pre-trained based on an end-to-end delay prediction, and the authors envisioned a replaceable decoder for other potential tasks.

To the best of our knowledge, our work stands as a pioneering effort in employing Transformer-based architecture in conjunction with packet-level information to predict throughput in RTC, laying focus on traffic extremes to bolster predictive performance. To thoroughly contextualize the novelty and significance of our model, we opt for a total number of 9 technologies that manifest superiority in the literature. Noteworthily, the methodology adopted in [76] is indeed able to discern abrupt changes thanks to the feature of chunk size, which, however, is unavailable in RTP-based traffic. Our model boasts a streamlined design of the architecture, efficiently and astutely harnessing a minimal set of packet-level information as features. Consequently, the need for resource-intensive processes, such as intricate feature extraction, exhaustive aggregation, and complicated calculations is eliminated, contributing to the lightweight nature of our proposed framework.

9. Conclusion

In this paper, our main objective is to predict the RTC traffic throughput, with a distinct emphasis on traffic extremes, encompassing peaks, valleys, and abrupt changes. We propose a novel DL framework named *DeX*, exclusively utilizing RTP packet-level information with the merit of ease of feature extraction, and consisting of three sophisticated devised components: a packet selection module that optimally extracts informative subset and reduces the total amount of input features, facilitating computational efficiency, a feature extraction block that partially incorporates a Transformer-based architecture to discern traffic intricacy, and a multi-task learning pipeline that is meticulously designed to improve the adeptness and adaptability for extremities. To consolidate the model universality and resilience, we anchor our work on ample RTC traffic collected under various conditions, compare our model against numerous technologies, and elaborate on diverse scenarios, undertaking thorough ablation studies and parametric analyses. As a result, we obtain satisfactory overall performance with preeminent outcomes for traffic extremes. Future work could involve endeavors to further reduce the model complexity, and additionally, we remain receptive to the prospect of incorporating exogenous factors, such as the router queue length when predictions are executed at the edge node, potentially continuing to enhance the performance. Furthermore, in order to further justify the generalizability of *DeX*, our work could also benefit from a more rigorous and systematic data collection campaign with a broader range of RTC applications.

CRediT authorship contribution statement

Tailai Song: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Paolo Garza:** Conceptualization, Project administration, Supervision, Writing – review & editing. **Michela Meo:** Writing – review & editing, Supervision, Resources, Project administration, Conceptualization. **Maurizio M. Munafò:** Writing – review & editing, Supervision, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

I have shared the link to my data in the paper.

Acknowledgments

This work has been supported by the SmartData@PoliTO center for BigData and Data Science, Cisco Systems Inc, and the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 — program "RESTART", focused project R4R).

References

[1] R. Frederick, S.L. Casner, V. Jacobson, H. Schulzrinne, RTP: A transport protocol for real-time applications, in: Request for Comments, (1889) RFC Editor, 1996, http://dx.doi.org/10.17487/RFC1889, URL https://rfc-editor.org/rfc/rfc1889.txt.

[2] C. Athanasiadou, G. Theriou, Telework: systematic literature review and future research agenda, Heliyon 7 (10) (2021) e08165.

[3] A. Nistico, D. Markudova, M. Trevisan, M. Meo, G. Carofiglio, A comparative study of RTC applications, in: 2020 IEEE International Symposium on Multimedia, ISM, IEEE, 2020, pp. 1–8.

[4] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, P. Halvorsen, Video streaming using a location-based bandwidth-lookup service for bitrate planning, ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) 8 (3) (2012) 1–19.

[5] J.R. Wilcox, Videoconferencing: The whole picture, Taylor & Francis, 2017.

[6] C. Liang, M. Zhao, Y. Liu, Optimal bandwidth sharing in multiswarm multiparty p2p video-conferencing systems, IEEE/ACM Trans. Netw. 19 (6) (2011) 1704–1716.

[7] B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, G. Zussman, Performance evaluation of webrtc-based video conferencing, ACM SIGMETRICS Perform. Eval. Rev. 45 (3) (2018) 56–68.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, Hypertext transfer protocol–HTTP/1.1, 1999.

