POLITECNICO DI TORINO
Repository ISTITUZIONALE

One Is Enough: Efficient Modeling of RTP Traffic for QoS Predictions in Real-Time Communications

Original
One Is Enough: Efficient Modeling of RTP Traffic for QoS Predictions in Real-Time Communications / Song, T., Garza, P., Meo, M., Matteo Munafo, M.. - In: IEEE TRANSACTIONS ON NETWORKING. - ISSN 2998-4157. - ELETTRONICO. - 34:(2026), pp. 2589-2604. [10.1109/ton.2025.3649980]

Availability:
This version is available at: 11583/3007446 since: 2026-02-09T12:31:43Z

Publisher:
IEEE

Published
DOI:10.1109/ton.2025.3649980

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026

One Is Enough: Efficient Modeling of RTP Traffic for QoS Predictions in Real-Time Communications

Tailai Song, *Member, IEEE*, Paolo Garza, *Member, IEEE*, Michela Meo, *Senior Member, IEEE*, and Maurizio Matteo Munafò

***Abstract*—In recent years, we have witnessed an unprecedented upsurge in popularity and advancement of Real-time Transport Protocol (RTP)-based real-time communication (RTC) applications. For the sake of their optimizations, Quality of Service (QoS) prediction serves as a viable venue for enhancing network monitoring and enabling preemptive solutions. However, existing methodologies are typically tailored and constrained to individual traffic flows and QoS metrics, lagging in correlation capturing and computational efficiency. In light of this, we argue that "*o*ne model is enoug*h*" to conquer these challenges, and propose a novel deep learning (DL) framework namely *Oh*, employing a teacher-student scheme with two training stages. The first (teacher) involves a sophisticated Long Short-Term Memory (LSTM) neural network (NN) empowered by a customized attention structure, and the second (student) comprises simple feedforward NNs to distill knowledge and reduce complexity. Specifically, *Oh* leverages a multi-task learning paradigm, mapping extracted features to four key QoS indicators. It is capable of simultaneously handling unlimited amount of concurrent RTP flows with packet-level information and performing end-to-end predictions of multiple QoS metrics in one single shot. Our work is based on massive traffic collected during real video-teleconferencing calls using various software, and benchmarked against multiple other machine learning (ML)/DL algorithms. As a result, *Oh*-teacher yields superior prediction performance, whereas *Oh*-student achieves distinctly enhanced temporal efficiency with comparable forecasting outcomes.**

***Index Terms*—Real-time communications, real-time transport protocol, QoS, deep learning, machine learning.**

I. Introduction

REAL-TIME communications (RTC) based on Real-time Transport Protocol (RTP) [1] have undergone unparalleled development, becoming indispensable tools for services such as video-teleconferencing, online gaming, live streaming, and more, across professional, recreational, and educational domains. The proliferation of RTC applications can be attributed to the widespread adoption of remote work, together with the boosted needs for enhanced lifestyles in the post-pandemic era. Today, consumer expectations and preferences diversify progressively, pursuing a satisfying overall experience characterized not only by high audiovisual quality but also by seamless, uninterrupted communication. Hence, ensuring the delivery of an enhanced user experience across multifarious contexts entails further comprehensive and effective technologies in RTC systems that go beyond conventional approaches.

In this context, there is a persuasive necessity to cultivate intelligent, robust, efficacious technologies, and to refine present solutions to further improve Quality of Experience (QoE) and network performance. Notably, the Quality of Service (QoS) metrics, such as bitrate, serve as key performance indicators, where their monitoring holds critical values [2], [3], [4], and their prediction plays an instrumental role in strengthening network observability and fostering proactive strategies [5], [6]. Predicting QoS metrics can significantly empower traffic and application management by enabling real-time network optimizations such as adaptive streaming, congestion control, resource allocation, and beyond. These predictions proffer the possibility of anticipating quality degradation and network oscillations to allow for preemptive corrective actions and mitigate adverse effects of fluctuating network conditions, thus ensuring consistent user satisfaction and reducing latency and packet loss. Despite the substantial attention QoS prediction has garnered in research, most existing solutions are limited only to a single QoS metric or an individual RTP flow at a time. In order to gain thorough insights into network performance through multiple metrics of all flows, it necessitates either sequential inference for each individual target (one metric/flow) or parallel computing via multithreading. Nonetheless, both modes are inefficient and resource-consuming, especially in the context of RTC where temporal and computational limitations are prevalent, not to mention that many scenarios require complicated data processing and feature engineering, which further aggravates the challenge. More importantly, the restriction to a single target prevents these methodologies from exploiting the interdependencies among different metrics and flows to optimize predictions.

In this paper, we introduce an innovative end-to-end deep learning (DL) framework named *Oh*, signifying that *o*ne model is enoug*h* to predict multiple QoS metrics for all RTP flows

Received 4 November 2024; revised 20 June 2025; accepted 27 December 2025; approved by IEEE TRANSACTIONS ON NETWORKING Editor L. Huang. Date of publication 31 December 2025; date of current version 12 January 2026. This work was supported in part by Cisco Systems Inc.; in part by the European Union under Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, Partnership on "Telecommunications of the Future" (Program "RESTART", Focused Project R4R), under Grant PE00000001; and in part by the SmartData@PoliTO Center on Big Data and Data Science. *(Corresponding author: Tailai Song.)*

Tailai Song and Michela Meo are with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Turin, Italy (e-mail: tailai.song@polito.it; michela.meo@polito.it).

Paolo Garza is with the Department of Control and Computer Engineering (DAUIN), Politecnico di Torino, 10129 Turin, Italy (e-mail: paolo.garza@polito.it).

Maurizio Matteo Munafò, deceased, was with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, 10129 Turin, Italy.

Digital Object Identifier 10.1109/TON.2025.3649980

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

in a single shot, with minimal time consumption and without relying on extra computational threads. *Oh* operates exclusively on packet-level information devoid of costly feature construction, and employs a knowledge distillation approach. It is composed of a feature extraction component and a multi-task learning block, in which the former constitutes of a Long Short-Term Memory (LSTM) neural network (NN) augmented by customized attention mechanism for the teacher model and basic feedforward neural networks (FNNs) for the student model, and the latter consists of parallel task-specific FNNs that map the extracted packet features to various predicted QoS metrics. Our work is grounded in abundant traffic collected during numerous real video-conferencing sessions under diverse conditions. We formulate a time-series problem, targeting four key QoS metrics regarding bitrate, jitter, packet loss, and frame per second (FPS). We create two distinct datasets and compare with multiple machine learning (ML) and DL algorithms. As a result, *Oh*-teacher yields premier outcomes, demonstrating an optimum candidate for knowledge distillation, and *Oh*-student can achieve comparable performance in negligible time regardless of the computational environment.

The remainder of the paper is organized as follows: Section II provides necessary context regarding problem backgrounds, and our motivations and contributions. Section III formulates the problem and introduces the datasets. Section IV delineates our proposed solution. Section V first introduces the experimental setup, and then exhibits the outcomes, including quantitative results, overhead evaluation, ablation study, and sensitivity analysis. Section VI discusses specific challenges, and Section VII summarizes related works. Finally, Section VIII concludes the work and explores future directions.

II. Problem Context

In this section, we present the necessary background, motivations, and overall contributions of our work.

A. Background

RTC applications can generally be categorized into two types: HTTP (Hypertext Transfer Protocol)-based and RTP-based, in which the former paradigm, exemplified by commercial video-streaming service providers like Netflix, is well-suited for delay-tolerant scenarios thanks to its stateless and reliable nature. Our work focuses on RTP over User Datagram Protocol (UDP), which fundamentally underpins a wide range of RTC applications, particularly for services like video-teleconferencing and online gaming, where minimal latency and near-instantaneous responsiveness are paramount. In the case of web browsers and numerous mobile devices (Android applications), an open-source framework built atop RTP, namely WebRTC [7], remains as the globally recognized standard. RTP traffic consists of multiple individual RTP flows, each encapsulating a specific media type (e.g., audio), and determined by a tuple composed of (IP_{source}, $IP_{destination}$, $Port_{source}$, $Port_{destination}$, $SSRC$, $Type_{payload}$).

B. Motives

Why do we need to predict QoS? In addition to the merits discussed in the literature, QoS metrics are essential objective measurements used to quantitatively evaluate the network/application performance and reliability in RTC. They are indispensable components of utmost significance for stakeholders including end-users and network operators, enabling full-stack observability to monitor streaming quality, and providing critical feedback for decision-making strategies. Traditionally, the Real-time Transport Control Protocol (RTCP) packets are transmitted periodically to provide quality reporting and controlling mechanism for an RTP session. Modern RTC applications offer runtime QoS monitoring (e.g., Zoom statistics dashboard) and post-session logging (e.g., WebRTC logging). However, all of them are historical observations with relatively outdated information, thereby hindering the implementation of preemptive techniques. Therefore, there is a growing interest in forecasting QoS, which could facilitate various network functionalities: improving dynamic adaptive streaming or encoding [8], [9], providing beforehand valuable feedback for congestion control [10], enabling anticipated QoE mapping [11], [12], contributing to software-defined networking (SDN) as key indicators for network management, such as bandwidth allocation [13], and beyond.

Why do we target multiple per-flow QoS metrics? Throughout an RTC session, multiple RTP flows coexist concurrently, each originating from an individual sender, encapsulating a unique media type, and traversing diverse network paths. First, QoS metrics are generally derived separately for each flow, e.g., packet loss is assessed based on sequence inconsistency, as the sequence number increases monotonically for an individual flow, and the bitrate of a single video stream is determined by the encoding conducted on each distinct source. Second, per-flow QoS evaluation could endow us with nuanced insights into the network status and performance from varied granular standpoints, enabling network operations to intervene locally or globally as needed. Furthermore, multiple metrics, rather than a single one, could operate either separately or in combination, contributing to numerous application and network optimization functions and unlocking more possibilities. Additionally, we envision that our utilization of the multi-task learning paradigm could unearth the underlying interdependencies among different metrics, e.g., the occurrence of packet loss might denote bitrate decline, to further enhance the prediction performance. The synchronized and coordinated optimization of all loss functions during model training could perceive and incorporate the interactions between different tasks, enabling predictions to surpass those of single-task learning, where the influence of other factors is absent. More significantly, notwithstanding the disparities among flows and the predominant interactions entwining packets within an individual flow, different RTP flows may still share a portion of common sources, centralized servers, bandwidth bottlenecks, and network conditions, manifesting interconnected relationships, which could be factored in during the processing of coexisting flows to boost prediction performance.

Why do we need efficiency? Following the motivation mentioned above, it is unfortunate that most existing works are task-specific, handling only an individual metric or flow at a time. In order to predict multiple QoS metrics for multiple concurrent flows, one is compelled to perform sequential or parallel computing, which, though feasible, is inefficient. Today's RTC applications often confront temporal and

computational constraints, given their real-time nature and restricted control over devices: (1) the execution of predictions must be swift so that there is enough time left to enact responsive operations, otherwise the QoS can be already observed, and (2) the number of available threads can be determined by a variety of factors such as the device capacity and competing programs, and thus the prediction can only leverage limited computational resource, whereas parallel flows may be dozens. Therefore, we aim to devise a model capable of efficiently predicting unlimited flows and metrics in a single shot without consuming excessive time and extra computational threads, which is also necessary for the all-flow processing to discern flow-wise correlations. We strive to achieve elevated efficiency in terms of time consumption without compromising prediction accuracy. Thus, we resort to knowledge distillation [14] to transfer knowledge from the best-performing, complex model to a simpler yet more efficient one.

Why do we choose packet-level information and attention-based LSTM? As the most fundamental network element with the finest granularity, packets encapsulate the rapidly-changing dynamics and intrinsic characteristics of network traffic [15], which can facilitate models built based on them to effectively apprehend nuanced traffic patterns, and thus gain an enhanced performance [16]. Moreover, extracting packet-level data requires minimal effort, circumventing extra overhead incurred by costly feature extraction and construction. Additionally, our model exclusively relies on RTP header attributes that are readily accessible, bypassing potential complexities associated with packet encryption. In short, we perform streamlined end-to-end predictions directly from raw packets without any intermediate procedures. Nonetheless, to reach our goals of modelling packet-level flow-based traffic while comprehending the interconnections among packets as well as predicted QoS metrics, a simple model might not be competent. The sequential nature of packet flows (analogous to sentences) and the various attributes within each packet (analogous to tokens) parallels problems found in Natural Language Processing (NLP), a domain revolutionized by the Transformer architecture [17]. However, it has been argued that Transformer may not be ideal for time-series problems [18], and therefore, we harness the core component instead of the intact Transformer, i.e., the multi-head self attention mechanism, to intuitively grasp the inherent and intricate correlations interweaving packet-level features, RTP flows, and QoS metrics. Meanwhile, the attention-based LSTM NN, that has been proved to be effective [19], [20], emerges as a promising candidate for our purpose to further improve the performance.

