Doctoral Dissertation
Doctoral Program in Electrical, Electronics and Communications Engineering
(38^{th}cycle)

From Coarse Single Theoretical to Fine Multiple Practical: Machine Learning-Empowered Real-Time Communications

By

Tailai Song

******

Supervisor(s):
Prof. Michela Meo, Supervisor
Prof. Marco Mellia, Co-Supervisor

Doctoral Examination Committee:
Prof. J.J. Garcia-Luna-Aceves, University of Toronto
Prof. Thomas Erich Zinner, Norwegian University of Science and Technology
Prof. Danilo Giordano, Politecnico di Torino

Politecnico di Torino
2026

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Tailai Song
2026

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Acknowledgements

First and foremost, I would like to express my deepest gratitude and esteem to my supervisor—Prof. Michela Meo, a professional, dependable mentor and an amiable, respectable lady—for her comprehensive guidance, steadfast support, and thoughtful consideration. I genuinely feel grateful and lucky for every opportunity that you have provided, and it is your reassuring trust and encouragement that helped me build confidence and develop career path.

Second, I would like acknowledge my previous group members, Dr. Dena Markudova and Dr. Gianluca Perna, for their cooperation and suggestions, which introduced me to the practical aspects in the research world and were especially important during my early PhD stage. Equal gratitude to the group advisors: Prof. Paolo Garza and Prof. Maurizio Matteo Munafò[1]—thank you sincerely for your immense support and every "picky" advice that indicated the missing or deficient parts of my work. Furthermore, I would like to thank my co-supervisor, Prof. Marco Mellia, and I am very happy to have a boss like you in the office. Although we have not officially collaborated, your helps and suggestions have been indeed significant throughout the three years.

Third, I would like to thank my reviewers, Prof. J.J. Garcia-Luna-Aceves and Prof. Thomas Erich Zinner, for your valuable time on reading this thesis and giving me instrumental feedback. Also, I would like to thank Prof. Danilo Giordano for kindly agreeing to serve as the chair of my committee. It is a pleasure and an honor to share my work with scholars of your caliber.

Moreover, I would like to thank the sponsorship from Cisco Systems Inc. represented by Dr. Giovanna Carofiglio, who also has provided the opportunity of

[1] Sadly, Prof. Munafò passed away on 29 November, a loss deeply felt by all who knew him. May he rest in peace.

academia-industry collaboration, and the research infrastructure and computational resources provided by SmartData@PoliTO center and HPC@PoliTO infrastructures.

Additionally, I would like to thank other co-authors and collaborators: Prof. Daniela Renga, Prof. David Lopez, Dr. Nicola Piovesan, Dr. Mukharbek Organokov, and Dr. Lennart Gulikers, from who I have truly learned a lot. I also want to thank Prof. Luca Vassio. It has been a pleasure and valuable experience working with you on the teaching activity. I would like to thank my advisor during the period abroad, Dr. Pedro Casas, who has offered me an invaluable chance and guidance to work on an interesting project. Many thanks to my friends and colleagues, and everyone else, who have provided helps and advices to, and worked with me.

Last, to my beloved parents, sincerely thank you for your emotional and financial supports for years. Language is not enough; I love you.

Abstract

In the current era of pervasive Internet access and rapidly expanding network infrastructures, real-time communications (RTC) have ascended as a linchpin of modern digital interaction, underpinning a wide spectrum of use cases ranging from remote work and telemedicine to online gaming and live media streaming. These applications entail ever-escalating demands for reliability, ultra-low latency, adaptive performance, and technologically advanced, robust, and scalable frameworks. In this context, machine learning (ML) techniques, which have flourished in recent years, have emerged as promising candidates for intelligently capturing complex network dynamics and enabling adaptive, data-driven, and proactive optimization of RTC systems.

This thesis centers on Real-time Transport Protocol (RTP)-based RTC, exploring various facets of ML algorithms—from coarse to fine features, from single to multiple objectives, and from theoretical examination to practical implementations—to progressively augment network performance and eventually elevate the Quality of Experience (QoE) in RTC applications.

We commence the research with an in-depth investigation of packet loss phenomena, aiming to preemptively identify the onset of future losses and thereby potentially avoid subsequent ramifications. We discover the predominance of continuous losses and implement as well as compare multiple ML approaches based on aggregated traffic features, successfully classifying the majority of lossy events without significantly penalizing normal conditions.

Secondly, we turn our attention to the well-established problem of traffic prediction, striving to estimate network throughput with a dedicated emphasis on traffic extremes, including peaks, valleys, and abrupt changes. Leveraging fine-grained packet-level information, we develop an innovative deep learning (DL) algorithm capable of adeptly recognizing/accommodating extremities.

Moving forward to multiple objectives, we next propose a multi-task learning framework that integrates packet- and flow-level correlations with a knowledge distillation paradigm to efficiently predict a range of Quality of Service (QoS) metrics for each RTP flow in a single shot.

Lastly, we transition from theoretical feasibility analyses to the evaluation of practical performance gains. We scrutinize the congestion control (CC) mechanism in RTC, refuting the necessity of reinforcement learning (RL) and ascertaining the deterministic role of bandwidth estimation (BWE), which motivates the design of a simple yet effective regressor.

In summary, we conduct a comprehensive study of RTC at both the application and network levels, uncovering latent deficiencies and formulating various problems. In response, we provide ML-based remedies with different, specific targets, while progressively addressing pertinent constraints. Our work is anticipated to not only advance the state of the art in RTC optimization but also pave the way for more resilient, adaptive, and efficient RTC systems in increasingly complicated network environments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Have you ever experienced blurry visuals or choppy audio during a video call? Do you feel frustrated when facing lag in online gaming? Have you ever been puzzled when discussing a live stream with friends, only to find yourself several seconds behind? For most users, the answer to these questions is likely affirmative, while for most operators, both the root causes and potential workarounds might remain equally obscure and challenging to resolve.

Today's real-time communications (RTC), which constitutes an imperative component across professional, recreational, educational, and health domains—upholding a multitude of applications including video-teleconferencing, internet telephony, online gaming, live streaming/broadcasting, remote healthcare, augmented/virtual reality, among others [1–5]—is far from optimal. Several factors contribute to this suboptimality: *1)* The proliferation of numerous heterogeneous RTC applications in the market, each with distinct network requirements and performance sensitivities, most of which are proprietary and poorly documented to safeguard intellectual property, complicates the creation of universal solutions and strains the adaptability of existing network infrastructures. *2)* Service providers are tasked with maintaining consistent high-quality across a broad spectrum of services, contending not only with the stringent, delay-sensitive nature of RTC traffic but also with bandwidth-hungry yet latency-tolerant applications such as competing video-on-demand applications. *3)* The dynamic and intricate nature of modern networks, especially with the inherent challenges posed by volatile environments such as wireless and mobile networks, introduces additional latency and losses, as well as unexpected disruptions, due to

fluctuating signal quality, varying channel conditions, user mobility, and intermittent connectivity. *4)* The technical insufficiency of traditional network protocols and heuristics often fail to meet the fine-grained, real-time demands of diverse RTC applications, while major service providers are further constrained by rigid legacy architectures and limited flexibility in adapting to evolving communication patterns.

Meanwhile, RTC has gained unprecedented popularity, driven by the widespread adoption of remote working as well as multitudinous lifestyles from consumers, and augmented bandwidth accessibility, alongside substantial growth in network infrastructures from service providers. Against the backdrop of this progression, user preferences and expectations continue to diversify, seeking a seamless, fluid overall communication experience that extends beyond basic audiovisual quality.

In this context, the tension between contemporary suboptimal RTC systems and escalating user demands spurs the development of multifaceted, innovative, and efficacious optimization approaches that transcend the scope of conventional solutions. To this end, the thriving field of artificial intelligence (AI), bolstered by the advent of Large Language Models (LLMs), proffers auspicious and sophisticated technologies, including machine learning (ML), deep learning (DL), reinforcement learning (RL) algorithms, and beyond. Their innate capability of extracting intricate patterns/correlations from massive data enables adaptive decision-making and predictive analytics, thereby facilitating a wide range of networking functionalities such as traffic engineering, condition forecasting, network management, and anomaly detection [6–8].