[10] J. Postel, User datagram protocol, 1980, https://tools.ietf.org/html/rfc768. (Accessed 16 November 2023).

[11] O. Said, Y. Albagory, M. Nofal, F. Al Raddady, IoT-RTP and IoT-RTCP: Adaptive protocols for multimedia transmission over internet of things environments, IEEE Access 5 (2017) 16757–16773.

[12] S. Loreto, S.P. Romano, Real-time communication with WebRTC: peer-to-peer in the browser, O'Reilly Media, Inc., 2014.

[13] T. Sharma, T. Mangla, A. Gupta, J. Jiang, N. Feamster, Estimating WebRTC video QoE metrics without using application headers, 2023, arXiv preprint arXiv:2306.01194.

[14] M. Alahmadi, P. Pocta, H. Melvin, An adaptive bitrate switching algorithm for speech applications in context of webrtc, ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) 17 (4) (2021) 1–21.

[15] Z. Zhang, H. Chen, X. Cao, Z. Ma, Anableps: Adapting bitrate for real-time communication using VBR-encoded video, in: 2023 IEEE International Conference on Multimedia and Expo, ICME, IEEE, 2023, pp. 1685–1690.

[16] G. Perna, D. Markudova, M. Trevisan, P. Garza, M. Meo, M.M. Munafò, G. Carofiglio, Real-time classification of real-time communications, IEEE Trans. Netw. Serv. Manag. 19 (4) (2022) 4676–4690.

[17] T. Song, D. Markudova, G. Perna, M. Meo, Where did my packet go? Real-time prediction of losses in networks, in: ICC 2023-IEEE International Conference on Communications, IEEE, 2023, pp. 3836–3841.

[18] D. Markudova, M. Meo, Rococo: Reinforcement learning-based congestion control for real-time applications, in: 2023 IEEE 24th International Conference on High Performance Switching and Routing, HPSR, IEEE, 2023, pp. 68–74.

[19] Y. Bandung, L.B. Subekti, D. Tanjung, C. Chrysostomou, Qos analysis for webrtc videoconference on bandwidth-limited network, in: 2017 20th International Symposium on Wireless Personal Multimedia Communications, WPMC, IEEE, 2017, pp. 547–553.

[20] N.M. Edan, A. Al-Sherbaz, S. Turner, Performance evaluation of resources management in webrtc for a scalable communication, in: Intelligent Computing: Proceedings of the 2018 Computing Conference, Volume 2, Springer, 2019, pp. 648–665.

[21] S. Huang, J. Xie, DAVE: Dynamic adaptive video encoding for real-time video streaming applications, in: 2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking, SECON, IEEE, 2021, pp. 1–9.

[22] L. Liu, J. Li, H. Xu, K. Xue, J.C. Xue, Efficient real-time video conferencing with adaptive frame delivery, Comput. Netw. 234 (2023) 109918.

[23] Y. Li, Z. Zhang, H. Chen, Z. Ma, Mamba: Bringing multi-dimensional ABR to webrtc, in: Proceedings of the 31st ACM International Conference on Multimedia, 2023, pp. 9262–9270.

[24] G. Carlucci, L. De Cicco, S. Holmer, S. Mascolo, Analysis and design of the google congestion control for web real-time communication (webrtc), in: Proceedings of the 7th International Conference on Multimedia Systems, 2016, pp. 1–12.

[25] E. Torres, R. Reale, L. Sampaio, J. Martins, A SDN/OpenFlow framework for dynamic resource allocation based on bandwidth allocation model, IEEE Latin Am. Trans. 18 (05) (2020) 853–860.

[26] R.A. Kirmizioglu, A.M. Tekalp, Multi-party webrtc services using delay and bandwidth aware SDN-assisted IP multicasting of scalable video over 5G networks, IEEE Trans. Multimed. 22 (4) (2019) 1005–1015.

[27] Intel, DPDK: Data Plane Development Kit, 2018, URL http://dpdk.org/. (Accessed 6 March 2024).