C. Our Contribution

The principal contribution of our work embodies a simple yet efficacious pure MLP-based NN that distills knowledge from a tailored Transformer encoder, facilitating accurate predictions of multiple QoS metrics for all concurrent RTP flows in a single pass. While conforming to a common paradigm of a feature extraction stage followed by downstream modules, and leveraging established model architectures such as LSTM, the novelty of *Oh* lies in two key aspects: *1)* the renovation of the attention mechanism, which meticulously captures packet- and flow-wise correlations to surpass the vanilla Transformer, and *2)* a refined pipeline incorporating per-flow processing and multi-task learning, which adeptly predicts multiple fine-grained QoS metrics with minimal time to tackle the temporal constraints of RTC. We fill the gap where alternative approaches exhibit excessive complexity and lack the flexibility to handle anything beyond individual flows or metrics, by proficiently extracting shared features directly from raw packets. We transcend ordinary feature representation/multi-objective learning schemes, which, while possibly efficient, often yield suboptimal performance, by incorporating heterogeneous relationships and adopting knowledge distillation. In summary, our work constitutes a pioneering effort, endowing RTC with advanced QoS observation that is comprehensive, multidimensional, real-time, and preemptive, criteria that prevailing solutions fail to satisfy.

III. Problem Statement

This section formulates the problem and introduces the dataset utilized throughout our works.

A. Problem Formulation

Our objective is to forecast multiple near-future QoS metrics for RTP flows based on previous packets. At time t, for RTC traffic consisting of M flows, we predict N QoS metrics per flow ($M \times N$ metrics in total) over the upcoming time window of length Δt, using the preceding L packets for each flow and K packet-level features for each packet. We formulate an irregular multivariate multi-target time-series problem as $\hat{\mathbf{Q}}^t = f(\mathbf{X}^t), \hat{\mathbf{Q}}^t \in \mathbb{R}^{M\times N}, \mathbf{X}^t \in \mathbb{R}^{M\times L\times K}$. Let $\hat{\mathbf{Q}}^t$ denote the matrix of predicted QoS in the window from t to $t+\Delta t$, wherein each element $\hat{\mathbf{Q}}^t_{m,n}$ represents the n^{th} type of QoS metric for the m^{th} flow. The feature matrix $\mathbf{X}^t$ consists of the packets precedent to time t, where each entry, $\mathbf{X}^t_{m,l,k}$, corresponds to the k^{th} packet-level attribute of the l^{th} preceding packet in the m^{th} flow. Our goal is to develop a model that learns a function $f(\cdot)$, mapping input features to the estimated QoS metrics that converges closely to actual values, $\mathbf{Q}^t$. In our work, we target $N = 4$ QoS metrics: **Bitrate** ($\hat{\mathbf{Q}}^t_{m,1} = R_{bps}$)—the amount of traffic, i.e., the total size of all packets, in a given flow; **Average jitter** ($\hat{\mathbf{Q}}^t_{m,2} = \overline{R}_{jitter}$)—the mean value of all inter-arrival jitters, each computed according to the protocol; **FPS** ($\hat{\mathbf{Q}}^t_{m,3} = R_{fps}$)—the number of video frames per second for video flows, determined by counting the unique RTP timestamps in the time window; **Loss condition** ($\hat{\mathbf{Q}}^t_{m,4} = y_{loss}$)—the presence of packet loss in the time window, i.e., a binary class label (0:lossless, 1:lossy), determined based on the consistency in the sequence numbers. In summary, we formulate a combination of 3 regression problems and 1 binary classification problem. We follow a moving-window schema, forwarding Δt each time post a prediction, and assimilating new packets with old ones discarded.

We initially consider a duration of $\Delta t = 500$ ms for the predicted time window, and define active flows for each window based on two criteria: an RTP flow is deemed active if (1) it has at least $L = 128$ preceding packets and (2) the latest packet in the past is no more than 1 second prior to

TABLE I
SUMMARY OF DATASETS

	Dataset 1	Dataset 2
Total number of *pcap* files	72	35
Total duration of traffic [h]	69.9	38.9
Period of collection	Apr, 2020 - Jan, 2021	Mar, 2020 - Feb, 2023
Number of time windows	485,271	238,082
Number of data samples	1,856,492	363,989
Application involved	Jitsi Meet Webex	Microsoft Teams Zoom Google Meet Skype BigBlueButton

the start of the time window (t). In other words, inactive or interrupted traffic flows are ignored. Notice that both Δt and L are modifiable parameters, and we elaborate on alternative scenarios in Section V-E. Furthermore, we select $K = 7$ packet-level attributes as features: $\boldsymbol{x_1}$**, absolute inter-arrival time (IAT)**: the interval between the arrivals of two successive packets; $\boldsymbol{x_2}$**, relative IAT**: the time elapsed from the first packet in the L-packet series to a given packet; $\boldsymbol{x_3}$**, lead time**: the duration between a packet's arrival and the start of the predicted window, time t; $\boldsymbol{x_4}$**, packet length**: the total size of a packet, including both header and payload; $\boldsymbol{x_5}$**, RTP timestamp**: a 32-bit value in the RTP header that provides several specific functions such as aiding to maintain the correct timing order of incoming packets; $\boldsymbol{x_6}$**, RTP marker**: a 1-bit flag in the RTP header indicating significant events, such as frame boundaries; $\boldsymbol{x_7}$**, sequence difference**: a binary label that is 0 if the sequence number pattern between two consecutive packets is regular, and 1 otherwise. The first 3 features capture both local and global timing behaviors, and the fourth feature directly reflects the volume of transmitted bits in the past. The remaining features represent RTP-event factors, bringing in potential influence arising from specific network events like prior packet loss. Collectively, these features incorporate manifold facets from temporal, spatial, and RTP-related components.

B. Dataset

Traffic Introduction: Our traffic was collected from numerous real-world video-teleconferencing calls conducted over two different periods using various RTC applications for the purposes of [21] and [22], as summarized in Table I. Each session involves 2 to 6 participants geographically located across Europe, connected via either WiFi, mobile networks (4G/5G), or Ethernet cable. Traffic was captured on the client side and stored in *pcap* format. Notably, we focus exclusively on incoming streams, aiming to predict QoS metrics of RTP flows sourcing from each sender, traversing through the network, and affected by all intermediate links and nodes.

Dataset Construction: We construct the dataset by creating multivariate time series of continuous time windows for each session. For each window, we identify active flows, and then for each flow, we extract preceding packets as features, and calculate corresponding QoS metrics in the window: aggregating all packet sizes to derive bitrate, averaging jitters of all packets, counting unique RTP timestamps, and detecting sequence number discontinuities. Consequently, each time window produces one or more samples (M active flows), with each sample comprising the $N = 4$ QoS metrics as prediction targets and its preceding $L = 128$ packets with their aforementioned $K = 7$ elements as features (a total of $L \cdot M = 128 \cdot M$ packets). Moreover, in order to comprehensively evaluate the performance, our problem can also be framed as a traditional regular time-series forecasting task, leveraging historical observations as features, e.g., QoS metrics from previous time windows. In our case, 128 packets merely translate to 2.97 ± 0.06 seconds, i.e., roughly 6 windows in the past, which is insufficient for typical time-series short-term forecasting problem. Thus, we extend our consideration to 10 s (20 500-ms windows) in the past, curating additional equivalent datasets with prior QoS values as features for the 3 regression problems (e.g., a univariate time-series problem where previous 20 bitrate values are used to predict the next bitrate). Regarding the classification task of packet loss, we refer to [23], creating datasets with traffic statistical features calculated based on aggregated packets within each window, and formulating a multivariate time-series problem. To sum up, for the traffic of each video-call session, we generate two similar datasets with identical prediction targets but different features of either preceding packets or historical information. With the same procedure, we construct two distinct compilation of datasets based on different applications and collection periods, aiming to evaluate the model's generalizability and versatility across different software and times.

Dataset Characteristics: To provide context and substantiate the reliability of the data in our study, we briefly present the traffic patterns, as indicated by the seven empirical cumulative distribution function (ECDF) plots in Figure 1.

Figure 1a illustrates the number of concurrent RTP flows in each window. Apparently, dataset 2 exhibits a significantly lower number of parallel flows, with nearly half of the windows containing only a single flow. This could be attributed to the fact that the traffic was collected for the sake of a basic analysis of video conferencing applications (VCAs) [22], and thus the focus was on gathering data from various apps with minimal participant involvement, resulting in fewer streams generated. In contrast, dataset 1, which comprises traffic generated during actual video-calls and closely reflects real-world scenarios, features plenty of coexisting RTP flows up to 11. On average, there are roughly four concurrent flows per time window, with only around 10% of the windows engaging 1 single flow, further underscoring our motivation.

For predicted targets, Figure 1b, 1c, and 1d outline three QoS metrics, with several individual flows displayed. Both datasets share similar trends across various metrics, with the majority (around 75%) of bitrates and jitters remaining below 0.1 Mbps and 10 ms, respectively. Nonetheless, with a closer peek at the randomly selected flows, diverse patterns emerge, differing from one another substantially. For FPS, we observe drastic ascents at 30 fps, which aligns exactly to the standard settings of common video encoders in VCA. However, a range of FPS values persists, justifying the presence of different video types, e.g., low/high quality and screen sharing. Regarding the packet loss, dataset 1 contains 28,614 lossy samples,

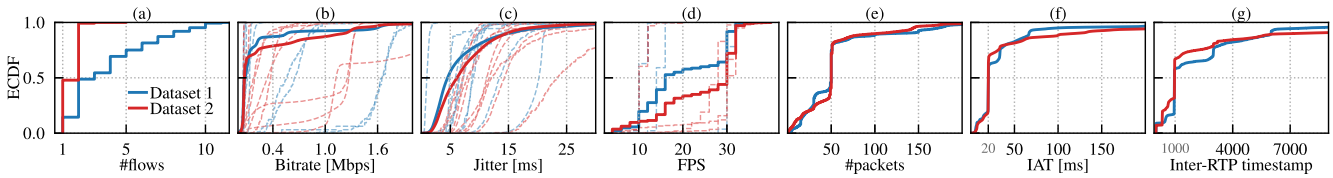


Fig. 1. Traffic patterns (light-colored, dashed lines in b, c, d correspond to 10 randomly selected individual flows).

while dataset 2 has 14,133, accounting for merely 1.5% and 3.9% of their respective datasets, which highlights a class imbalance classification problem.

Additionally, Figure 1e, 1f, and 1g provide insights into the actual transmitted multimedia data by analyzing 3 types of per-second packet statistics: total packet count, average inter-arrival time, and average inter-difference between RTP timestamps in each flow. Generally, they all show a notable degree of statistical similarity, manifesting multiple marked spikes in ECDFs, which can be ascribed to the presence of various payload types, e.g., audio flows. Hence, it is reasonable to infer that our dataset is composed of rich and diverse media content rather than monotonous streams. All of the aforementioned observations illustrate the diversity and comprehensiveness of our traffic, affirming the datasets' representativeness and relevance in capturing the multifaceted nature of RTC traffic. Although such mixture of different patterns may perplex somewhat the problem, it simultaneously upholds the generalizability and versatility of our model.

IV. Methodology

In this section, we begin with a brief introduction to relevant DL backgrounds, and then delineate our proposed solution.

A. Background

Our framework is rooted in knowledge distillation, a technique frequently adopted in ML/DL domains to transfer knowledge from a computationally intensive model with high capacity to a smaller yet efficient one. Typically, knowledge distillation involves two models: the teacher, a capable pre-trained large model, and the student, a comparatively smaller model that aims to learn from the teacher by incorporating auxiliary losses between itself and the teacher model. Although the teacher model excels in performance, its complexity hinders efficiency, making it unsuitable for hardware- and time-sensitive scenarios such as RTC in our case. Conversely, the standalone student model lacks the capacity to learn a concise knowledge representation and thus needs guidance from the teacher to steer itself, emulating expert behavior.