In this thesis, we delve into Real-time Transport Protocol (RTP)-based RTC and present a suite of four different and unique yet thematically cohesive studies that not only address specific problems but also pursue methodological renovation and practical relevance. We first concentrate on the prevalent and detrimental event of packet loss, framing two imbalanced binary classification tasks and leveraging multiple ML/DL algorithms to either detect the onset of loss bursts or distinguish between sparse and concentrated losses. We subsequently propose a DL framework, namely *DeX* to predict network throughput with emphasis on traffic extremes that are particularly crucial for RTC performance. The former work aggregates packets into time windows and calculates integrated features (e.g., average inter-arrival time), operating in a **coarse** manner that inherently carries incomplete information. With this shortcoming in mind, the latter work employs **fine** granular packet-level information

as features. Nonetheless, both studies remain entrenched in the scenario of a **single** objective, lacking comprehensiveness, synergy, and efficiency. To overcome this, we thirdly introduce *Oh*, a multi-task learning framework empowered by an elaborate architecture and knowledge distillation paradigm, demonstrating that *o*ne model is enoug*h* for **multiple** objectives—efficiently predicting various Quality of Service (QoS) indicators for all concurrent RTP flows in a single pass. Despite thorough analytical and commendable experimental outcomes, aforementioned works "merely" establish **theoretical** validation on the feasibility and effectiveness of ML models. Hence, we finally evaluate the **practical** performance gains achieved through ML technologies. Specifically, we contradict the recent trend of RL for congestion control (CC) in RTC and reveal the pivotal significance of bandwidth estimation (BWE) accuracy, rendering a simple regressor sufficient.

In summary, we gradually refine both the problem formulation and methodological recipe, shedding light on two major concerns in RTC: QoS prediction and CC operation, which provide essential underpinnings for intelligent network management and open the avenue for more adaptive, efficient, and robust RTC systems.

1.1 Thesis Outline

In the following, we briefly outline the thesis structure and enumerate the content of each chapter/section, as illustrated in Figure 1.1. Notably, rather than dedicating a single synthetical chapter to related work, we opt to integrate targeted literature reviews into each individual chapter to more effectively discuss knowledge gaps and contextualize our specific motivations.

Chapter 2 presents the essential background on real-time communications and machine learning, explores their connections, and concludes with a summary of the overarching motivations and objectives.

Chapters 3, 4, 5, and 6 elucidate in detail our work on packet loss detection, throughput estimation, QoS prediction, and bandwidth estimation for congestion control, respectively. In general, each chapter comprises five to six sections: 1) *Background* briefly introduces indispensable context with an extensive literature review on related works to depict the problem and motivate our work. 2) An optional *Observation* section highlights critical findings to further elucidate the

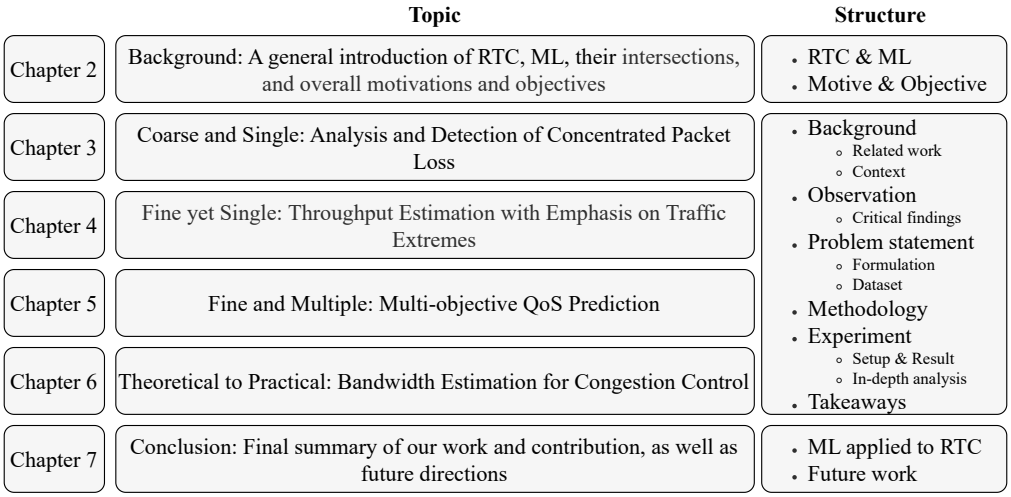


Fig. 1.1 Thesis overview.

underlying crux and impetus. 3) A possible *Problem statement* section formalizes the problem with an introduction of auxiliary information and materials. 4) *Methodology* delineates the technical details of our proposed solutions. 5) *Experiment* describes the setup, presents the results, and offers additional in-depth analyses[1]. 6) The last section of *Takeaways* concludes a chapter with final remarks.

Chapter 7 summarizes the thesis and provides future research directions.

1.2 List of Publications

Herein, we list the papers published or under review during the PhD career, categorizing them into three subsets:

Publications described in this thesis:

- **Packet Loss in Real-Time Communications: Can ML Tame its Unpredictable Nature?**, Tailai Song; Gianluca Perna; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2024) In IEEE Transactions on Network and Service Management (TNSM) [9] (Chapter 3: journal version).

[1]Notably, ablation studies and parametric analyses that may appear in the original papers are omitted here, as this thesis emphasizes conceptual clarity and coherent narrative over exhaustive procedural descriptions of rote works. However, to avoid obfuscation, a dedicated subsection of supplementary material is included to briefly discuss additional experiments.

- **DeX: Deep Learning-based Throughput Prediction for Real-Time Communications with Emphasis on Traffic eXtremes**, Tailai Song; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2024) In Computer Networks (ComNet) [10] (Chapter 4: journal version).
- **One is Enough: Efficient Modelling of RTP Traffic for QoS Predictions in Real Time Communications**, Tailai Song; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2025) In IEEE Transactions on Networking (ToN) [11] (Chapter 5: journal version).
- **Debunking Reinforcement Learning for Bandwidth Estimation in Real-Time Communications: When a Simple Regressor Suffices**, Tailai Song; Michela Meo; (2025) In 21st International Conference on Network and Service Management (CNSM) [12] (Chapter 6: conference version).

Publications related to this thesis:

- **Where Did My Packet Go? Real-Time Prediction of Losses in Networks**, Tailai Song; Dena Markudova; Gianluca Perna; Michela Meo; (2023) In IEEE International Conference on Communications (ICC) [13] (conference version).
- **Throughput Prediction in Real-Time Communications: Spotlight on Traffic Extremes**, Tailai Song; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2024) In 29th IEEE Symposium on Computers and Communications (ISCC) [14] (conference version).
- **Modelling Concurrent RTP Flows for End-to-end Predictions of QoS in Real Time Communications**, Tailai Song; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2024) In 26th IEEE International Symposium on Multimedia (ISM) [15] (conference version).
- **Rethinking Bandwidth Estimation in Real-Time Communications: Do We Really Need Reinforcement Learning?**, Tailai Song; Michela Meo; Review in IEEE Transactions on Networking (ToN) (journal version).

Publications for other topics:

- **High altitude platform stations: the new network energy efficiency enabler in the 6G era**, Tailai Song; David Lopez; Michela Meo; Nicola Piovesan;

Daniela Renga; (2024) In IEEE Wireless Communications and Networking Conference (WCNC) [16] (conference version).