[28] PcapPlusPlus, PcapPlusPlus, 2018, Available online: https://pcapplusplus.github.io/. (Accessed March 2024).

[29] J. Sonchack, O. Michel, A.J. Aviv, E. Keller, J.M. Smith, Scaling hardware accelerated network monitoring to concurrent and dynamic queries with {*flow}, in: 2018 USENIX Annual Technical Conference, USENIX ATC 18, 2018, pp. 823–835.

[30] Y. Go, M.A. Jamshed, Y. Moon, C. Hwang, K. Park, {APUNet}: Revitalizing {GPU} as packet processing accelerator, in: 14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 17, 2017, pp. 83–96.

[31] NetLimiter, 2005, https://www.netlimiter.com/. (Last accessed 6 March 2024).

[32] NetBalancer, 2011, https://netbalancer.com/. (Last accessed 6 March 2024).

[33] A. Dainotti, A. Pescapé, P.S. Rossi, F. Palmieri, G. Ventre, Internet traffic modeling by means of hidden Markov models, Comput. Netw. 52 (14) (2008) 2645–2662.

[34] J. Uberti, C. Jennings, S. Murillo, RFC 9335: Completely encrypting RTP header extensions and contributing sources, 2023.

[35] B. Marczak, J. Scott-Railton, Move fast and roll your own crypto: A quick look at the confidentiality of zoom meetings, 2020, https://citizenlab.ca/2020/04/move-fast-roll-your-own-crypto-a-quick-look-at-the-confidentiality-of-zoom-meetings/.

[36] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A transport protocol for real-time applications, 2003, https://tools.ietf.org/html/rfc3550, RFC 3550.

[37] M. Maruschke, O. Jokisch, M. Meszaros, V. Iaroshenko, Review of the opus codec in a webrtc scenario for audio and speech communication, in: Speech and Computer: 17th International Conference, SPECOM 2015, Athens, Greece, September 20-24, 2015, Proceedings 17, Springer, 2015, pp. 348–355.

[38] J.-M. Valin, K. Vos, T. Terriberry, Definition of the opus audio codec, 2012.

[39] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.

[40] A.T. Liu, S.-W. Li, H.-y. Lee, Tera: Self-supervised learning of transformer encoder representation for speech, IEEE/ACM Trans. Audio Speech Lang. Process. 29 (2021) 2351–2366.

[41] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7482–7491.

[42] A.F. Agarap, Deep learning using rectified linear units (relu), 2018, arXiv preprint arXiv:1803.08375.

[43] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[44] R.J. Hyndman, Moving averages, 2011.

[45] A.H. Sayed, Fundamentals of adaptive filtering, John Wiley & Sons, 2003.

[46] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.

[47] T. Chen, T. He, M. Benesty, V. Khotilovich, Y. Tang, H. Cho, K. Chen, R. Mitchell, I. Cano, T. Zhou, et al., Xgboost: extreme gradient boosting, R Package Version 0.4-2 1 (4) (2015) 1–4.

[48] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press, 2016.

[49] G. Lai, W.-C. Chang, Y. Yang, H. Liu, Modeling long-and short-term temporal patterns with deep neural networks, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, 2018, pp. 95–104.

[50] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[51] B.N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, 2019, arXiv preprint arXiv:1905.10437.

[52] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019).

[53] H. Kaur, H.S. Pannu, A.K. Malhi, A systematic review on imbalanced data challenges in machine learning: Applications and solutions, ACM Comput. Surv. 52 (4) (2019) 1–36.

[54] T. Liang, J. Glossner, L. Wang, S. Shi, X. Zhang, Pruning and quantization for deep neural network acceleration: A survey, Neurocomputing 461 (2021) 370–403.

[55] Y. Tay, M. Dehghani, D. Bahri, D. Metzler, Efficient transformers: A survey, ACM Comput. Surv. 55 (6) (2022) http://dx.doi.org/10.1145/3530811.

[56] T. Cerquitelli, M. Meo, M. Curado, L. Skorin-Kapov, E.E. Tsiropoulou, Machine learning empowered computer networks, Comput. Netw. (2023) 109807.