In our work, we employ an attention-based LSTM NN in the teacher model. The attention mechanism serves a pivotal role in sequence modelling, with the most influential and renowned contribution being the introduction of the Transformer with the multi-head self-attention [17]. It involves a series of computation of the query (Q), key (K), value (V) matrices derived from the input sequence:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \quad (1)$$

However, Transformer presents 2 inherent issues for our specific problem. First, the vanilla version of Transformer supports only a fixed input length, whereas we deal with an unfixed number of RTP flows, necessitating architectural customization. It is worth noting that while certain variants of Transformer can indeed handle unlimited input [24], they are not implementable to our case. Second, despite the proliferation of innovative models, the applicability of Transformer to time-series problems remains contentious [18], [25]. To this end, rather than simply using Transformer, we opt for a hybrid approach, integrating multi-head self-attention with LSTM, thus combing the strengths of both architectures. LSTM NN [26] is a special type of recurrent neural network (RNN) that conquers the problem of vanishing gradient, and demonstrates efficacy in time-series tasks [27], [28]. Although LSTM has lost its competitiveness against other more sophisticated ensemble models, it is still an auspicious option as an individual component for sequence modelling [29], [30]. It is characterized by one or more LSTM layers, each comprising a three-gate cell that sequentially processes the input and updates its parameters accordingly. Notably, unlike most approaches that leverage attention mechanism to assign weights to the output of LSTM, we first apply attention to refine packet sequence and then inject them into LSTM layers.

B. Proposed Framework

Overview: We propose *Oh*, a DL framework as depicted in Figure 2, which adheres to a teacher-student scheme. The teacher is a large model capable of effectively apprehending knowledge representation, while the student is a simpler model tasked with supplementary objectives to distill knowledge from the teacher. Generally, at a given time instant, we extract and combine preceding valid traffic flows, passing packet-level attributes to the packet enrichment process. The output of this process is subjected to a feature extraction module responsible for sequence (RTP flow) modelling and then a multi-task learning block that maps extracted features to various QoS metrics.

Packet Enrichment: For a given targeted time window, we arrange active RTP flows with their respective preceding packets in chronological order, as illustrated in Section III-B. All valid flows (the 128 packets with their 7 packet-level features of each active flow) are concatenated, forming a $128 \cdot M \times 7$ matrix as the raw input for the packet enrichment process. Next, we incorporate a linear layer namely the packet embedding layer to transform the input into embedded features, i.e., a $128{\cdot}M \times N_{embedding}$ matrix ($7 \rightarrow N_{embedding}$), mapping packet attributes to their latent embeddings and thereby enriching the traffic features. Although each packet has temporal components to indicate its order, we follow the good practice of the original Transformer model, stack-

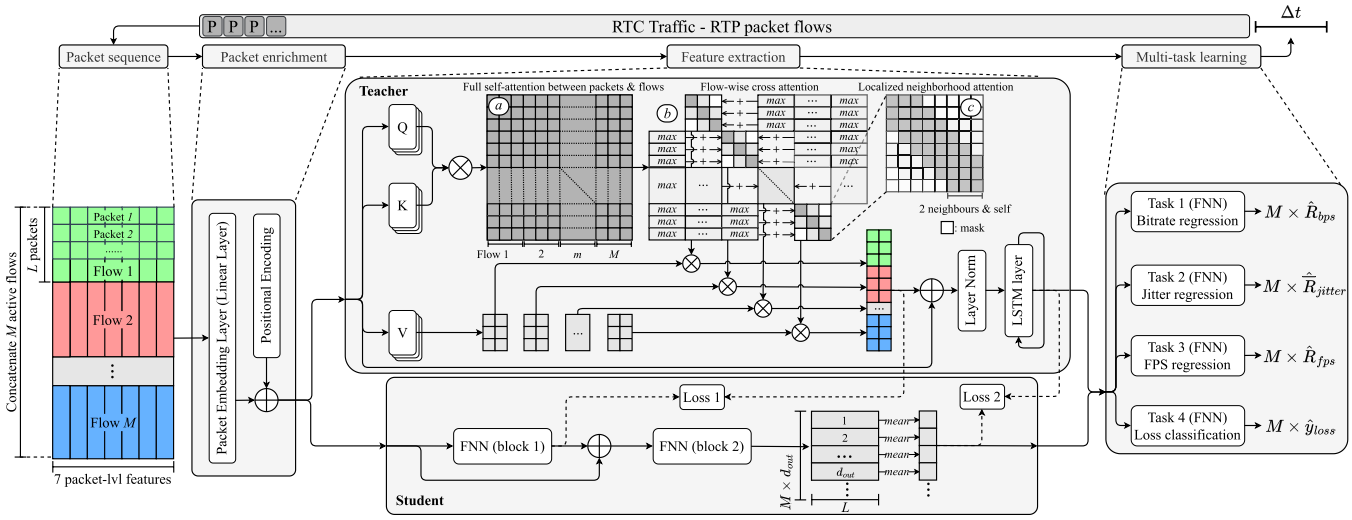


Fig. 2. *Oh*: model architecture.

ing *sine/cosine* positional encoding on top of the embedded features, and thus infusing positional information into other packet entities. Consequently, the traffic sequence of embedded packet features is then fed into the subsequent feature extraction module for either model.

Teacher Model—Preliminary: The first part mirrors the original Transformer, where each sample (enriched packet embedding) in the sequence is fed into three independent linear layers to generate the corresponding query, key and value vectors. We then devise and customize a dual attention architecture called the cross and neighborhood multi-head self-attention, aiming to eliminate the restriction on input quantity (variable M) and correlate concurrent RTP flows. The initial step in Equation 1 forms the attention weight matrix by multiplying the query and key matrices ($W = QK^T$). As exemplified by the top-left large grayish matrix (marked by a in Fig. 2), the full (regular) self-attention yields a square matrix (attention weights) with dimension $3M \times 3M$, assuming each of the M flows contains $K = 3$ packets. Specifically, $W_{3M\times 3M}(1{:}3, 1{:}3)$ represents the attention weights acknowledged within flow 1, e.g., $W_{3M\times 3M}(1,1)$ and $W_{3M\times 3M}(1,2)$ are the attentions paid by the 1^{st} packet in flow 1 to itself and to the 2^{nd} packet also in flow 1, respectively.[1] In contrast, $W_{3M\times 3M}(1, 4{:}6)$ indicates the attentions paid by the 1^{st} packet in flow 1 to packets from flow 2, and similarly, $W_{3M\times 3M}(1, 3m-2{:}3m)$ shows the attentions paid by the same packet to m^{th}-flow packets. Accordingly, entries in $W_{3M\times 3M}(1{:}3, 4{:}3M)$ reflect the attention weights assigned by flow 1 to other flows.

Teacher Model—Flow-Wise Cross Attention: As portrayed by the manipulation of a collection of matrices (marked by b in Fig. 2), the 3-by-3 sub-matrices along the diagonal are computed multiplying the query matrix of a certain flow by the key matrix of the same flow, i.e., $Q(3m-2{:}3m,:)\cdot K(3m-2{:}3m,:)^T$, and the sub-matrices off the diagonal is derived based on different flows, i.e., $Q(3m-2{:}3m,:)\cdot K(3n-2{:}3n,:)^T, m \neq n$. In this context, the diagonal sub-matrices stand for intra-correlations within individual flows, while other off-diagonal square matrices with the same size represent inter-correlations between different flows. As a result, it is viable to scale up to a theoretically infinite amount of concurrent RTP flows, thereby overcoming the obstacles associated to Transformer's input limitations by treating packets from all flows as a collective input while processing them separately for attention weight computations. Furthermore, the flow-wise cross attention also facilitates the discovery of potential relationships among concomitant flows, in addition to the primary intra-correlations interlacing packets within the same flow. This is achieved by augmenting the focused diagonal matrices with refined information from other flows (off-diagonal square matrices). In particular, the attention weights recognized between different flows (flow-wise inter-correlation, e.g., m^{th} vs. n^{th}) are represented by an off-diagonal sub-matrix $W_{3M\times 3M}(3m-2{:}3m, 3n-2{:}3n), m \neq n$, wherein each row denotes the attentions paid by a certain packet in the current flow (m) to all packets in another flow (n). In order to incorporate the inter-correlations from other flows, we first apply max-pooling to each row of these off-diagonal sub-matrices, extracting the most influential factors. We then aggregate the resultant max values from all other flows in a row-wise manner and add them to each attention weight on the corresponding row within the current flow (diagonal sub-matrix), e.g., $W_{3M\times 3M}(1,1{:}3) \leftarrow W_{3M\times 3M}(1,1{:}3) + \sum_{m=2}^{M} \max(W_{3M\times 3M}(1, 3m-2{:}3m))$. Afterwards, we retain only the diagonal sub-matrices, abandoning the off-diagonal ones entirely. In essence, the flow-wise cross-attention prioritizes and confines packet interactions to those localized within the same flow, while still accounting for a certain degree of impact from other flows.

Teacher Model—Localized Neighborhood Attention: We further modify each sub-matrix of attention weights on the diagonal, inspired by [31]. In our work, we initially employ $L = 128$ packets per flow, which may span over several seconds. We contend that the correlations are substantial between adjacent packets, while minor between temporally distant ones. This is especially relevant in RTC, where network dynamics fluctuate swiftly, causing characteristics embodied

[1]For a matrix W, $W(i,j)$ denotes the element in row i and column j, and $W(n{:}m, q{:}p)$ represent all entries from the n^{th} to the m^{th} row as well as from the q^{th} to the p^{th} column.

by, for instance, early and late packets with a relatively elongated duration in between to differ essentially, thereby manifesting only trivial relationships. To this end, we deliberately localize the attention paid by a packet only to itself and its k-nearest neighbors within a flow, inserting prior knowledge into the model rather than naively letting the NN to arduously learn such a pattern at a risk of failure. The rightmost zoomed-in 8-by-8 matrix with gray and white mixed (marked by c in Fig. 2) depicts an example for a flow (sub-matrix) of 8 packets and a neighborhood degree of $k = 2$. We apply masks to redundant entries (distant attention weights), only retaining the diagonal region, where a minimum of 3 weights (the edge packet itself and the its 2 immediate neighbors) and a maximum of 5 weights (itself and 2 packets on each side) per row are preserved. In other words, the retained diagonal sub-matrix for an L-packet flow is an $L \times L$ square matrix, where, in row l, the $(l-k)^{th}$ to $(l+k)^{th}$ elements are preserved, while others are set to $-\infty$. Consequently, the localized neighborhood attention ensures the focus of the model on closely related packets, alleviating noises produced by irrelevant packets and improving the precision of the attention mechanism.

Teacher Model—Remaining Parts: We finally integrate the 2 aforementioned operations, keeping only diagonal sub-matrices enhanced by instrumental information retrieved from discarded off-diagonal sub-matrices, to finalize the attention weights for individual flows, each then multiplying its corresponding value matrix (V) to obtain the attention score (a $128 \times N_{embedding}$ matrix). We concatenate scores of all M flows to derive the final output, a $128M \times N_{embedding}$ matrix. The complete process is formalized in Equation 2, as shown at the bottom of the page, where d is the head dimension ($\lceil N_{embedding}/num_{head} \rceil$), k is the number of confined neighbor packets (i.e., neighborhood degree), and f_q, f_k, f_v represent three independent linear layers that convert the packet sequence of the m^{th} flow ($\mathcal{P}_m$) into its corresponding query ($\mathcal{Q}_m$), key ($\mathcal{K}_m$), and value ($\mathcal{V}_m$). Subsequently, the result of our customized attention mechanism is superimposed on the original input (the sequence of enriched packet features) through a residual connection. We also apply layer normalization to standardize values dependently for an individual packet to stabilize the data pattern. Consequently, we obtain the final sequence of encoded features, which basically merges the primary packet sequence with contextual information, including the correlations among packets and RTP flows. To further model the integrated sequence, we channel the encoded features into an LSTM layer, and the last hidden states of all flows, arranged in a matrix of shape $M \times d_{out}$ where each d_{out}-long vector corresponds to the extracted features of one of the M flows, are produced and then fed into the following block.