- **BitFormer: Transformer-Based Neural Network for Bitrate Prediction in Real-Time Communications**, Tailai Song; Gianluca Perna; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2024) In 21st IEEE Consumer Communications and Networking Conference (CCNC) [17] (conference version).
- **Towards the Detection of Unobservable Losses in Real-Time Communications**, Tailai Song; Paolo Garza; Michela Meo; Maurizio Matteo Munafò; (2024) In 30th IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN) [18] (conference version).
- **Advancing Cloud-Native Cyber Threat Detection with Graph-Based Feature Engineering**, Tailai Song; Mukharbek Organokov; Lennart Gulikers; Giulio Grassi; Giovanna Carofiglio; Michela Meo; (2025) In 41st IEEE International Conference on Data Engineering (ICDE) [19] (conference version).
- **Do We Really Need Reference-Based Phishing Detection? Unleashing the Power of GNN**, Tailai Song; Pedro Casas; Michela Meo; (2025) In 9th Network Traffic Measurement and Analysis Conference (TMA) [20] (poster).
- **SpecularNet: Towards Reference-free Web Phishing Detection via Hierarchical Graph AutoEncoding**, Tailai Song; Pedro Casas; Michela Meo; Review in ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2026 (conference version).
- **PHISHGRAPH: When HTML Graph Simplicity Outsmarts Deep and Reference-Based Phishing Detectors**, Lucas Torrealba Aravena; Pedro Casas; Tailai Song; Javier Bustos-Jime nez; Ivana Bachmann; Accepted in IEEE/IFIP Network Operations and Management Symposium (NOMS) 2026 (conference version).

For most of the above-mentioned publications, I have made the major contribution as the first author. Typically, our academic workflow involved formulating the problem and conceptualizing the solution for a conference paper, followed by a comprehensive extension featuring an enhanced model and in-depth analyses/experiments for a journal publication. Moreover, my career has been further enriched via both

industrial and academic collaborations, including with globally renowned companies like Cisco and Huawei, as well as national research institutes like the AIT Austrian Institute of Technology. Having benefited from these experiences, I have had invaluable opportunities to learn from/work with exceptional experts on various topics other than RTC, fostering a rich and multifaceted research journey that has greatly strengthened my academic foundation and strategically advanced my long-term career trajectory.

Chapter 2

Background

In this chapter, to establish the background and context before addressing the core topics, we provide a comprehensive overview of RTC and ML. Subsequently, we illustrate our overall motivations and objectives.

2.1 Real-time communications

RTC represents a specific mode of telecommunications, where entities exchange bidirectional or multidirectional information, or transmit unidirectional messages in a real-time manner with negligible, imperceptible, or tolerable latency. RTC underpins a wide array of applications, including video conferencing, cloud gaming, and live streaming, enabling individuals and systems to interact remotely as if they were co-located.

Strictly speaking, the history of RTC can be traced back to the invention of the telephone, which marked the advent of instantaneous, interactive voice transmission over a distance. With the Internet becoming ubiquitous, the form of RTC is evolving drastically. Skype[1], while not the very first, is considered the earliest mature and widely adopted proprietary RTC application, offering reliable voice and video calls over IP networks since its launch in 2003. The true surge in RTC services dates back to early 2020, when the lockdown measures adopted to curb the outbreak of COVID-19 forced millions of people to switch communication medium for both work and

[1] https://support.microsoft.com/en-us/skype. Unfortunately, Skype has stopped its service in May 2025.

leisure all at once, resulting in a global increase in RTC traffic of over 200% [21]. Since then, the repercussions of the pandemic have significantly influenced human society, especially in the education domain where e-learning turned prevalent [22], and in professional settings, where remote work became a global norm [23] (which remains pervasive even today). Nowadays, the importance of RTC continues to escalate, as exemplified by the elevated network demands of high-end online gaming and the advancing frontiers of telemedicine and AR/VR, all accelerated by the emergence of 5G and forthcoming 6G infrastructures.

From a technical perspective, RTC can be classified into two categories: HTTP (Hypertext Transfer Protocol)-based and RTP-based. The former paradigm, exemplified by commercial live-streaming platforms such as Twitch[2], is favored for its statelessness and reliability, making it well suited to scenarios where delay tolerance is permissible. This paradigm, including widely adopted standards such as HTTP Live Streaming (HLS) [24] and Dynamic Adaptive Streaming over HTTP (DASH) [25], implements segment-based Adaptive Bitrate (ABR) streaming over HTTP/TCP and relies on multi-second media chunks and CDN-friendly delivery mechanisms. As a result, it is optimized for scalability, robustness, and playback stability rather than sub-second interactivity. In contrast, RTP, as discussed in the following section, transports packetized, timestamped media and incorporates explicit feedback to enable control mechanisms tailored to tight interactive latency constraints. HTTP streaming addresses fundamentally different objectives and operates at distinct time scales, leading to intrinsic differences in the underlying protocols, operational mechanisms, and performance sensitivities. Consequently, its optimization strategies, control loops, performance measurements, and software solutions are inherently non-transferable to RTP-based services. For example, a deliberate delay for HLS is reserved to buffer and stabilize the video transmission, as a short latency is tolerable in exchange for higher reliability and playback quality—an approach that is unacceptable for RTP-based applications. This thesis focuses on RTP, while HTTP-related topics fall outside our current scope.

[2] https://www.twitch.tv

2.1.1 Real-time Transport Protocol

RTP [26] is the most popular protocol for real-time multimedia content delivery, functioning in various scenarios that require near-instantaneous responsiveness and minimal latency. In this context, the prompt and immediate transmission upon packetization is prioritized over absolute information integrity. For this purpose, RTP packets are created at the application layer and subsequently handed to the transport layer for encapsulation within UDP [27] packets. RTP exploits UDP's innate advantages of being fast, lightweight, and delay-tolerant, extending its property via the additive RTP header and the concept of multimedia streams.

In particular, an RTP header contains the following critical elements:

- **Payload type**: A 7-bit field that identifies the format (encoding) of the RTP payload and dictates its interpretation (decoding) by the application.
- **Sequence number**: A 16-bit value that increments monotonically by one for each RTP packet sent and is used to detect packet loss and restore the correct packet sequence.
- **Timestamp**: A 32-bit number that indicates the sampling instant of the first octet in the RTP packet and is randomly initialized for a session. Each audio packet has a distinct RTP timestamp, whereas multiple video packets associated with the same frame share an identical timestamp.
- **SSRC**: A 32-bit identifier that denotes the synchronization source of an individual RTP stream (flow) during a session.
- **RTP marker**: A single-bit flag used to reflect specific RTP events, e.g., the last packet (boundary) of a video frame.
- **CSRC**: The contributing source list that contains all sources contributing to compose an RTP packet.
- **Header extension**: A 32-bit filed implemented by specific applications for their own interests to add auxiliary information, e.g., absolute sending time.

Due to the encryption of packet payload, the RTP header entities, in tandem with other protocol headers, not only enable various functionalities such as synchronizing

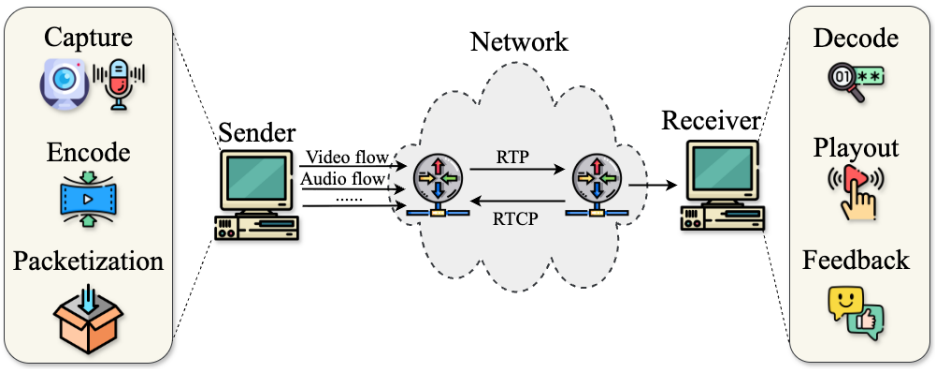


Fig. 2.1 The workflow of the RTP-based RTC system.

video and audio for RTC applications, but also provide rich information that we have relied on heavily for the work in this thesis, including traffic volume computation, flow identification, packet loss detection, and more. However, the growing prevalence and strictness of packet encryption, extending even to headers [28, 29], may further hamper the problems, representing a valuable future research direction.