[57] W. Samek, S. Stanczak, T. Wiegand, The convergence of machine learning and communications, 2017, arXiv preprint arXiv:1708.08299.

[58] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppänen, L. Loven, A. Anttonen, A.H. Sodhro, M.M. Alam, M. Juntti, et al., Machine learning meets communication networks: Current trends and future challenges, IEEE Access 8 (2020) 223418–223460.

[59] R. Boutaba, M.A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, O.M. Caicedo, A comprehensive survey on machine learning for networking: evolution, applications and research opportunities, J. Internet Serv. Appl. 9 (1) (2018) 1–99.

[60] G. Perna, D. Markudova, M. Trevisan, P. Garza, M. Meo, M.M. Munafò, G. Carofiglio, Online classification of RTC traffic, in: 2021 IEEE 18th Annual Consumer Communications & Networking Conference, CCNC, IEEE, 2021, pp. 1–6.

[61] D. Markudova, M. Trevisan, P. Garza, M. Meo, M.M. Munafo, G. Carofiglio, What's my app? ml-based classification of rtc applications, ACM SIGMETRICS Perform. Eval. Rev. 48 (4) (2021) 41–44.

[62] S. Cheng, H. Hu, X. Zhang, Z. Guo, Deeprs: Deep-learning based network-adaptive FEC for real-time video communications, in: 2020 IEEE International Symposium on Circuits and Systems, ISCAS, IEEE, 2020, pp. 1–5.

[63] Z. Wang, Y. Na, B. Tian, Q. Fu, NN3a: Neural network supported acoustic echo cancellation, noise suppression and automatic gain control for real-time communications, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2022, pp. 661–665.

[64] X. Jiang, X. Peng, C. Zheng, H. Xue, Y. Zhang, Y. Lu, End-to-end neural speech coding for real-time communications, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2022, pp. 866–870.

[65] J. Fang, M. Ellis, B. Li, S. Liu, Y. Hosseinkashi, M. Revow, A. Sadovnikov, Z. Liu, P. Cheng, S. Ashok, et al., Reinforcement learning for bandwidth estimation and congestion control in real-time communications, 2019, arXiv preprint arXiv:1912.02222.

[66] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, B. Sinopoli, CS2p: Improving video bitrate selection and adaptation with data-driven throughput prediction, in: Proceedings of the 2016 ACM SIGCOMM Conference, 2016, pp. 272–285.

[67] D. Raca, A.H. Zahran, C.J. Sreenan, R.K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, M. Varvello, Empowering video players in cellular: Throughput prediction from radio network measurements, in: Proceedings of the 10th ACM Multimedia Systems Conference, 2019, pp. 201–212.

[68] D. Raca, A.H. Zahran, C.J. Sreenan, R.K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges, IEEE Commun. Mag. 58 (3) (2020) 11–17.

[69] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, B. Scholkopf, Support vector machines, IEEE Intell. Syst. Appl. 13 (4) (1998) 18–28.

[70] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, J. Lyu, Real-time bandwidth prediction and rate adaptation for video calls over cellular networks, in: Proceedings of the 7th International Conference on Multimedia Systems, 2016, pp. 1–11.

[71] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, W. Wei, LinkForecast: Cellular link bandwidth prediction in LTE networks, IEEE Trans. Mob. Comput. 17 (7) (2017) 1582–1594.

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

[73] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li, J. Li, Realtime mobile bandwidth prediction using LSTM neural network and Bayesian fusion, Comput. Netw. 182 (2020) 107515.

[74] A. Lekharu, K. Moulii, A. Sur, A. Sarkar, Deep learning based prediction model for adaptive video streaming, in: 2020 International Conference on COMmunication Systems & NETworkS, COMSNETS, IEEE, 2020, pp. 152–159.

[75] A. Bentaleb, M.N. Akcay, M. Lim, A.C. Begen, R. Zimmermann, Bob: Bandwidth prediction for real-time communications using heuristic and reinforcement learning, IEEE Trans. Multimed. (2022).