Multi-Task Learning Block: To efficiently predict various QoS metrics, we incorporate a multi-task learning block, sharing outputs of the previous stage instead of developing independent task-specific features for each QoS metric. Specifically, we implement four sub-blocks, each corresponding to one of the target metrics, and comprising an identical FNN architecture. The extracted features from each flow are passed to all sub-blocks, generating the respective predicted QoS metrics, and thus resulting in a total of $4M$ outputs. Moreover, instead of simply summing the four losses during the training phase, we refer to [32], implementing learnable weights that dynamically adjust the relative importance of different tasks and systematically blend losses:

$$\begin{aligned}\mathcal{L}_{\text{teacher}} &= e^{-w_1} \cdot \mathcal{L}_{\text{bps}} + w_1 + e^{-w_2} \cdot \mathcal{L}_{\text{jitter}} + w_2 + \\ &\quad e^{-w_3} \cdot \mathcal{L}_{\text{fps}} + w_3 + e^{-w_4} \cdot \mathcal{L}_{\text{loss}} + w_4 \\ \text{with } \mathcal{L}_{\text{bps}} &= \ell_{\text{MAE}}(R_{bps}, \hat{R}_{bps}), \\ \mathcal{L}_{\text{jitter}} &= \ell_{\text{MAE}}(\overline{R}_{jitter}, \hat{\overline{R}}_{jitter}), \\ \mathcal{L}_{\text{fps}} &= \ell_{\text{MAE}}(R_{fps}, \hat{R}_{fps}), \\ \mathcal{L}_{\text{loss}} &= \ell_{\text{wBCE}}(y_{loss}, \hat{y}_{loss}, w_{\text{class}}),\end{aligned} \tag{3}$$

where $\mathcal{L}_{\text{teacher}}$ is the final loss for the teacher model, computed by combining the individual losses of tasks ($\mathcal{L}_{\text{bps}}$, $\mathcal{L}_{\text{jitter}}$, $\mathcal{L}_{\text{fps}}$, $\mathcal{L}_{\text{loss}}$) through 4 trainable parameters (w_1, w_2, w_3, w_4). For regression tasks, we adopt the Mean Absolute Error (MAE) loss function $\ell_{\text{MAE}}(\cdot)$, while for the classification task, we apply a weighted Binary Cross Entropy (BCE) loss function $\ell_{\text{wBCE}}(\cdot)$, with larger weights w_{class} assigned to the minority class (lossy time windows) to tackle the issue of class imbalance.

Student Model: The feature extraction component in *Oh*-student is essentially distinct with a substantially simplified architecture, comprising solely fully connected layers, while other blocks remain unchanged. Knowledge distillation forms the core of *Oh*-student, and we conform to the feature-based paradigm [33], fetching knowledge from intermediate layers of the teacher model. Specifically, we construct 2 FNN blocks, in which the former one is designed to learn the feature

$$\begin{aligned}&\text{Cross-Neighbourhood-Attention}(Q,K,V) = \text{Softmax}\left(\frac{W}{\sqrt{d}}\right) \circ V \\ &\text{with } W = \begin{bmatrix} \vdots \\ \mathcal{W}_{mask}(128(m-1)+1:128m, 128(m-1)+1:128m) + \\ \left[\ldots \sum_{i=1, i\neq m}^{M} \max(\mathcal{W}(128(m-1)+n, 128(i-1)+1:128i)\ \ldots\right]^T \\ \vdots \end{bmatrix}, \ n \in [1,128], m \in [1,M], \mathcal{W} = QK, \\ &\mathcal{W}_{mask}(i,j) = \text{mask}(\mathcal{W}(i,j)) = -\infty; i \in [1:128], j \in [1:i-k-1] \cup [i+k+1:128], \\ &Q = [\mathcal{Q}_1\ \ldots\ \mathcal{Q}_m\ \ldots]^T, \mathcal{Q}_m = f_q(\mathcal{P}_m), K = [\mathcal{K}_1^T\ \ldots\ \mathcal{K}_m^T\ \ldots], \mathcal{K}_m = f_k(\mathcal{P}_m), V = [\mathcal{V}_1\ \ldots\ \mathcal{V}_m\ \ldots]^T, \mathcal{V}_m = f_v(\mathcal{P}_m).\end{aligned} \tag{2}$$

representation from the attention mechanism, and the latter one aims to imitate the LSTM NN. First, the enriched packet sequence of each flow is fed into block 1, whose output is directly compared with respect to the attention scores produced by the cross and neighborhood attention in the teacher model. As before, a residual connection is applied, adding the block's output back to the original input sequence. Next, the sequence is processed by block 2, generating an output matrix with dimension of $M \cdot d_{out} \times L$, where d_{out} represents both the hidden state size of the LSTM in the teacher model and the dimension of the last layer (number of neurons) in the FNN block, and each sub-matrix of size $d_{out} \times L$ corresponds to one out of the M flows. Afterwards, we average on each L-long vector along the first dimension, thereby condensing each of the d_{out} output projections across all L packets. As a result, we obtain a vector of length $M \cdot d_{out}$ as the final output of sequence modelling, which is entailed to contrast the LSTM output from the teacher model.

Student Model—Loss Function: The multi-task learning block in *Oh*-student remains unaltered, with the exception of introducing two additional losses related to knowledge distillation. Instead of conducting a soft matching of output distributions, we expect the student model to precisely converge to the teacher model, i.e., the outputs of the two FNN feature extraction blocks in the student model is anticipated to closely approximate the corresponding outputs in the teacher model. We aim to minimize the MAE losses between these entities in tandem with the original task-related losses:

$$\begin{aligned} \mathcal{L}_{\text{student}} &= \mathcal{L}_{\text{teacher}} + e^{-w_5} \cdot \mathcal{L}_{\text{b1}} + w_5 + e^{-w_6} \cdot \mathcal{L}_{\text{b2}} + w_6 \\ \text{with } \mathcal{L}_{\text{b1}} &= \ell_{\text{MAE}}(out_{attention}, out_{b1}), \\ \mathcal{L}_{\text{b2}} &= \ell_{\text{MAE}}(out_{LSTM}, out_{b2}), \end{aligned} \tag{4}$$

in which $\mathcal{L}_{\text{student}}$ is the eventual loss of *Oh*-student, derived by combing the loss of *Oh*-teacher ($\mathcal{L}_{\text{teacher}}$ from Equation 3) with the losses from the two FNN blocks ($\mathcal{L}_{\text{b1}}, \mathcal{L}_{\text{b2}}$) through two additional trainable parameters, w_5 and w_6. In other words, we transform the 4-task learning problem into a 6-task one. In consequence, the complicated as well as computationally expensive components of the attention and LSTM are distilled into two lean FNNs, resulting in a pure multi-layer perceptron (MLP)-based student model.

V. Experimental Result

In this section, we first present experimental setup in detail. Subsequently, we discuss the quantitative performance, and showcase the time consumption across various environments. Following this, we perform a series of ablation tests and sensitivity analyses to further validate the superiority of *Oh*.

A. Experimental Setup

Model Training: Oh is developed on a single GPU of NVIDIA Tesla V100-16GB, and Table II enumerates the implementation details. We adopt a three-stage development strategy, training firstly *Oh*-teacher in an usual way, secondly *Oh*-student with the guidance from the teacher model, and thirdly *Oh*-student itself without auxiliary losses. During the second stage, *Oh*-teacher remains frozen, acting as a pre-trained inference-only model to generate intermediate outputs. More critically, we subsequently discard completely the teacher model during the third stage, continuing the training of the student model based solely on the four task-specific losses, with the knowledge already distilled. In this respect, it bears resemblance to a self-learning paradigm, transcending the typical knowledge distillation scenario, where the student model remains tethered to the teacher. Therefore, the parameters of the "pre-trained" student model, now imbued with distilled knowledge from the second stage, serve as an optimal starting point to unleash potentials and further explore advantageous solutions, thanks to the relaxed constraints devoid of the interference from the teacher model.

Model Evaluation: As discussed in Section III-B, we construct 2 dataset packs, called dataset 1 and 2, with the same structure but different traffic sources. For dataset 1, we intentionally and randomly partition the entire corpus based on individual *pcap* files (video-call sessions) instead of following the conventional ML paradigm of shuffling all available samples. In particular, the 72 *pcaps* are divided into 3 independent groups (50, 10 and 12) to form the training (1,406,398 samples), validation (162,472), and test (287,622) sets, respectively. Moreover, the entire dataset 2 is used as another test set, and we compute based on the training set the statistics of mean value and standard deviation for features and targets to standardize other datasets. In consequence, we ensure that *Oh* is trained on traffic manifesting distinctive conditions different from other datasets regarding software,

TABLE II
Implementation Detail of Oh

Parameter	Value
Learning rate*, η	10^{-3}
Size of feature embedding, $N_{\text{embedding}}$	32
Number of heads, num_{head}	8
Dimension of heads, d	4
Neighbourhood degree, k	32
Number of LSTM layer	1
Size of hidden state, d_{out}	32
Number of layers for task-specific FNN	2
Number of neurons for task-specific FNN	32, 16
Activation function	*Leaky ReLU* [34]
Number of layers for FNN block 1	2
Number of neurons for FNN block 1	64, 32
Number of layers for FNN block 2	2
Number of neurons for FNN block 2	64, 32
Dropout ratio	0.2
Training optimizer	*Adam* [35]
Batch size	8

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

TABLE III
Considered Models for Comparison

Category	Model
Time-series	Random Forest (RF) [36]
	eXtreme Gradient Boosting (XGB) [37]
	Multi-layer Perceptron (MLP) [38]
	Long Short-Term Memory (LSTM) [26]
	N-BEATS [39]
Packet-level	LSTM
	Long- and Short-term Time-series network (LSTNet) [40]
	Non-stationary Transformer (NST) [41]
	PatchTST [42]
	TimesNet [43]
	TSMixer [44]

TABLE IV
EXPERIMENTAL RESULTS OF THE TEST SET IN DATASET 1 AND THE ENTIRE DATASET 2

Model	Bitrate [Mbps]				Jitter [ms]				Frame per second [fps]				Packet loss [-]		
	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	Recall-0	Recall-1	F1-score
Dataset 1															
RF	0.097	0.031	13.22%	0.968	**1.535**	**0.764**	14.42%	0.903	1.947	0.876	6.01%	0.960	0.834	**0.645**	0.522
XGB	0.098	0.032	14.65%	0.968	1.603	0.802	16.07%	0.894	2.624	1.065	9.25%	0.927	**0.967**	0.456	0.641
MLP	0.099	0.036	17.84%	0.967	2.071	0.861	14.66%	0.823	2.407	1.198	8.38%	0.939	0.629	0.227	0.395
LSTM	0.101	0.034	13.62%	0.965	2.083	1.109	14.26%	**0.904**	2.352	0.951	6.53%	0.941	0.783	0.177	0.451
N-BEATS	0.102	0.043	16.19%	0.965	2.142	1.117	**14.12%**	0.898	2.493	1.153	7.65%	0.934	0.675	0.284	0.417
LSTM	0.103	0.031	8.58%	0.964	1.911	0.884	16.45%	0.850	2.217	0.974	7.01%	0.945	0.961	0.493	0.637
LSTNet	0.100	0.030	9.94%	0.966	1.975	0.977	17.30%	0.840	1.710	0.948	6.51%	0.968	0.950	0.590	0.637
NST	0.098	0.030	9.24%	0.968	2.427	1.155	20.93%	0.758	2.005	1.144	8.11%	0.955	0.947	0.568	0.626
PatchTST	0.176	0.059	20.76%	0.895	3.467	1.898	42.78%	0.505	3.998	2.614	21.39%	0.823	0.950	0.575	0.634
TimesNet	0.109	0.033	11.77%	0.960	2.327	1.103	20.17%	0.777	1.852	0.890	6.61%	0.962	0.942	0.615	0.626
TSMixer	0.098	**0.028**	8.56%	0.968	2.005	0.932	17.13%	0.834	1.918	1.103	7.35%	0.959	0.954	0.580	0.642
Oh-Teacher	**0.095**	**0.028**	**8.19%**	**0.970**	1.692	0.786	15.30%	0.882	1.621	**0.710**	**4.83%**	0.971	0.954	0.586	0.644
Oh-Student	**0.095**	**0.028**	8.44%	0.969	1.958	0.892	16.26%	0.842	**1.582**	0.725	5.03%	**0.972**	0.955	0.592	**0.647**
Dataset 2															
RF	0.102	0.041	18.13%	0.957	2.055	1.245	17.96%	0.863	2.538	1.467	8.80%	0.924	0.810	0.295	0.490
XGB	0.097	0.037	16.28%	0.962	**2.029**	**1.244**	18.55%	**0.867**	3.097	1.852	18.62%	0.886	0.922	0.102	0.504
MLP	0.088	0.038	18.04%	0.968	2.086	1.252	17.95%	0.859	2.589	1.609	9.75%	0.921	0.587	0.697	0.426
LSTM	0.094	0.033	13.32%	0.964	3.095	1.877	18.11%	0.839	2.469	1.490	9.18%	**0.928**	0.756	0.677	0.515
N-BEATS	0.091	0.044	27.55%	0.966	2.936	1.790	**17.12%**	0.855	2.575	1.513	8.87%	0.922	0.850	0.043	0.459
LSTM	0.119	0.044	10.80%	0.942	2.449	1.589	22.90%	0.805	2.937	1.534	9.01%	0.891	0.952	0.688	**0.724**
LSTNet	0.100	0.035	10.96%	0.959	2.304	1.384	18.48%	0.828	2.479	1.470	**8.03%**	0.922	**0.949**	0.697	0.719
NST	0.111	0.039	11.37%	0.949	3.848	2.341	33.13%	0.520	2.945	1.686	9.56%	0.890	0.928	0.679	0.674
PatchTST	0.272	0.112	35.83%	0.697	4.917	3.039	43.62%	0.216	5.209	3.846	23.99%	0.657	0.946	0.700	0.713
TimesNet	0.491	0.145	159.50%	0.014	3.232	2.027	26.26%	0.661	2.852	1.557	9.98%	0.897	0.942	**0.710**	0.707
TSMixer	0.113	0.037	**10.04%**	0.948	2.687	1.661	22.79%	0.766	2.944	1.878	10.84%	0.891	0.947	0.701	0.717
Oh-Teacher	0.099	0.036	10.05%	0.960	2.182	1.334	18.93%	0.845	2.490	1.389	8.47%	0.922	0.941	0.707	0.705
Oh-Student	**0.087**	**0.032**	10.32%	**0.969**	2.581	1.602	21.89%	0.784	**2.395**	**1.318**	8.26%	**0.928**	0.941	0.696	0.702

[1] RMSE ↓, MAE ↓, MAPE ↓, R^2 ↑, Recall-0 ↑, Recall-1 ↑, F1-score ↑ (↓: the lower the better, ↑: the higher the better).
[2] **Bold** indicates the best value, and underline denotes the second best.