Figure 2.1 portrays the general workflow of RTP applications. The sender first captures multimedia data, which is then encoded using a specific codec (e.g., H.264 for video). Afterwards, the sender packetizes processed data into standardized RTP format, transmitting traffic altogether while splitting it into individual flows for different payload types such as video, audio, and probing streams. A single flow is uniquely determined by a six-element tuple composed of the source and destination IP addresses and ports, SSRC, and payload type. Sequence numbers are maintained within each individual flow but are uncorrelated among flows. Traffic is delivered following standard network paradigms, where a centralized server may exist to manipulate/recombine multiple streams into one and adapt quality. Upon reception, the receiver decodes packets after any necessary buffering and reordering, presenting the reconstructed multimedia content to the user. Additionally, to optimize performance and manage the session, the receiver collects certain statistics such as packet loss and jitter metrics as feedback, sending messages in the receiver report (RR)[3] through Real-time Transport Control Protocol (RTCP) [30] to steer the sender operations including encoding strategy, bitrate selection, etc.

[3]The sender also actively participates the control loop, transmitting sender reports (SR) encompassing information such as NTP timestamp and packet count for synchronization, monitoring, and analysis purposes.

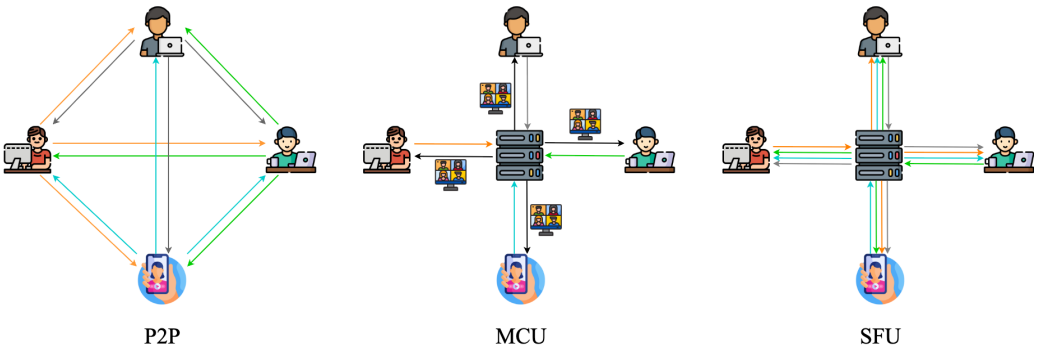


Fig. 2.2 VCA system topology.

Video-conferencing application

One of the most prominent and representative services based on RTP is video-conferencing applications (VCAs), which are indispensable for telecommuting and e-learning. In this thesis, our work extensively engages with VCA traffic, focusing primarily on optimization and configuration in the context of video-call scenarios.

The high-level architecture of VCAs adheres to that of the aforementioned RTP-based systems, with three different types of connection topologies (as depicted in Figure 2.2):

- **Peer-to-Peer (P2P)** enables participants to connect directly to each other without an intermediary server for traffic relay. In this mesh topology, each participant must send a separate copy of their media stream to every other participant. While P2P is simple and incurs minimal server-side cost, its uplink bandwidth and CPU usage scale poorly with the number of participants: for N participants, each sender transmits $N-1$ streams, leading to exponential growth in network and processing load. Consequently, P2P is generally feasible only for small groups (typically 2 to 4 participants) and can severely impact QoE on real networks and resource-constrained devices, particularly mobile clients.

- **Multipoint Control Unit (MCU)** acts as a centralized media center/server, accepting all incoming flows, decoding and recomposing media content, and generating a single stream per recipient. This approach reduces client-side complexity and bandwidth requirements, as each participant downloads only one consolidated stream (e.g., a grid layout of all participants or a composite

speaker-plus-thumbnails view). However, the MCU introduces significant server-side computational overhead, requiring substantial CPU/GPU resources for real-time mixing, and adds processing latency that can affect interactivity. While theoretically efficient for bandwidth-limited clients, the MCU scales poorly for large meetings due to its centralized processing cost and potential bottlenecks at the server.

- **Selective Forwarding Unit (SFU)** also functions as a media center but simply forwards streams without recoding them. Each participant uploads a single stream, and the SFU selectively routes streams to relevant recipients, potentially leveraging simulcast or scalable video coding (SVC) to adapt quality per receiver. SFUs thus shift the processing burden from clients to a moderate server footprint, reduce overall latency compared with MCUs, and support interactive meetings at larger scales. However, clients must handle the mixing of multiple incoming streams, which can impact CPU usage and battery consumption on mobile devices. SFUs also support operational flexibility, such as layout control, TURN fallback, and cascading deployments for very large conferences.

For multi-party calls, the latter two schemes are typically employed. MCUs minimize client complexity but incur high server-side costs and latency, while SFUs distribute processing to clients, reducing server load and latency but requiring more capable receivers. Overall, system design/topology selection is dictated by practical trade-offs such as scalability (number of participants), computational complexity (client vs. server CPU/GPU), resource budget (uplink/downlink bandwidth), latency sensitivity, device constraints (battery/thermal impact), and operational requirements (recording, compliance, layout management). Beyond determining system topology, these trade-offs also raise critical ML-wise concerns for modeling (e.g., feature selection and availability), inference (e.g., computational efficiency and responsiveness), and evaluation (e.g., coarse- versus fine-grained prediction targets). Although this thesis mainly focuses on end-to-end optimization algorithms that are topology-agnostic, we believe that ML development should factor in/could benefit from topology-specific information—an aspect we partially address in this thesis and plan to explore in future works.

Notably, under the hood of most VCAs, RTP typically dose not operate independently; rather, it cooperates with other protocols to utilize off-the-shelf technolo-

gies [1]. For instance, TLS [31] and DTLS [32] are employed to build encrypted channels when confidentiality is needed. Moreover, to establish a stream session, RTP-based applications often rely on STUN [33] protocol for NAT detection and TURN [34] to relay the traffic through a server, with these control messages often multiplexed alongside RTP over the same UDP flow.

Native VCAs generally have their own implementations, whereas web browsers and mobile platforms (e.g., Android) universally hinge on the acclaimed standard—WebRTC [35][4], which is an open-source framework built atop RTP and a set of high-level standard APIs for real-time media communication via web. Although webRTC proffers additional features such as coordinating various protocols, implementing a standardized set of codecs, and employing a default congestion control algorithm (Google Congestion Control), the underlying multimedia content delivery is still carried out by RTP. In this thesis, besides the traffic sourced from webRTC applications, our last work also focuses on congestion control in the webRTC framework.

Other variants

To cater to diverse requirements and operational contexts, RTP exists in certain variants, is frequently customized by VCAs, and is deployed across a range of applications.

Secure RTP (SRTP) [36], used by applications such as WebRTC, is an RTP variant introduced to ensure confidentiality by encrypting the payload while leaving headers unchanged. While most RTC applications use RTP for streaming, many obfuscate their protocol implementation [1, 37, 38]. For example, Zoom[5] leverages an unknown, bespoke wrapper on top of RTP, and Microsoft Teams[6] uses a non-standard yet documented method to encapsulate RTP. Furthermore, a small portion of RTP traffic generated by applications like Google Meet[7] exhibits peculiar customized mechanism, including dynamic payload type within the same flow and irregular gaps in sequence numbers between video frames [18]. Additionally, RTP also plays critical roles among online (cloud) gaming, live streaming, IoT applications, and

[4] https://webrtc.org/
[5] https://www.zoom.com
[6] https://www.microsoft.com/en-us/microsoft-teams
[7] https://meet.google.com/

more, which tend to revamp RTP to a certain extend for specific purposes, perturbing ordinary traffic patterns (domain shift) and thereby impeding ML algorithms. For instance, audio and video may be mixed within the same flow for cloud gaming traffic [2].

In our work, we are compelled to exclude portions of data in a few cases due to factors described above, while opting and intending to consider various scenarios to validate model generalizability in most cases. Indeed, our solutions proposed in this thesis demonstrate varying degrees of generalizability, achieving acceptable performance mostly and subpar outcomes occasionally under domain shift. However, we still envisage the versatility and transferability of our models, which can be readily extended to different RTC applications, ascribed not only to the consistent underlying usage of RTP but also to modern, mature technologies including transfer learning, incremental learning, domain adaptation, and model fine-tuning.