[76] G. Lv, Q. Wu, W. Wang, Z. Li, G. Xie, Lumos: Towards better video streaming QOE through accurate throughput prediction, in: IEEE INFOCOM 2022-IEEE Conference on Computer Communications, IEEE, 2022, pp. 650–659.

[77] B. Wei, H. Song, S. Wang, K. Kanai, J. Katto, Evaluation of throughput prediction for adaptive bitrate control using trace-based emulation, IEEE Access 7 (2019) 51346–51356.

[78] J. Yin, Y. Xu, H. Chen, Y. Zhang, S. Appleby, Z. Ma, ANT: Learning accurate network throughput for better adaptive video streaming, 2021, arXiv preprint arXiv:2104.12507.

[79] A. Sobhani, A. Yassine, S. Shirmohammadi, A video bitrate adaptation and prediction mechanism for HTTP adaptive streaming, ACM Trans. Multimed. Comput. Commun. Appl. (TOMM) 13 (2) (2017) 1–25.

[80] P.J. Kaufman, Smarter trading, vol. 22, McGraw-Hill, New York, 1995.

[81] A. Montieri, G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, A. Pescapè, Packet-level prediction of mobile-app traffic using multitask deep learning, Comput. Netw. 200 (2021) 108529.

[82] R. Babaria, S.C. Madanapalli, H. Kumar, V. Sivaraman, FlowFormers: Transformer-based models for real-time network flow classification, in: 2021 17th International Conference on Mobility, Sensing and Networking, MSN, IEEE, 2021, pp. 231–238.

[83] A. Dietmüller, S. Ray, R. Jacob, L. Vanbever, A new hope for network model generalization, in: Proceedings of the 21st ACM Workshop on Hot Topics in Networks, 2022, pp. 152–159.

Tailai Song: He obtained the B.Sc. in Automotive Engineering at Politecnico di Torino in 2020 and the M.Sc. in ICT for Smart Societies at Politecnico di Torino in 2022. Currently, he is a Ph.D. student in the Telecommunication Networks Group (TNG) from Department of Electronics and Telecommunications (DET) at Politecnico di Torino (PoliTO), Italy, and also a member of the SmartData@Polito research center. His research focuses on machine learning techniques applied to real-time communications to improve Quality of Experience (QoE) and the objective of full-stack observability through end-to-end telemetry.

Paolo Garza: He received the master's and Ph.D. degrees in computer engineering from the Politecnico di Torino. He has been an associate professor at the Dipartimento di Automatica e Informatica, Politecnico di Torino, since December 2018. He spent three years as an assistant professor at Politecnico di Milano. He coauthored about 100 papers in the areas of data mining and machine learning. His current research interests are in the fields of data mining, database systems, and big data analytics. He has worked on classification, clustering, itemset mining and scalable algorithms.

Michela Meo: She is a Professor of Telecommunication Engineering with the Politecnico di Torino. She coauthored about 200 papers, 80 of which on international journals. She edited a book Green Communications (Wiley) and several special issues of international journals. Her research interests include green networking, energy-efficient mobile networks and data centers, Internet traffic classification, and characterization. Prof. Meo was an Associate Editor of ACM/IEEE Transactions of Networking, Green Series of the IEEE Journal on Selected Areas of Communications Networking and IEEE Communication Surveys and Tutorials. She is a Senior Editor of IEEE Transactions on Green Communications. In the role of a General or Technical Chair, she has led the organization of several conferences, including ITC, ICC symposia, ISCC. She chairs the International Advisory Council of the International Teletraffic Congress. She was the Deputy Rector of Politecnico di Torino from March 2017 to March 2018.

Maurizio Matteo Munafò: He is Assistant Professor at the Department of Electronics and Telecommunications of Politecnico di Torino. He holds a Dr.Ing. degree in Electronic Engineering since 1991 and a Ph.D. in Telecommunications Engineering since 1994, both from Politecnico di Torino. He has co-authored about 80 journal and conference papers in the area of communication networks and systems. His current research interests are in simulation and performance analysis of communication systems and traffic modeling, measurement, and classification.