TABLE V
RESULTS OF ABLATION ANALYSES (DATASET 1: UPPER PART, DATASET 2: LOWER PART)

Scenario	Bitrate [Mbps]				Jitter [ms]				Frame per second [fps]				Packet loss [-]		
	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	Recall-0	Recall-1	F1-score
Ablation test 1	0.097	0.030	8.56%	0.968	1.770	0.804	14.15%	0.871	1.662	0.744	5.12%	0.969	0.952	0.579	0.638
Ablation test 2	0.093	0.028	8.18%	0.971	1.853	0.845	15.42%	0.859	1.580	0.733	5.11%	0.972	0.954	0.588	0.645
Ablation test 3	0.101	0.030	8.92%	0.966	1.864	0.887	16.62%	0.857	1.704	0.801	5.40%	0.968	0.954	0.586	0.644
Ablation test 4	0.101	0.032	10.02%	0.965	2.203	1.023	17.04%	0.800	1.802	1.107	8.13%	0.964	0.949	0.601	0.637
Ablation test 5	0.096	0.029	8.96%	0.969	1.710	0.758	13.73%	0.880	1.739	0.726	4.93%	0.966	0.955	0.561	0.639
Ablation test 1	0.116	0.040	10.78%	0.945	2.231	1.342	18.47%	0.838	2.554	1.447	9.07%	0.918	0.937	0.698	0.696
Ablation test 2	0.090	0.032	10.26%	0.967	2.448	1.609	23.15%	0.806	2.360	1.270	7.52%	0.930	0.933	0.699	0.687
Ablation test 3	0.110	0.039	10.76%	0.950	2.613	1.681	23.08%	0.778	2.594	1.432	8.30%	0.915	0.947	0.692	0.714
Ablation test 4	0.096	0.033	10.90%	0.962	2.693	1.635	21.00%	0.765	2.733	1.840	10.89%	0.906	0.944	0.706	0.710
Ablation test 5	0.103	0.035	10.61%	0.956	2.189	1.325	18.19%	0.845	2.740	1.522	9.01%	0.905	0.955	0.215	0.573

location, connectivity, time, etc., to avoid data leakage among traffic, and obtain a more universal and versatile solution.

On top of that, to comprehensively assess the performance, we compare *Oh* against an array of ML/DL algorithms, either appearing in the literature or delivering state-of-the-art performance in time-series problems. As listed in Table III, V models are implemented for time-series features, while 6 are applied to packet-level features. Notably, we also examine multiple other approaches such as DLinear [18], but they do not align well with our specific problem. For each task (QoS metric), an individual model is required, meaning four independent models are built for each comparative algorithm. Quantitatively, we evaluate 4 metrics between the ground truth and prediction for regression problems: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the coefficient of determination (R^2 score). As for the classification one, we only employ class recalls and the F1-score to measure both individual class accuracy and overall performance, as some metrics, such as precision, are highly skewed in the case of class imbalance and thus not meaningful.

B. Quantitative Results

Table IV summarizes the experimental results for all models and both datasets. Overall, *Oh*-teacher demonstrates superior predictive performance on dataset 1, albeit slight overfitting on dataset 2. *Oh*-student achieves comparatively decent outcomes, with even enhanced performance for dataset 2.

For dataset 1, *Oh*-teacher emerges as the top contender, outstripping predominantly other competitive models. It delivers exceptional performance in terms of bitrate and FPS, being, for example, the only model with an R^2 score reaching 0.97 for bitrate and an MAPE lower than 5% for FPS. As for

average jitter, it is important to note that all of the packet-level models cannot surpass time-series ones, owing to the fact that time-series features consist of historical jitters, enabling models to capture recent jitter trends and thus operate in an autoregressive manner. Packet-level features, on the other hand, do not encompass the preceding jitter value for each packet, relying on models to autonomously discern underlying relationships between jitters and considered packet features. We intentionally exclude jitter from packet-level features, aiming to avoid the overhead incurred by jitter computation, as jitter is derived iteratively for each packet. Although this decision may limit jitter prediction, we remain committed to our current approach instead of simply injecting jitter into packet features, since the primary goal is to improve efficiency. Meanwhile, *Oh*-teacher still outperforms other packet-level models, exhibiting comparable performance with respect to time-series models, exemplified by the second-best MAE. Regarding packet loss predictions, *Oh*-teacher may not seem to be the prominent choice with the best class recalls at a first glance. Nonetheless, the degradation concerning class 0 is trivial comparing to the top rank (XGB), which, however, suffers a 13% lower recall for class 1, and RF misclassifies excessive lossless time windows, yielding a subpar recall for class 0, even it excels *Oh*-teacher by roughly 6% in class-1 recall. On the contrary, *Oh*-teacher effectively identifies the majority of lossy time windows without sacrificing the performance of lossless ones, reaching a balance between both classes, and thereby attaining a commendable F1-score. Meanwhile, *Oh*-student inherits advantages of the teacher, managing to achieve comparable performance even with certain slightly better results, e.g., the highest F1-score for packet loss.

When it comes to dataset 2, *Oh*-teacher fails to output the premier performance (though it remains among top ranks), manifesting marginal overfitting, which could stem from the fact that the effectiveness of *Oh*-teacher, trained by capitalizing on relationships among parallel flows, cannot be fully unleashed due to the nature of dataset 2, where most time windows contain no concurrent flows. However, this is considered a minor issue, as the modern RTC landscape typically features multiple coexisting flows to adequately support multidimensional multimedia content, functional traffic, and an unlimited number of participants. Fortunately, *Oh*-student mitigates the overfitting issue, producing outstanding performance, especially for bitrate and FPS. Although we still face the drawbacks on jitter prediction and a slight decline for packet loss, the overall performance remains satisfactory. Interestingly, all time-series models fall short in packet loss prediction regardless of the dataset, e.g., 9 out of 10 F1-scores are below 0.6.

To summarize, *Oh*-teacher stands as the optimal candidate for knowledge distillation, while *Oh*-student, with its simplicity, maintains robust performance without compromise.

C. Temporal Overhead Evaluation

The supremacy of *Oh* is justified by the quantitative outcomes, but our primary objective lies on the model efficiency. To this end, we examine the time consumption across 3 different computational environments, each with varying CPU capabilities and without GPU acceleration. We evaluate the inference time needed to predict all 4 QoS metrics for a range of concurrent RTP flows, from 1 to 11, as in Figure 3.

All models experience a relatively linear growth (notice the logarithm scale on y-axis) as the number of flows increases. Remarkably, *Oh*-student demonstrates distinctly lower time consumption, with a negligible magnitude of 1 ms irrespective of computational environments. At the same time, it does not scale up significantly with the addition of more flows, e.g., the time required for 11 flows even outpaces other models to address 1 single flow, illustrating the feasibility of applying our solution to complicated scenarios with more RTP flows. Such merits arise from the simplest FNN-only architecture and the mutual features for all tasks. Moreover, while *Oh*-teacher characterizes a sophisticated structure that is supposed to be temporally inefficient, it still generates relatively lower time consumption, thanks to the simultaneous processing of input in one shot and the shared features, unlocking the potentiality of utilizing *Oh*-teacher when performance is prioritized. As for other models, due to the inherent limitation of sequentially predicting a single metric for an individual flow each time, most of them fail to accomplish the task within a reasonable duration in the context of the targeted 500-ms time window, e.g., TimesNet takes up to 10 seconds for 11 flows. Additionally, although certain models such as PatchTST, LSTM-ts, and MLP, exhibit acceptable time consumption, their prediction performance remains insufficient. In essence, the time conserved by *Oh*-student empowers predictions to execute in a real-time manner, endowing us with a higher degree of flexibility to enact network optimization operations timely and promptly. Noteworthily, the performance herein might be further improved, because we do not factor in any other potential optimizations including well-known NN pruning techniques [45] and other effective NN architectures [46].

D. Ablation Study

In the following, we perform a series of ablation tests in order to understand and validate the contribution of specific model designs. In general, *Oh* surpasses its alternatives according to the results presented in Table V.

Ablation Test 1 — Regular Attention: We first substantiate our customized cross and neighbourhood attention by substituting it with a standard multi-head attention while retaining other components, including the LSTM NN. As elucidated by the result comparison, the default attention mechanism, which does not incorporate flow-wise correlations but considers the entire packet sequence per flow, cannot achieve the same level of performance as *Oh*-teacher for both datasets.

Ablation Test 2 — Transformer Encoder: Secondly, we maintain the customized attention structure but omit the LSTM NN, replacing it with a linear layer, whose output is also involved with a residual connection and layer normalization. As a result, we restore the general architecture of a Transformer encoder that is commonly adopted for sequence representation [47], [48], to assess the contribution of the LSTM in our design. According to the results, Transformer encoder with a customized structure delivers a comparatively equivalent outcome for bitrate and FPS for both datasets, demonstrating the innate capability of the attention mechanism. In regard of packet loss, Transformer encoder encounters

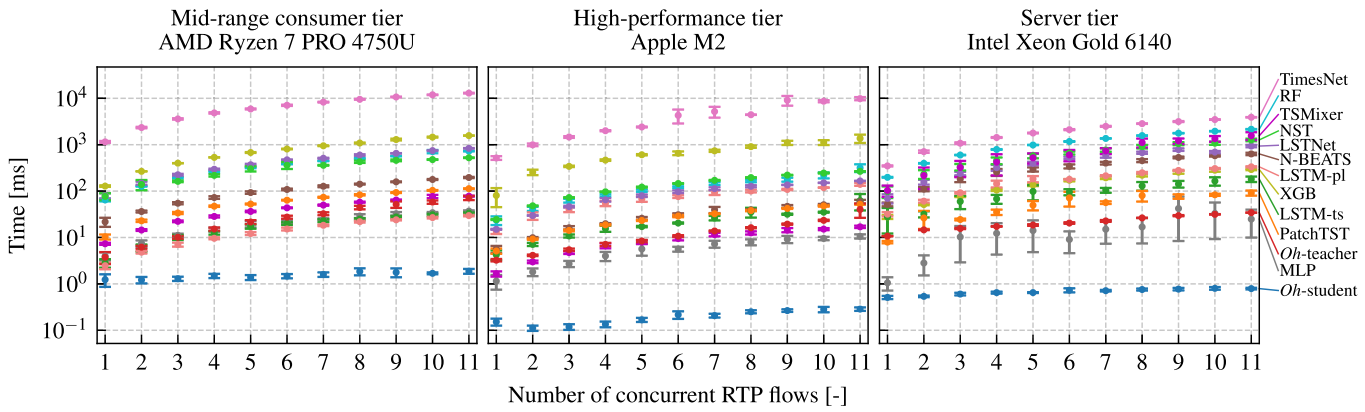


Fig. 3. Time consumption of different number of flows for all models in various computational environments.

a decline in dataset 2, while the performance gap is even more widened for average jitter, e.g., a decrease of roughly 0.02 and 0.04 in R^2 for datasets 1 and 2, respectively.

Ablation Test 3 — Plain LSTM: To evaluate the effectiveness of the attention mechanism that augments packet sequences with contextual information, we remove the cross and neighbourhood attention, channelling the enriched packets directly to the LSTM layer, and outputting the last hidden state to the following multi-task learning block. Similarly, a purely LSTM-based feature extraction block yields inferior performance except for a marginally improved classification result on dataset 2.

Ablation Test 4 — No Teacher Model: Here, we seek to verify the necessity of the knowledge distillation process by training from scratch an MLP with an architecture identical to *Oh*-student, but without the intervention from a teacher model, i.e., only the 4 task-specific losses are utilized. Apparently, a student model trained without guidance confronts a certain degree of performance degradation albeit an unexpected improvement for packet loss for dataset 2, indicating the intrinsic defect of a smaller model in learning a concise knowledge representation on its own, and underscoring the crucial role played by *Oh*-teacher.