Quality of Service & Quality of Experience

Our work in this thesis heavily involves the monitoring, quantification, prediction, and improvement of RTC-related QoS and QoE, wherein QoS embodies a group of key objective metrics used to evaluate the level of performance for an RTC session, and QoE represents subjective, perceived indicators of network/application quality from a user perspective.

One the one hand, QoS is essential for assessing and guaranteeing the smoothness and consistency of time-sensitive applications. Optimizing QoS metrics is an explicit and straightforward way to quantitatively enhance network performance. Some key QoS metrics include:

- **Bitrate**: The volume of data transmitted between a sender and a receiver for a single RTP flow. The total bitrate of all co-existing flows over a link/node/end-device is referred to as the throughput, in which the useful portion, such as packets carrying actual multimedia content (excluding functional traffic like error correction packets) is termed goodput.
- **Packet loss**: The proportion/count of dropped packets that fail to reach their final destination. It can be identified and quantified according to gaps in sequence numbers.

- **Latency**: The time needed for a packet to travel from the sender to the receiver. Different pertinent terminologies exist. For example, *1)* one-way delay, generally considered equivalent to latency, is roughly approximated as half of the round-trip time (RTT), i.e., the time for a packet to transmit from the sender to the receiver and back, and *2)* queuing delay refers to the time a packet spends waiting at network nodes before processing, i.e., latency excluding the physical path traversal time.

- **Jitter**: The variation of packet arrival/latency. Notably, the common definition of jitter is the variance of inter-arrival times, which is, however, imprecise and less informative. Instead, we adhere to [39], calculating packet jitters based on the clock rate (packet generation frequency), which can either be acquired from application logs or inferred via the difference in RTP timestamps between consecutive packets.

- **Bandwidth**: The available capacity of the network's bottleneck link/node. It determines and throttles the maximum amount of data that can be transmitted over the network connection.

Most QoS measurements can be obtained directly and accurately from packet information, but some are inherently challenging to collect. For instance, the RTT cannot be accessed for each RTP packet (as there is no acknowledgment like in TCP) but can be derived with a relatively longer granularity through RTCP reports. The absolute one-way delay cannot be measured precisely, but can be replaced by relative delay (or delay variation/jitter) or roughly estimated using the difference between sending and arrival times (which are certainly not synchronized and affected by clock drift). In our work, we focus on accessible QoS metrics, while also elaborating on unavailable ones.

On the other hand, the shift from QoS to QoE represents a major concern in evaluating and optimizing RTC services, because while QoS and QoE are fundamentally correlated, a proper-managed network with commendable QoS indicators does not necessarily reflect a high-quality user experience [40, 41]. However, it is arduous to objectively measure QoE, and therefore, researchers/engineers either collect user feedback after a session or employ quantitative metrics that reflect human perception, such as mean opinion score (MOS) and video quality metrics (e.g., PSNR and VMAF). In this thesis, we primarily focus on QoS with the ultimate goal of

enhancing QoE, but quantifying improvements in user experience remains difficult, forming a worth-pursuing future direction.

Furthermore, the relationship between QoS and QoE has been extensively studied and formalized in RTC. Numerous prior works have proposed foundational models that mathematically relate these two entities. Classical telephony and VoIP QoE frameworks employ the ITU-T E-model [42] to explicitly map delay, packet loss, and codec-related impairments to an intermediate R-factor, which is subsequently translated into MOS via a nonlinear mapping function. Another prominent example is the IQX hypothesis [43], which posits that QoE degrades exponentially with increasing QoS impairment:

$$QoE(X) = \alpha e^{-\beta X} + \gamma, \tag{2.1}$$

where X denotes a dominant network impairment metric (e.g., packet loss rate, latency, or jitter), α and β are content- and service-dependent parameters, and γ represents a lower asymptotic bound on perceived quality. This captures the empirically observed phenomenon that small degradations at high quality levels incur disproportionately large perceptual penalties, whereas additional impairments near poor quality exhibit diminishing impact. Beyond assuming a single dominant impairment, recent works adopt statistical or ML-based methods to quantitatively map QoS to QoE [40, 44, 41]. These approaches provide analytical tools that enable predictive QoE provisioning when supported by accurate QoS anticipation. Another critical motivation driving our focus on QoS prediction arises from the Provisioning–Delivery Hysteresis (QoE-PDH) [45], which identifies asymmetric QoE effects between controlled and uncontrolled degradations. Specifically, QoE-PDH highlights that deliberate (controlled) reductions in provisioning (e.g., adaptive bitrate throttling or resolution downscaling) tend to have a milder impact on QoE than delivery-side (uncontrolled) impairments such as network congestion. This asymmetry explains why proactive adaptation mechanisms are often preferred over reactive recovery: some QoS degradations are tolerable when managed explicitly, while others necessitate preemptive control or admission decisions. Taken together, QoE is not merely correlated with QoS but is governed by structured, nonlinear relationships shaped by system behavior, protocol dynamics, and human perception. In this context, packet-level effects such as delay, jitter, and loss enter a quantitative QoE pipeline, propagating through system components before culminating in sub-

jective quality assessments. This perspective also underscores why ML can play a complementary role by unearthing low-level network statistics to capture complex interactions through data-driven modelling, thereby augmenting well-established QoS-QoE theory.

2.1.2 Traffic & Dataset

The dataset represents a crucial component for training and evaluating ML models. In this thesis, the first three works employ similar datasets collected in-house[8], and the last work relies on publicly available datasets.

The traffic associated with the datasets for loss detection, throughput estimation, and QoS prediction was collected either during real video-calls at edge nodes or through general RTP traces from border routers. First, we have conducted plenty of video conferences under various conditions in terms of number of participants, network connectivity (WiFi, mobile, Ethernet), collection periods/locations, and RTC applications [46, 1]. The two major applications are Jitsi Meet[9] and Webex[10], which correspond to a total of roughly 70 hours of traffic. Other involved applications are Microsoft Teams, Zoom, Google Meet, Skype, and BigBlueButton[11], accounting for approximately 40 hours of traffic. During each call, we capture the traffic on the client side and stored in *pcap* format, focusing exclusively on incoming streams with the objective to investigate RTP flows sourcing from each sender, traversing through the network, and being affected by all intermediate links and nodes. Second, we also have utilized the TCP STatistic and Analysis Tool (Tstat) [47][12] to gather general, in-bound, application-agnostic RTP traffic from a campus router, anonymized using the Cryptography-based Prefix-preserving Anonymization (Crypto-PAn) algorithm [48]. The vantage point at the exit node of the university campus offers unique benefits for large-scale traffic acquisition due to two key factors: *1)* the campus's output link boasts a broad bandwidth of 10 Gb/s, with mean daily usage of roughly 3.5 Gb/s, enabling rapid and substantial data collection, and *2)* the collection is performed in

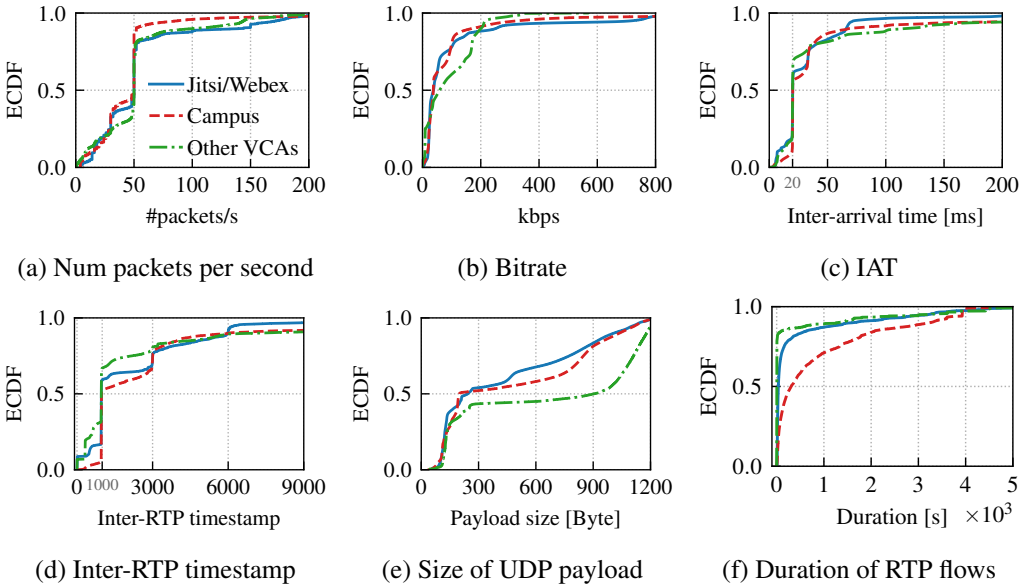
[8] Here, we provide a general overview of the traffic referenced in Chapters 3, 4, and 5. However, throughout the thesis, different portions or variations of these datasets may be used under different names to accommodate the specific requirements of each problem.

[9] https://jitsi.org/jitsi-meet/, an open-source video-call platform.

[10] https://www.webex.com/, a commercial native application.

[11] https://www.webhostingzone.org, a web conferencing platform with open-source learning management systems.

[12] http://tstat.polito.it/



(a) Num packets per second (b) Bitrate (c) IAT

(d) Inter-RTP timestamp (e) Size of UDP payload (f) Duration of RTP flows

Fig. 2.3 Flow-wise traffic patterns.

an automated and continuous manner without manual intervention. Similarly, the traffic was dumped and stored in *pcap* files. However, given the enormous amount of traffic frequenting the university per day, it is infeasible and unneeded to consider/process all of them, and thereby, we opt to use a random portion. Additionally, as different tasks require distinct features, the data preprocessing steps are detailed in each respective chapter.

Moreover, to justify the comprehensiveness, diversity, and representativeness of these datasets, we provide a detailed characterization through six Estimated Cumulative Distribution Function (ECDF) plots in Figure 2.3, each corresponding to a networking property on a flow-basis for all three sources of traffic. In general, all datasets exhibit distinct variations both among and within datasets to a certain extend. For instance, most traffic from other VCAs was collected over short durations, as shown in Figure 2.3f, and its packet (media) type is relative uniform, with either small or large UDP payloads, evident from the plateau in Figure 2.3e, which potentially originates from the comparatively simplified experimental environment, with minimal participants and single streams. In contrast, notable statistical similarities appear across all traffic sources, exemplified by the prominent rises at around 50 packets/s, an IAT of 20 ms, and a difference of 1000 between RTP timestamps in Figures 2.3a, 2.3c, and 2.3d, respectively. These phenomena can be attributed to the presence of audio flows packetized in accordance to the RTP specification [39]. Given the

most common audio packetization granularity at 20 ms and the widely used Opus codec with an operation frequency of 48 kHz [49, 50], the difference in RTP timestamp between two consecutive audio packets is 960 (i.e., 48kHz $\times$ 20ms $\approx$ 1000). Taken together, these observations, alongside the variety of traffic patterns, collection conditions, and other sudden increases signifying additional payload (media) types, validate the dataset thoroughness and confirm the relevance in capturing diverse aspects of RTC traffic.

Noteworthily, regarding the traffic and datasets employed in the last work of bandwidth estimation for congestion control, we leverage publicly available traces recorded either in real-world scenarios, emulations, or at testbed nodes. Details are deferred to Chapter 6. Additionally, each chapter provides references to the corresponding data and code, where available. Note that we are unable to share raw materials (i.e., *pcap* files) due to confidentiality and privacy constraints.

2.2 Machine learning

The term ML was coined in 1959 in the context of computer gaming [51]. Since then, the field has undergone an up-and-down trajectory, spanning from early-stage statistical algorithms, through the setbacks of the "AI winter" caused by limited learning effectiveness, to the resurgence brought by the rediscovery of backpropagation, the rise of support vector machines (SVMs) and kernel methods, and culminating in the bourgeoning advancement of DL models [52]. Under the contemporary definition, ML is a field of study that focuses on the design and development of algorithms capable of learning patterns from data and making decisions or predictions without being explicitly instructed/programmed [51].

Today's AI has transitioned from a conceptual vision to a tangible reality, especially with the emergence and rapid evolution of generative AI and LLMs. Amidst the trend, ML, along with its most important subfield of DL, plays a pivotal role, leveraging statistical, algorithmic, or blackboxified approaches to contribute radically across modern society and to reshape the methodological scopes of a broad spectrum of domains [53–55], in which the computer network field has likewise been inexorably and profoundly influenced.

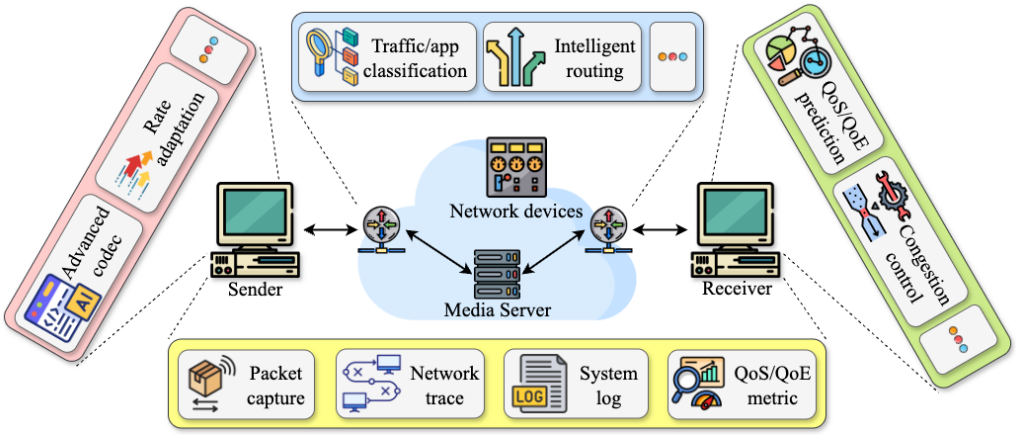


Fig. 2.4 ML-based applications in RTC.

2.2.1 ML for networking

ML-based algorithms have been widely adopted in all stacks of networking systems and applications [56, 7], proffering unprecedented advantages in areas such as network control [57, 58], traffic engineering [59, 60], traffic classification [61, 62], anomaly detection [63, 64], resource allocation [65, 66], and more. By leveraging data-driven insights, ML/DL/RL models can learn underlying patterns, discern intricate nuances, and uncover interweaving correlations, thus enabling intelligent decision-making, adaptive responses to network dynamics, and enhanced performance beyond the reach of traditional heuristic approaches.

ML in RTC

ML algorithms are characterized by their automated, intelligent decision-making and data-driven nature, in which the former facilitates RTC to adapt swiftly and adeptly to network fluctuations in real-time, and the latter aligns inherently with the abundant data sourced from packet captures, network traces, system logs, and QoS/QoE metrics gathered during or after RTC sessions, as outlined in Figure 2.4. For instance, packet streams can be captured using Tshark[13] or Wireshark[14] for online or post processing/analysis. WebRTC, supported by modern browsers, provides logging for troubleshooting and debugging[15], and the meeting statistics of Zoom allow users

[13] https://tshark.dev/
[14] https://www.wireshark.org/
[15] https://webrtc.github.io/webrtc-org/native-code/logging/

to view diagnostic information[16]. In general, packet captures are dumped/collected packets that enables traffic replay and carry fundamental information for various purposes. In the ML domain, traffic packets can be modeled/represented in the form of aggregated features[9, 67], sequences [10, 68], graphs [69], images [70], etc., which are then fed into downstream models for different tasks. The terminologies of trace, log, and metric stem from the concept of observability [71], which offers compact, transparent, and comprehensive descriptions for the sake of monitoring, management, and root-cause analysis. All of these elements can serve as either predictive targets or input features for ML technologies in RTC [17, 15, 72–74].