Ablation Test 5 — Separate Training of Tasks: Previously, we affirm that the employment of multi-task learning aims not only to obtain shared features for model efficiency but also to capture the implicit correlations among different QoS metrics. To this end, we conduct 4 independent single-task learning for each prediction target, feeding the output of feature extraction block into a single FNN, and thus solely optimizing one task-related loss. The results from these experiments reveal mediocre performance, especially in predicting packet loss for dataset 2, where the model struggles to identify lossy windows.

E. Sensitivity Analysis

All results presented so far are derived based on a predicted time window of 500 ms using the preceding 128 packets. In this subsection, we evaluate the versatility and scalability of *Oh*, by elaborating on 4 alternative scenarios with different numbers of packets as features (L, 64 and 256 packets) and durations of time windows (Δt, 300 and 1000 ms). To compare the performance, we select 3 packet-level algorithms that demonstrate acceptable performance in accordance to Table IV instead of retraining all of the originally considered models including those proving to be incompetent. Notably, although time-series models exhibit adequate performance particularly for average jitter, they are still abandoned given their deficiency regarding packet loss prediction. Additionally, we only implement *Oh*-teacher herein, as the entire performance comparison process focuses on identifying the best choice for knowledge distillation. In other words, given that the student model is heavily reliant on its teacher, the performance of *Oh*-teacher across different scenarios can also reflect the potential of *Oh*-student, and therefore, there is no need to retrain and showcase the student model.

Table VI illustrates all the experimental results for both datasets. In general, *Oh*-teacher achieves overall distinguished performance, albeit with an acceptable degree of overfitting on dataset 2 and a similar performance for packet loss across models. With a smaller amount of 64 packets, *Oh*-teacher can still generate comparable outcomes with respect to the original 128 packets, demonstrating the possibility of further improving model efficiency thanks to the reduced input size. Interestingly, fewer packets advocate for lossless time windows for packet loss prediction, yielding an enhanced recall for class 0 but a diminished one for class 1, which is deemed a trivial issue, since the model can be tuned by adjusting the classification weight. Counterintuitively, using a large number of 256 packets does not lead to a better performance, possibly because of the fact that temporally distant packets in the past are not informative for capturing the latest network trends. In terms of a short predicted time window, *Oh*-teacher achieves so far the best performance for jitter prediction, with R^2 scores of nearly 0.9 and 0.87 for dataset 1 and 2, respectively. Comparing to other QoS metrics, we take an average of jitters of all packets within a window as the prediction target. Thus, a short time encompasses less packets, minimizing information loss when averaging the highly volatile jitter variations, which facilitates the model to more easily capture the most-recent traffic evolution. On top of that, given the minimal time consumption of *Oh*-student, the decent performance of 300 ms highlights the feasibility of implementing our solution when more frequent updates of QoS indicators are needed, not to mention the superior performance on dataset 2. For 1000 ms, *Oh*-teacher obtains further improved results regarding bitrate

TABLE VI
EXPERIMENTAL RESULTS OF BOTH DATASET FOR SENSITIVITY ANALYSIS

Model	Bitrate [Mbps]				Jitter [ms]				Frame per second [fps]				Packet loss [-]		
	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	Recall-0	Recall-1	F1
$L = 64$ packets:															
LSTM	0.101	0.030	8.35%	0.965	1.965	0.886	16.75%	0.841	1.703	0.730	5.01%	0.968	0.965	0.456	0.638
LSTNet	0.102	0.031	10.12%	0.965	1.921	0.913	16.33%	0.848	1.693	0.912	6.49%	0.968	**0.967**	0.459	**0.641**
TSMixer	0.099	**0.029**	8.48%	0.967	2.175	1.006	19.36%	0.805	1.718	0.768	5.31%	0.967	0.961	**0.476**	0.632
Oh-Teacher	**0.096**	**0.029**	**8.12%**	**0.969**	**1.691**	**0.807**	**15.90%**	**0.882**	**1.650**	**0.728**	**5.00%**	**0.970**	0.966	0.461	0.639
LSTM	0.116	0.039	10.37%	0.945	2.652	1.661	22.13%	0.772	2.569	**1.360**	**7.99%**	**0.917**	0.961	0.651	0.735
LSTNet	0.116	0.039	11.17%	0.945	2.409	1.494	20.44%	0.812	2.564	1.508	8.57%	**0.917**	**0.962**	0.653	**0.740**
TSMixer	**0.113**	**0.038**	**10.16%**	**0.947**	2.994	1.845	25.85%	0.709	2.689	1.402	8.01%	0.909	0.949	**0.667**	0.710
Oh-Teacher	0.117	0.042	10.92%	0.944	**2.196**	**1.352**	**20.05%**	**0.843**	**2.558**	1.399	8.15%	**0.917**	0.936	0.635	0.677
$L = 256$ packets:															
LSTM	0.100	0.029	8.20%	0.966	1.891	0.857	16.21%	0.853	1.717	0.775	5.32%	0.967	0.939	0.639	0.626
LSTNet	0.104	0.031	9.33%	0.963	2.030	0.950	16.25%	0.830	1.721	0.926	6.56%	0.967	0.944	0.628	0.633
TSMixer	0.143	0.049	11.69%	0.930	2.517	1.254	25.39%	0.739	1.938	1.232	8.79%	0.958	**0.956**	0.553	**0.641**
Oh-Teacher	**0.095**	**0.028**	**8.14%**	**0.970**	**1.725**	**0.768**	**13.82%**	**0.877**	**1.666**	**0.748**	**5.08%**	**0.969**	0.942	**0.646**	0.633
LSTM	**0.100**	**0.034**	**9.65%**	**0.959**	2.455	1.476	20.13%	0.804	2.505	**1.345**	**7.83%**	0.921	0.933	0.722	0.694
LSTNet	0.101	**0.034**	10.57%	0.958	2.476	1.498	19.66%	0.801	**2.431**	1.428	8.32%	**0.925**	0.931	**0.737**	0.694
TSMixer	0.117	0.042	12.16%	0.944	4.039	2.455	33.66%	0.471	2.721	1.631	9.42%	0.906	**0.951**	0.694	**0.723**
Oh-Teacher	0.112	0.042	11.45%	0.949	**2.335**	**1.391**	**20.02%**	**0.823**	2.527	1.486	9.01%	0.919	0.934	0.721	0.695
$\Delta t = 300$ ms:															
LSTM	0.125	0.042	11.45%	0.948	1.912	0.923	17.44%	0.849	2.326	1.562	12.64%	0.941	0.957	0.566	0.611
LSTNet	0.115	0.037	11.87%	0.956	1.785	0.844	15.41%	0.869	2.099	1.196	8.70%	**0.952**	**0.960**	0.535	0.612
TSMixer	0.110	0.036	12.57%	0.960	2.092	0.976	18.69%	0.820	2.142	1.007	6.79%	0.950	0.957	0.568	0.611
Oh-Teacher	**0.105**	**0.033**	**9.94%**	**0.963**	**1.579**	**0.725**	**13.56%**	**0.897**	**2.097**	**0.973**	**6.78%**	**0.952**	0.959	**0.570**	**0.616**
LSTM	0.134	0.045	14.33%	0.928	2.172	1.326	19.40%	0.848	3.423	2.452	16.34%	0.872	**0.954**	0.757	0.727
LSTNet	0.120	**0.040**	13.75%	0.942	2.128	1.260	17.72%	0.854	3.161	2.099	12.68%	0.890	**0.954**	0.762	**0.728**
TSMixer	0.119	0.041	13.31%	0.943	3.007	1.824	26.23%	0.708	3.166	1.892	11.48%	0.887	0.953	**0.763**	0.724
Oh-Teacher	**0.116**	**0.040**	**11.90%**	**0.945**	**2.039**	**1.219**	**17.24%**	**0.866**	**3.039**	**1.827**	**11.47%**	**0.896**	0.953	0.757	0.725
$\Delta t = 1000$ ms:															
LSTM	0.121	0.044	10.97%	0.950	2.215	1.061	19.96%	0.796	1.906	1.472	9.61%	0.960	**0.951**	0.557	**0.677**
LSTNet	0.100	0.031	12.51%	0.966	2.210	1.003	16.98%	0.797	1.401	0.667	4.97%	0.978	0.939	0.572	0.659
TSMixer	**0.094**	**0.025**	7.50%	**0.970**	2.505	1.208	21.93%	0.739	1.372	0.606	4.43%	0.979	0.934	**0.584**	0.653
Oh-Teacher	**0.094**	0.026	**7.40%**	**0.970**	**2.098**	**0.936**	**16.45%**	**0.817**	**1.330**	**0.545**	**4.00%**	**0.980**	0.917	0.577	0.626
LSTM	0.130	0.045	10.86%	0.930	**2.425**	**1.496**	21.03%	**0.804**	2.483	1.876	11.56%	0.918	**0.940**	0.599	**0.706**
LSTNet	0.101	0.035	11.97%	0.957	2.459	**1.496**	**20.05%**	0.798	2.115	1.160	8.08%	0.941	0.934	0.604	0.698
TSMixer	**0.090**	**0.031**	**8.68%**	**0.967**	3.592	2.230	29.27%	0.576	2.186	1.102	**7.00%**	0.937	0.907	0.622	0.663
Oh-Teacher	0.107	0.038	9.38%	0.953	2.909	1.805	24.51%	0.722	**2.108**	**1.089**	7.25%	**0.942**	0.829	**0.658**	0.591

* For each sub-block associated to a certain parameter, the upper part is for dataset 1 and the lower is for dataset 2.

and FPS, ascribed to an elongated duration that smooths out fluctuations of metrics. More importantly, a longer time window affords an elevated degree of freedom to enforce network optimization measures, despite of a minor performance decline for average jitter. Notably, we also evaluate various edge conditions (e.g., limited bandwidth availability) and observe generally consistent performance, but these results are omitted here due to space constraints. In a word, *Oh*-teacher appears to be a satisfactory candidate regardless of system configurations, although we observe instances of overfitting across domains that prove to be solvable using established techniques such as transfer [49] and incremental learning [50].

VI. CHALLENGES

Herein, we provide an in-depth discussion of particular challenges and potential remedies.

A. Generalizability

In our work, we adopt two distinct datasets from ample video-call traffic which, despite spanning different periods and applications, may not offer exhaustive coverage for evaluating model generalizability. Thus, we consider additional traces to construct a general RTC traffic dataset, collected via Tstat [51] that dumps application-agnostic RTP traffic at our campus ingress node. It is based on traffic during a randomly selected hour[2] in late 2023 and contains over 200,000 samples.

By evaluating both teacher and student models, we attain satisfactory performance for the majority of traffic (roughly 70%), while observing abnormal behaviors[3] in a minor subset (30%), as in Table VII. Given the broad coverage of RTC applications across the entire campus, the competitive performance achieved from abundant new traffic fundamentally validates the model generalizability. Although abnormal traces experience noticeable performance drop, we discover peculiar patterns upon closer inspection. For example, the oddly high MAPEs (around 40% higher than generalized traffic) alongside acceptable R^2 scores and relatively low RMSEs/MAEs (merely about 0.02 higher) denote an overall ultra-low bitrate in abnormal traffic such that even tiny

[2]As it includes the entire population and all incoming RTP traffic of the university, an hour of traffic is deemed representative and voluminous.

[3]Flows exhibiting subpar performance (e.g., those with a MAPE exceeding 20% for bitrate or 30% for jitter) are considered abnormal. The definition of anomaly in this context is trivial, as the predictions evidently deviate from the ground truth.

TABLE VII
EXPERIMENTAL RESULTS ON THE GENERAL RTC TRAFFIC DATASET

Model	Bitrate [Mbps]				Jitter [ms]				Frame per second [fps]				Packet loss [-]		
	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	RMSE	MAE	MAPE	R^2	Recall-0	Recall-1	F1
Generalized traffic (69.2% of the dataset):															
Oh-Teacher	0.091	0.044	16.48%	0.953	2.057	1.303	21.08%	0.881	2.665	1.601	8.69%	0.883	0.974	0.534	0.682
Oh-Student	0.103	0.045	18.33%	0.952	2.456	1.682	28.12%	0.844	2.317	1.540	8.28%	0.911	0.987	0.521	0.736
Abnormal traffic (30.8% of the dataset):															
Oh-Teacher	0.118	0.065	60.35%	0.911	8.899	4.586	444%	-0.31	3.228	1.936	19.03%	0.823	0.862	0.653	0.598
Oh-Student	0.143	0.063	75.79%	0.884	11.75	7.387	734%	-1.29	4.639	3.253	44.99%	0.634	0.916	0.670	0.664

prediction deviations result in disproportionately inflated MAPE values. Moreover, the failure in jitter prediction originates from the inapplicability of standard jitter inference based on session log or packet frequency, rendering the ground truth for jitter in these cases unreliable. Taken together, these findings suggest that anomalous traffic deviates from standard RTP implementations, significantly modifying transmission patterns and thus hampering the transferability of *Oh*. Nevertheless, this is not deemed a critical issue, as abnormal traffic constitutes a limited proportion, and the corresponding RTC applications that appear to feature a low bitrate generally do not necessitate sophisticated QoS predictions.