Figure 2.4 also showcases multiple specific RTC functions empowered by ML, including neural codec [75–77], adaptive transmission rate control [78, 79], classifications of traffic types and RTC applications [67, 80, 81], intelligent routing algorithms [82, 83], predictions of QoS or QoE [15, 84, 85], bandwidth estimation for congestion control [86–88], and more. These examples are not exhaustive, and other scenarios include ML applied to security [89], scaling [90], error correction [91], speech [92], cloud computing [93] aspects within RTC. Each of aforementioned ML-enabled RTC functions operates either independently or collaboratively at different vantage points, actively optimizing sender-side traffic flows, holistically managing network configurations through network devices (e.g., an orchestrator) or SDN paradigms, and passively estimating traffic statistics on receiver-sides. In RTC, ML normally constitutes a component of an integrated framework, where the model is technically decoupled from other system elements, i.e., the model simply consumes incoming features and produces a prediction without taking care of how the framework uses the outcome. This approach leverages off-the-shelf tools and circumvents the complexity of constructing everything from scratch. A typical workflow of an ML-based RTC system could be: the receiver predicts certain metrics (e.g., bandwidth availability) based on received traffic and feeds back the information to the sender, which subsequently adjusts operational parameters (e.g., modifying bitrate) in response to the feedback that reflects network conditions/variations.

[16] https://support.zoom.com/hc/zh/article?id=zm_kb&sysparm_article=KB0070504

Challenge

Unlike well-established domains such as natural language processing (NLP) and computer vision (CV), where most problem formulations and pipelines have been consolidated and standardized[17], ML in RTC forms multidisciplinary problems, and the associated challenges are further complicated and manifold:

1. **Domain knowledge**: Effectively integrating ML into RTC systems demands interdisciplinary proficiency and deep expertise in the networking and computer systems, but the organic synthesis of domain knowledge and the design of ML algorithms remains problematic.

2. **Generalizability**: Models trained on historical network conditions or traffic patterns often struggle to generalize to unseen environments, limiting their robustness and practical value, especially in today's ever-evolving networking landscape.

3. **Adaptability**: Models must adapt promptly to network oscillations, but such adaptability often compromises generalizability by overfitting to transient patterns.

4. **Efficiency**: Achieving low-latency inference and minimal computational overhead is critical for real-time applications, yet ML models tend to be resource-intensive by nature.

5. **Real-world deployment**: Bridging the gap between simulated environments and real-world RTC systems poses significant engineering hurdles, including integration, testing, and backward compatibility.

6. **Performance trade-offs**: Optimizing for one target metric (e.g., delay) often comes at the expense of others (e.g., throughput or fairness), making multi-objective optimization a persistent dilemma.

7. **Device constraints**: Many RTC applications run on end-devices, edge equipments, or network nodes with limited CPU, memory, and power, restricting the complexity and size of deployable ML models.

[17]These fields feature widely recognized problems, datasets, frameworks, and validated theories, and the primary objective is typically to surpass state-of-the-art performance.

8. **Scarcity of public datasets and benchmarks**: The lack of standardized, publicly available datasets and benchmarks hampers reproducibility, fair evaluation, and progress in ML for RTC research.

The design of ML-based RTC systems entails careful consideration of many, if not all, of these challenges. In our work, we have sought and intended to combat multiple issues but had to temporarily put aside some aspects (e.g., real-world deployment) because of the problem scope and limited resources.

Scope of this work

Herein, we briefly introduce/formulate each work from an ML perspective (as outlined in Figure 2.5). In general, the first three studies validate the feasibility of employing ML/DL algorithms to improve prediction performance, while the final work evaluate the tangible network enhancements achievable through ML. Specifically:

1. The first work (Chapter 3) formulates a supervised binary classification problem, wherein feature engineering is employed to capture traffic patterns and predict the onset of lossy events. A time bin-based approach is adopted, whereby packets within historical bins are aggregated to compute a set of statistical features. We then leverage these features to classify whether the subsequent time bin signifies impending losses. As feature condensing and compression inherently incur information loss, this approach is consequently deemed coarse. For performance evaluation, data imbalance is explicitly taken into account, with recall used to assess performance for each class individually.

2. The second work (Chapter 4) addresses a supervised learning regression problem, aiming to predict RTC throughput with a particular emphasis on traffic extremes. In contrast to, and as an improvement over, the previous work, we employ fine-grained packet-level features (i.e., header information from received packets) to estimate throughput in the forthcoming time window. Despite its RTC-specific focus, the problem conforms to a standard regression scheme, and thus common regression evaluation metrics are adopted.

3. The third work (Chapter 5) invovles a multi-task learning problem that enables unified QoS prediction for all coexisting RTP flows in one shot. We continue

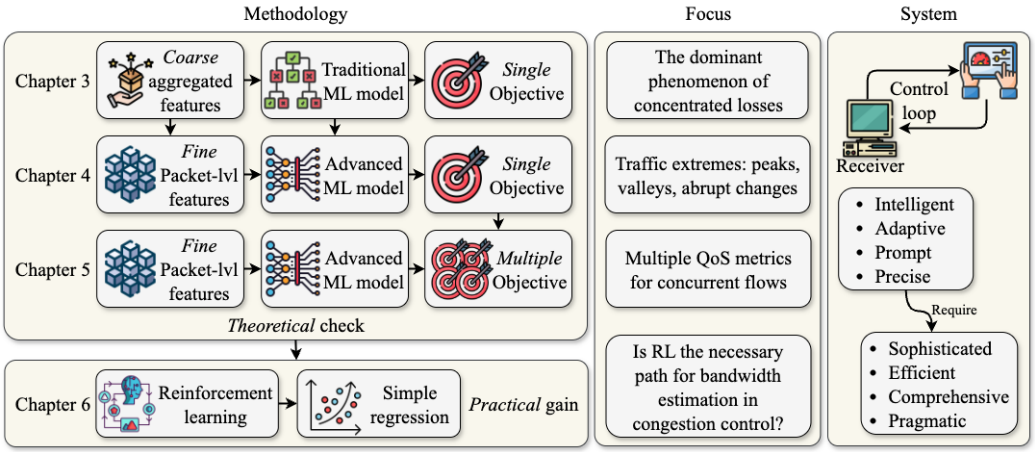


Fig. 2.5 Overview of motivations and objectives.

to leverage historical packet-level features to predict multiple, distinct per-flow QoS metrics in the near-future time window. Given the heterogeneous learning objectives across tasks, different evaluation metrics are adopted for each task individually.

4. The fourth work (Chapter 6) moves beyond a rudimentary feasibility check and focuses on assessing the practical gains achieved via ML techniques. We consider a congestion control scenario, in which common traffic features are used to regress the current bandwidth availability—a simple approach that nevertheless outperforms existing RL-based algorithms. Beyond performance indicators for the ML task itself, we evaluate domain-specific QoE metrics to capture the impact of ML models.

Additionally, three critical factors are taken into account: *1)* Vantage point—our works assume an end-to-end deployment operating at the client side, while preserving the potential for extension toward in-network optimization; *2)* RTC constraints—the real-time nature of RTC imposes numerous limitations, including the latency of feature construction and model inference, hardware resource budgets, the achievable prediction horizon, and more; *3)* Data split—rather than employing conventional stratified splits for training, validation, and testing, traffic is deliberately partitioned by identity and source, e.g., packets from a given flow appear in only a single dataset.

2.3 Overall motive & objective

Remarkably, ML techniques can indeed address certain issues and boost application performance, thanks to their natural compatibility with RTC scenarios, where abundant data are available and the management for volatile network conditions can be beneficial from intelligent and adaptive decision-making. However, the current scale and landscape of RTC impose various additional complicated constraints, exceeding the capabilities of present solutions and entailing more sophisticated, tailored, and efficient workarounds. In light of these considerations and informed by our comprehensive literature review[18], we elucidate three overall motives and their corresponding objectives (as illustrated in Figure 2.5):

1. From a methodological point of view, existing approaches manifest different degrees of defectiveness, including model insufficiency and deficient considerations. Throughout our work, we aim to gradually improve the technologies, ranging from coarse to fine features, traditional to advanced models, single to multiple objectives, and theoretical to practical investigations.

2. Each task highlights a specific target, unearthing and focusing on a crucial problem, i.e., the loss burst (Chapter 3), traffic extremes (Chapter 4), multiple per-flow QoS metrics (Chapter 5), and the necessity of RL (Chapter 6), which prior works either underperformed or overlooked.