Indeed, model generalizability remains an arduous challenge inherent in DL realm, and performance degradation is often inevitable due to domain shift (e.g., audio and video can be mixed in the same flow for cloud gaming traffic [52]). Evidently, *Oh* manages to deliver satisfactory results for numerous instances, with potential for further enhancement through modern techniques—a direction worth pursuing in future work.

B. Extension to Other QoS Metrics

Although the 4 targeted QoS metrics we select represent vital criteria in RTC, they remain imperfect due to the absence of other informative indicators. Beyond metrics deemed unsuitable, such as packet count that is highly correlated with bitrate, thus conveying redundant information, the primary concern arises from the unavailability of specialized information in the dataset. Our dataset is intentionally collected in a general way to utilize easily-accessible off-the-shelf tools for the sake of generating ample traffic, whereas additional QoS metrics such as end-to-end delay require a more sophisticated (potentially prohibitive) experimental setup or a tightly controlled environment, which falls outside our current scope. Nevertheless, it is conceivable that our model can be readily extended to predict other metrics by integrating supplementary tasks, given that the task-specific NN operates in a plug-and-play fashion, leveraging shared features without requiring substantial model reconstruction or feature re-engineering.

C. Real-World Deployment

Our solution is envisaged to work in real time, facilitating QoS-aware, proactive, and generalized RTC management. The main challenges concerning real-world deployment stem from two aspects: *1)* Run-time overhead—while real-world deployment often entails more complex requirements, *Oh*-student excels in efficiency, remaining a viable solution attributed not only to the simplicity of its pure MLP architecture but also to modern programming best practices and well-established techniques such as DPDK [53] for prompt packet access. *2)* Dynamic network environments—while our model is trained on extensive traffic, it is barely possible to achieve an exhaustive and permanent fix in the long run given the ever-evolving network dynamics, thus leading to subpar performance occasionally. Fortunately, this issue (while perhaps not entirely resolvable) can be substantially alleviated thanks to matured ML pipelines and powerful MLOps. Despite being relatively impractical in our current stage, it is still crucial to examine the feasibility of eventual implementation, which warrants one of our future directions.

VII. RELATED WORK

Understanding the performance of RTC applications is always a critical effort [54], [55], [56], and numerous works have sought to tackle the difficulties in QoS monitoring in modern scopes [3], [57], [58], [59]. On top of that, a considerable amount of research has explored QoS prediction using ML/DL technologies in RTC [5], [6], [60], [61]. In [62], the authors focused on video QoE estimation, predicting 5 key indicators such as average bitrate, using a bidirectional LSTM-based convolutional neural network (CNN) to investigate the performance across 3 different scenarios. The studies in [63] and [64] aimed to predict multiple QoS metrics, e.g., jitter and packet count. The former framed a data mining problem and introduced a Support Vector Regressor (SVR) that takes a series of network key indicators as input. The latter centered on the video quality under time-varying loads, and proposed a load-adjusted learning scheme, wherein 4 conventional regression methods are implemented. Reference [65] examined an array of ML algorithms, and proposed a group of DL models like LSTM to predict various QoS metrics including latency, received/sent packet counts, and download/upload traffic rates for real-time videos. In terms of individual metrics, more specialized and sophisticated models have emerged. For traffic prediction, several works from [66], [67], and [68] adopted various approaches to forecast bitrate. The first discussed an online ML system to predict future average bitrate along with resolution for encrypted video streaming, the second integrated LSTM in a RL framework as the bitrate predictor to optimize QoE, and the third employed the autoregressive integrated moving average (ARIMA) method to forecast the rate of upcoming packet sequences for video streaming in 4G wireless networks. Meanwhile, multiple works have addressed the problem of throughput prediction [69], [70], wherein the authors intended to improve adaptive streaming by accurately forecasting throughput, and they proposed a decision

tree-based regressor and a CNN-augmented RL framework. Moreover, delay prediction has also been studied by the works of [71] and [72], with LSTM and Transformer-based NNs being developed, respectively. Additionally, [73], [74] have investigated the packet loss prediction in various scenarios using a range of ML approaches, including random forest, gradient boosting tree, artificial NN, etc. In summary, although numerous works, which either incorporate QoS within broader frameworks or focus exclusively on predictive purposes, characterize different requirements, they typically rely on general existing ML/DL technologies with minor customization under the hood.

Furthermore, QoS prediction can be generally formulated as a time series forecasting problem, where inputs are either univariate historical data or multivariate correlated features, and an advanced DL model commonly serves as the predictor [75], [76]. However, recent trends of long-term forecasting [77] and LLM [78] are ill-suited to our context, while other DL approaches tend to employ sophisticated yet costly architecture [79], [80], or are tailored to canonical datasets that markedly diverge from our scenario [81], [82]. Given the rapid evolution and sheer volume of deep time-series models emerging in the vibrant community, identifying the most appropriate solution for our problem is non-trivial. Therefore, we refer to [83], selecting and implementing several top-performing and applicable models, such as TimesNet [43] that disentangles temporal patterns into multiple intraperiod- and interperiod-variations and proposes a residual stack of TimesBlocks, and Non-stationary Transformer [41] that incorporates series stationarization and de-stationary attention modules to unify sample statistics and reintroduce non-stationary signals into temporal dependencies.

To the best of our knowledge, our work represents the first endeavor that aims to efficiently predict multiple QoS metrics on a per-flow basis. The conception originates from two key factors: on one hand, we have previously explored employing ML/DL techniques in RTP traffic predictions [84], [85], [86], such as uncovering patterns for packet loss and throughput prediction with emphasis on traffic extremes, which helped us to recognize the limitations of developing isolated QoS prediction methods and laid the groundwork for the innovation of a more unified, versatile approach. On the other hand, despite the effectiveness of solutions in the literature, existing approaches are tailored specifically to a single target, but all QoS metrics, whether considered independently or cooperatively, could offer instructive feedback to support multitudinous network functions and a comprehensive management system, eliminating the need as well as redundancy of implementing several standalone and incompatible prediction techniques. Such a prospect becomes increasingly efficacious and attainable, especially as our predictions are anchored in each flow, proffering fine-grained information.

VIII. Conclusion

In this paper, we propose a DL framework, named *Oh*, stating that *o*ne model is enoug*h* to predict various QoS metrics for multiple concurrent RTP flows in RTC sessions. Existing approaches, which can only handle one metric or flow at a time, are inherently inefficient. To surmount the challenges, *Oh* employs a teacher-student paradigm, wherein the teacher model seeks to apprehend network intricacies by modeling both flow- and packet-wise correlations, and distills knowledge into a more efficient student model. Particularly, the teacher integrates a custom attention mechanism that augments an LSTM NN to extract traffic features, while the student characterizes a simpler FNN-only architecture. To predict various QoS metrics, we adopt a multi-task learning scheme, mapping shared features to 4 target values, i.e., bitrate, average jitter, FPS, and loss condition. Our work is based on massive traffic collected during real video-teleconferencing under diverse conditions. We formulate a multivariate, multi-target time-series problem, comparing against a variety of ML/DL algorithms. Meanwhile, we conduct thorough ablation studies, and elaborate on additional experimental configurations. Consequently, *Oh*-teacher achieves superior outcomes, positioning it as a promising candidate for knowledge distillation, and *Oh*-student results in a negligible amount of time consumption while guaranteeing a comparable level of performance. In future work, we aspire to investigate the possibility of applying our methodology to a broader range of QoS metrics, contributing to an even more systematic and comprehensive network observability and management.

Acknowledgment

The authors dedicate this paper to the memory of their colleague and dear friend, Maurizio Matteo Munafò, who passed away prematurely shortly before the publication of this work. His scientific insight, generous support, and friendship will be deeply missed. This work was also supported by computational resources provided by hpc@polito (http://www.hpc.polito.it).

References

[1] R. Frederick, S. L. Casner, V. Jacobson, and H. Schulzrinne, *RTP: A Transport Protocol for Real-Time Applications*, document RFC 1889, Jan. 1996. [Online]. Available: https://rfc-editor.org/rfc/rfc1889.txt

[2] G. Carofiglio, G. Grassi, E. Loparco, L. Muscariello, M. Papalini, and J. Samain, "Characterizing the relationship between application QoE and network QoS for real-time services," in *Proc. ACM SIGCOMM Workshop Netw. Appl. Integr. (NAI)*, 2021, pp. 20–25.

[3] K. MacMillan, T. Mangla, J. Saxon, and N. Feamster, "Measuring the performance and network utilization of popular video conferencing applications," in *Proc. 21st ACM Internet Meas. Conf.*, 2021, pp. 229–244.

[4] N. Rao et al., "Analysis of the effect of QoS on video conferencing QoE," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 1267–1272.

[5] O. Izima, R. de Fréin, and A. Malik, "A survey of machine learning techniques for video quality prediction from quality of delivery metrics," *Electronics*, vol. 10, no. 22, p. 2851, Nov. 2021.

[6] M. Morshedi and J. Noll, "A survey on prediction of PQoS using machine learning on Wi-Fi networks," in *Proc. Int. Conf. Adv. Technol. Commun. (ATC)*, Oct. 2020, pp. 5–11.

[7] S. Loreto and S. P. Romano, *Real-time Communication with WebRTC: Peer-to-peer in the Browser*. Sebastopol, CA, USA: O'Reilly Media, 2014.

[8] S. Huang and J. Xie, "DAVE: Dynamic adaptive video encoding for real-time video streaming applications," in *Proc. 18th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jul. 2021, pp. 1–9.

[9] L. Liu, J. Li, H. Xu, K. Xue, and J. C. Xue, "Efficient real-time video conferencing with adaptive frame delivery," *Comput. Netw.*, vol. 234, Oct. 2023, Art. no. 109918.

[10] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the Google congestion control for web real-time communication (WebRTC)," in *Proc. 7th Int. Conf. Multimedia Syst.*, May 2016, pp. 1–12.

[11] S. Yan, Y. Guo, Y. Chen, F. Xie, C. Yu, and Y. Liu, "Enabling qoe learning and prediction of webrtc video communication in wifi networks," in *Proc. ICC*, 2017.

[12] T. Begluk, J. B. Husic, and S. Barakovic, "Machine learning-based QoE prediction for video streaming over LTE network," in *Proc. 17th Int. Symp. INFOTEH-JAHORINA (INFOTEH)*, Mar. 2018, pp. 1–5.

[13] E. Torres, R. Reale, L. Sampaio, and J. Martins, "A SDN/OpenFlow framework for dynamic resource allocation based on bandwidth allocation model," *IEEE Latin Amer. Trans.*, vol. 18, no. 5, pp. 853–860, May 2020.

[14] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, 2021.

[15] A. Dainotti, A. Pescapé, P. S. Rossi, F. Palmieri, and G. Ventre, "Internet traffic modeling by means of hidden Markov models," *Comput. Netw.*, vol. 52, no. 14, pp. 2645–2662, Oct. 2008.

[16] A. Montieri, G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapè, "Packet-level prediction of mobile-app traffic using multitask deep learning," *Comput. Netw.*, vol. 200, Dec. 2021, Art. no. 108529.

[17] A. Vaswani et al., "Attention is all you need," 2017, *arXiv:1706.03762*.

[18] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are transformers effective for time series forecasting?," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 9, pp. 11121–11128.

[19] Y. Li, Z. Zhu, D. Kong, H. Han, and Y. Zhao, "EA-LSTM: Evolutionary attention-based LSTM for time series prediction," *Knowledge-Based Syst.*, vol. 181, Oct. 2019, Art. no. 104785.

[20] H. Abbasimehr and R. Paki, "Improving time series forecasting using LSTM and attention models," *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 1, pp. 673–691, Jan. 2022.

[21] G. Perna et al., "Real-time classification of Real-time communications," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4676–4690, Apr. 2022.

[22] A. Nistico, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio, "A comparative study of RTC applications," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Dec. 2020, pp. 1–8.

[23] T. Song, D. Markudova, G. Perna, and M. Meo, "Where did my packet go? real-time prediction of losses in networks," in *Proc. IEEE Int. Conf. Commun.*, May 2023, pp. 3836–3841.

[24] A. Bertsch, U. Alon, G. Neubig, and M. R. Gormley, "Unlimiformer: Long-range transformers with unlimited length input," in *Proc. 37th Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 35522–35543.

[25] Q. Wen et al., "Transformers in time series: A survey," 2022, *arXiv:2202.07125*.

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[27] S. Siami-Namini, N. Tavakoli, and A. S. Namin, "The performance of LSTM and BiLSTM in forecasting time series," in *Proc. IEEE Int. Conf. Big Data*, Dec. 2019, pp. 3285–3292.

[28] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Netw.*, vol. 116, pp. 237–245, Aug. 2019.

[29] B. Krause, L. Lu, I. Murray, and S. Renals, "Multiplicative LSTM for sequence modelling," 2016, *arXiv:1609.07959*.

[30] D. Soutner and L. Müller, "Application of LSTM neural networks in language modelling," in *Proc. 16th Int. Conf. Text, Speech, Dialogue*, Pilsen, Czech Republic. Cham, Switzerland: Springer, 2013, pp. 105–112.

[31] A. Hassani, S. Walton, J. Li, S. Li, and H. Shi, "Neighborhood attention transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 6185–6194.

[32] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, Jun. 2018, pp. 7482–7491.

[33] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*.

[34] J. Xu, Z. Li, B. Du, M. Zhang, and J. Liu, "Reluplex made more practical: Leaky ReLU," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–7.

[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[36] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[37] T. Chen et al., "Xgboost: Extreme gradient boosting," *R Package Version*, vol. 1, no. 4, pp. 1–4, 2015.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

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

[40] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling long-and short-term temporal patterns with deep neural networks," in *Proc. Int. ACM Conf. Res. Devel. Inf. Retrieval (SIGIR)*, 2018, pp. 95–104.

[41] Y. Liu, H. Wu, J. Wang, and M. Long, "Non-stationary transformers: Exploring the stationarity in time series forecasting," in *Proc. 36th Conf. Neural Inf. Process. Syst.*, 2022, pp. 9881–9893.

[42] Y. Nie, N. H. Nguyen, P. Sinthong, and J. Kalagnanam, "A time series is worth 64 words: Long-term forecasting with transformers," 2022, *arXiv:2211.14730*.

[43] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long, "TimesNet: Temporal 2D-variation modeling for general time series analysis," 2022, *arXiv:2210.02186*.

[44] S.-A. Chen, C.-L. Li, S. O. Arik, N. C. Yoder, and T. Pfister, "TSMixer: An All-MLP architecture for time series forecasting," *Trans. Mach. Learn. Res.*, vol. 2023, 2023.

[45] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[46] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surveys*, vol. 55, no. 6, pp. 1–28, Dec. 2022, doi: 10.1145/3530811.

[47] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[48] A. T. Liu, S.-W. Li, and H.-Y. Lee, "TERA: Self-supervised learning of transformer encoder representation for speech," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 2351–2366, 2021.

[49] M. Iman, H. R. Arabnia, and K. Rasheed, "A review of deep transfer learning and recent advancements," *Technologies*, vol. 11, no. 2, p. 40, Mar. 2023.

[50] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," in *Proc. Eur. Symp. Artif. neural Netw. (ESANN)*, 2016.

[51] M. Mellia, A. Carpani, and R. L. Cigno, "TStat: TCP STatistic and analysis tool," in *Proc. Int. Workshop Quality Service Multiservice IP Netw.* Cham, Switzerland: Springer, 2003, pp. 145–157.

[52] A. Di Domenico, G. Perna, M. Trevisan, L. Vassio, and D. Giordano, "A network analysis on cloud gaming: Stadia, GeForce now and PSNow," *Network*, vol. 1, no. 3, pp. 247–260, Oct. 2021.

[53] (2025). *DPDK: Data Plane Development Kit*. Accessed: Apr. 19, 2025. [Online]. Available: https://www.dpdk.org/

[54] B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, and G. Zussman, "Performance evaluation of WebRTC-based video conferencing," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 3, pp. 56–68, Mar. 2018.

[55] M. Varvello, H. Chang, and Y. Zaki, "Performance characterization of videoconferencing in the wild," in *Proc. 22nd ACM Internet Meas. Conf.*, Oct. 2022, pp. 261–273.

[56] J. Skowronek et al., "Quality of experience in telemeetings and videoconferencing: A comprehensive survey," *IEEE Access*, vol. 10, pp. 63885–63931, 2022.

[57] T. Sharma, T. Mangla, A. Gupta, J. Jiang, and N. Feamster, "Estimating WebRTC video QoE metrics without using application headers," in *Proc. ACM Internet Meas. Conf.*, Oct. 2023, pp. 485–500.

[58] O. Michel, S. Sengupta, H. Kim, R. Netravali, and J. Rexford, "Enabling passive measurement of zoom performance in production networks," in *Proc. 22nd ACM Internet Meas. Conf.*, Oct. 2022, pp. 244–260.

[59] H. Chang, M. Varvello, F. Hao, and S. Mukherjee, "Can you see me now?: A measurement study of zoom, webex, and meet," in *Proc. 21st ACM Internet Meas. Conf.*, Nov. 2021, pp. 216–228.

[60] G. Kougioumtzidis, V. Poulkov, Z. D. Zaharis, and P. I. Lazaridis, "A survey on multimedia services QoE assessment and machine learning-based prediction," *IEEE Access*, vol. 10, pp. 19507–19538, 2022.

[61] L. N. Onyejegbu, U. A. Okengwu, L. U. Oghenekaro, M. O. Musa, and A. O. Ugbari, "AI-based QOS/QOE framework for multimedia systems," in *Proc. Future Technol. Conf.*, 2022, pp. 248–259.

[62] H. E. Dinaki, S. Shirmohammadi, E. Janulewicz, and D. Côté, "Forecasting video QoE with deep learning from multivariate time-series," *IEEE Open J. Signal Process.*, vol. 2, pp. 512–521, 2021.

[63] S. Handurukande, S. Fedor, S. Wallin, and M. Zach, "Magneto approach to QoS monitoring," in *Proc. 12th IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM) Workshops*, May 2011, pp. 209–216.

[64] O. Izima, R. de Fréin, and M. Davis, "Video quality prediction under time-varying loads," in *Proc. IEEE Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Dec. 2018, pp. 129–132.

[65] L. Babooram and T. P. Fowdur, "Performance analysis of collaborative real-time video quality of service prediction with machine learning algorithms," *Int. J. Data Sci. Analytics*, vol. 20, no. 2, pp. 1–33, Aug. 2025.
[66] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Let me decrypt your beauty: Real-time prediction of video resolution and bitrate for encrypted video streaming," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2019, pp. 199–200.
[67] A. Lekharu, K. Moulii, A. Sur, and A. Sarkar, "Deep learning based prediction model for adaptive video streaming," in *Proc. Int. Conf. Commun. Syst. NETw. (COMSNETS)*, Sep. 2020, pp. 152–159.
[68] D. Kumar, S. Aishwarya, A. Srinivasan, and L. A. Raj, "Adaptive video streaming over HTTP using stochastic bitrate prediction in 4G wireless networks," in *Proc. ITU Kaleidoscope, ICTs Sustain. World*, Nov. 2016, pp. 1–8.
[69] G. Lv, Q. Wu, W. Wang, Z. Li, and G. Xie, "Lumos: Towards better video streaming QoE through accurate throughput prediction," in *Proc. IEEE Conf. Comput. Commun.*, May 2022, pp. 650–659.
[70] J. Yin, Y. Xu, H. Chen, Y. Zhang, S. Appleby, and Z. Ma, "ANT: Learning accurate network throughput for better adaptive video streaming," 2021, *arXiv:2104.12507*.
[71] Z. Zhang, J. Huang, L. Tian, C. Yuan, and Y. Wang, "Delay prediction for real-time video based on improved LSTM," in *Proc. IEEE 8th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2022, pp. 2053–2057.
[72] A. Dietmüller, S. Ray, R. Jacob, and L. Vanbever, "A new hope for network model generalization," in *Proc. 21st ACM Workshop Hot Topics Netw.*, Nov. 2022, pp. 152–159.
[73] I. Ali, S. Hong, and T. Cheung, "Congestion or no congestion: Packet loss identification and prediction using machine learning," 2024, *arXiv:2408.03007*.
[74] A. Giannakou, D. Dwivedi, and S. Peisert, "A machine learning approach for packet loss prediction in science flows," *Future Gener. Comput. Syst.*, vol. 102, pp. 190–197, Jan. 2020.
[75] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, "Deep learning for time series forecasting: A survey," *Big data*, vol. 9, no. 1, pp. 3–21, 2021.
[76] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Phil. Trans. Roy. Soc. A*, vol. 379, no. 2194, 2021, Art. no. 20200209.
[77] Y. Li et al., "Towards long-term time-series forecasting: Feature, pattern, and distribution," in *Proc. IEEE 39th Int. Conf. Data Eng. (ICDE)*, Apr. 2023, pp. 1611–1624.
[78] T. Althoff, V. Gupta, T. Hartvigsen, M. Merrill, and M. Tan, "Are language models actually useful for time series forecasting?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, pp. 60162–60191.
[79] H. Wu, J. Xu, J. Wang, and M. Long, "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2021, pp. 22419–22430.
[80] H. Zhou et al., "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, May 2021, pp. 11106–11115.
[81] M. Fliess, C. Join, and C. Voyant, "Prediction bands for solar energy: New short-term time series forecasting techniques," *Sol. Energy*, vol. 166, pp. 519–528, May 2018.
[82] X. Liu, Z. Lin, and Z. Feng, "Short-term offshore wind speed forecast by seasonal ARIMA—A comparison against GRU and LSTM," *Energy*, vol. 227, Jul. 2021, Art. no. 120492.
[83] Y. Wang et al., "Deep time series models: A comprehensive survey and benchmark," 2024, *arXiv:2407.13278*.
[84] T. Song, G. Perna, P. Garza, M. Meo, and M. M. Munafo, "Packet loss in real-time communications: Can ML tame its unpredictable nature?," *IEEE Trans. Netw. Service Manage.*, vol. 22, no. 1, pp. 72–91, Feb. 2025.
[85] T. Song, G. Perna, P. Garza, M. Meo, and M. M. Munafò, "BitFormer: Transformer-based neural network for bitrate prediction in real-time communications," in *Proc. IEEE 21st Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2024, pp. 65–70.
[86] T. Song, P. Garza, M. Meo, and M. M. Munafò, "DeX: Deep learning-based throughput prediction for real-time communications with emphasis on traffic eXtremes," *Comput. Netw.*, vol. 249, Jul. 2024, Art. no. 110507.



Tailai Song (Member, IEEE) received the B.Sc. degree in automotive engineering and the M.Sc. degree in ICT for smart societies from the Politecnico di Torino (PoliTO), Italy, in 2020 and 2022, respectively. He is currently pursuing the Ph.D. degree in the telecommunication networks group (TNG) with the Department of Electronics and Telecommunications (DET), PoliTO. He is a member of the SmartData@Polito Research Centre, PoliTO. 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 (Member, IEEE) received the master's and Ph.D. degrees in computer engineering from the Politecnico di Torino. He has been an Associate Professor with the Dipartimento di Automatica e Informatica, Politecnico di Torino, since December 2018. He spent three years as an Assistant Professor with Politecnico di Milano. He co-authored about 100 articles in the areas of data mining and machine learning. He has worked on classification, clustering, itemset mining, and scalable algorithms. His current research interests include data mining, database systems, and big data analytics.



Michela Meo (Senior Member, IEEE) is currently a Professor of telecommunication engineering with the Politecnico di Torino. She edited a book *Green Communications* (Wiley) and several special issues of international journals. She co-authored about 200 articles, 80 of which on international journals. Her research interests include green networking, energy-efficient mobile networks and data centers, the Internet traffic classification, and characterization. In the role of the General or Technical Chair, she has led the organization of several conferences, including ITC, ICC symposia, and 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. She was an Associate Editor of IEEE/ACM TRANSACTIONS ON NETWORKING, Green Series of IEEE JOURNAL ON SELECTED AREAS OF COMMUNICATIONS NETWORKING, and IEEE COMMUNICATION SURVEYS AND TUTORIALS. She is an Senior Editor of IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING.



Maurizio Matteo Munafò received the Dr.Ing. degree in electronic engineering and the Ph.D. degree in telecommunications engineering from the Politecnico di Torino in 1991 and 1994, respectively. He was an Assistant Professor with the Department of Electronics and Telecommunications, Politecnico di Torino. He has co-authored about 80 journals and conference papers in the area of communication networks and systems. His current research interests include simulation and performance analysis of communication systems and traffic modeling, measurement, and classification.