3. Our work mainly concentrates on the receiver side, providing predictive signals and QoS feedback for various functionalities. From the perspective of the holistic RTC system, existing solutions are inadequate in various ways, e.g., heuristics are relatively naive, while complex methods are capable yet inefficient. Catering to real-time demands and to realize modern preemptive, swift, and accurate control loops, our ultimate objective is to develop sophisticated and comprehensive ML models that are efficient and pragmatic at the same time.

Specifically, we aim to address the following research questions:

- **RQ1**: Can ML algorithms be used to forecast the occurrence of packet loss? (Chapter 3)

[18] Detailed introductions of related works are listed in their respective chapters.

- **RQ2**: Can sporadic and sustained loss events be effectively distinguished? (Chapter 3)
- **RQ3**: Do RTP packets inherently carry the critical information required for performance enhancement? (Chapters 4 & 5)
- **RQ4**: Is it possible to accurately identify and predict extreme conditions in RTC traffic? (Chapter 4)
- **RQ5**: How can QoS prediction be performed in an efficient and comprehensive manner? (Chapter 5)
- **RQ6**: Is reinforcement learning truly necessary for bandwidth estimation in RTC? (Chapter 6)

In each of the following chapters, we present a brief introduction of the problem background and elaborate on relevant literature to further underscore existing challenges and clarify our specific motivations.

Chapter 3

Coarse and Single: Analysis and Detection of Concentrated Packet Loss

In this chapter, we present our work: *Packet Loss in Real-Time Communications: Can ML Tame its Unpredictable Nature?* [9]. Packet loss is an inevitable and detrimental networking event, occurring when one or more packets fail to reach the destination. Unlike TCP, which identifies and recovers lost packets via acknowledgments and retransmissions, RTP cannot guarantee packet delivery as it runs over UDP, a protocol that prioritizes minimal latency over loss detection and recovery.

Building on the background presented in Chapter 2, this chapter examines the potential of ML models for loss prediction. Specifically, we observe the trending phenomenon of concentrated losses, which signifies that predicting/identifying the onset of continuous losses could potentially avoid incoming losses in advance. To this end, we construct three datasets, training and tuning ML models on one while evaluating the others. We frame two similar problems (directions), and our results demonstrate the prospect of packet loss forecasting albeit several challenges. In the following, Section 3.1 introduces the background with related works, followed by Section 3.2, which highlights a critical observation to further illustrate the underlying motives. Section 3.3 formulates the problem. Section 3.4 describes the methodology. Section 3.5 reports the experimental results. Section 3.6 concludes with key takeaways.

3.1 Background

Of noteworthy concern, packet loss constitutes a primary detriment to satisfactory performance in RTC [94, 40, 95], engendering adverse repercussions, including QoS declines such as network throughput reduction, latency growth, etc., and QoE impairments such as audio inconsistency, video frame distortions, and complete service disruption in severe cases [96–98]. While several techniques have been proposed in response to this challenge, e.g., congestion control that adapts bitrate [99], error correction that compensates missing information [100], and packet loss concealment that reconstructs lost signals from received packets [101], they incur additional overhead and remain passive countermeasures, reacting after the occurrence of packet loss and thereby retaining inherent limitations. In tandem with the increasingly complicated networking landscape such as wireless environments, it becomes pressing to develop innovative and effective technologies to mitigate packet loss.

At the application-level, packet loss can be inferred from inconsistencies in sequence numbers or by comparing the number of sending packets reported in sender reports (SRs) with those actually received, which is, however, only accessible to endpoint applications themselves. For holistic system-level and full-stack observability, conventional approaches of active/passive network loss measurement are able to estimate the loss rate, through probing packets or Management Information Base (MIB) based on Simple Network Management Protocol (SNMP) [102, 103]. Nevertheless, they often suffer from inadequate estimation accuracy in numerous cases [104], fail to uncover actual losses within particular RTP flows, and impose extra resource overhead due to, for example, additional auxiliary packets. [105] has introduced a novel technology, inserting measurement marks into the user data field to measure losses for video streaming, though its applicability is still unverified for existing RTC applications. Moreover, while [106, 107] have performed extensive evaluation campaigns of videoconferencing performance, and [108, 72, 40] have tried to convert QoS metrics to QoE, they coincidentally presuppose the availability of packet loss knowledge. Authors in [73, 109–112] have developed various advanced technologies to estimate QoE for common VCAs, but they all have cast a veil over packet loss measurement, explicitly acknowledging such a limitation in their works. Overall, packet loss assessment in RTC remains relatively understudied. On top of that, these methods are still hindsight for post-analysis, lacking proactivity and timeliness.

In this context, packet loss prediction emerges as a plausible candidate to enable proactive and timely network management. By forecasting network losses beforehand, it becomes possible to react preemptively to the occurrence, thereby optimizing performance and alleviating ramification in advance. However, only a limited number of studies have explored packet loss prediction. For instance, both [113] and [114] built mathematical models to predict packet loss, where the former work aimed to enhance video codec in wireless networks and relied on simulated data to determine parameters via linear regression, and the latter one investigated the correlation between delay and loss in audio streams, employing delay variation along with short- and long-term trends to construct a loss predictor. Similarly, authors from [115, 116] attempted short-term loss rate forecasting, formulating time series problems and developing simple models including an artificial neural network (ANN) with particle swarm optimization and a hidden Markov model based on loss distribution. Beyond RTC, [117] proposed a lightweight ML model to estimate retransmissions a-posteriori on a per-flow basis in bulky TCP traffic, and [118] adopted a Decision Tree and Logistic Regression models to predict packet losses ("high" or "low") for Call Admission Control systems. These works manifest a fundamental limitation in naively formulating the problem with simplistic solutions and without deeper exploration. To the best of our knowledge, our work represents a pioneering effort that extensively studies packet loss in RTC and proposes meticulous ML frameworks for its fine-grained prediction in a unique and systematic way. Our proposed solution is envisioned to function as a software module in network devices, such as media servers, to establish a comprehensive, ML-based, RTC-aware, and proactive traffic management and monitoring system. The system provides application-level observability at the network control plane, empowering efficient and informed decision-making, and incorporates a feedback mechanism to notify deteriorating network conditions promptly. It consists of several offline-trained classifiers that predict in real-time various network conditions, one of which is the possible packet loss in the near future. Subsequently, a controller (e.g., an SDN orchestrator) can apply specific network management techniques to alleviate the worsening conditions, such as rerouting flows along alternative paths or allocating additional bandwidth to them.

Coarse and Single

The term "coarse" reflects our adoption of aggregated traffic characteristics, wherein RTP packets are condensed within time windows, and this feature compression inevitably coarsens the available information. The reason behind is twofold: 1) The formidable challenge (predictability of packet loss) of the problem stems from network complexity, which already constrains the upper performance bound of data-driven approaches; hence, we refer to such a typical and safe feature-engineering strategy. 2) Part of our work involves enhancing an open-source software that proffers off-the-shelf functionality for feature construction. The term "single" denotes the single target of packet loss, a topic deserving thorough investigation during the early research stage, as our objective lies also in traffic monitoring/analysis rather than pure ML model development.

3.2 Observation

3.2.1 Preliminary

Dataset

In order to parse packet captures, we utilize a configurable networking tool Retina [37][1], which aggregates time-series packets into consecutive time bins (500 ms in our case) arranged in chronological order on a per-flow basis. In each bin, a set of 73 traffic statistics[2] is calculated to effectively encapsulate the patterns of aggregated packets. A simple example is illustrated in Figure 3.1.

We employ Retina to generate three distinct datasets, namely "Supervised", "Campus2020", and "Campus2023", sourced from the traffic described in Section 2.1.2. The "Supervised" dataset is from the two major RTC applications, Jitsi Meet and Webex, and is named as it was intentionally collected during multiple real video teleconferences in a supervised manner, where rich logs are available. The "Campus" datasets originate from the border router using Tstat in different periods as suggested

[1] https://github.com/gianlucaperna/Retina
[2] Refer to https://smartdata.polito.it/rtc-classification/ for more details.

Table 3.1 Dataset overview.

Name	File Size [GB]	Collection Period	Vantage Point
Supervised	66.54	2020-2021	Edge-node
Campus2020	68.33	2020-2021	Campus router
Campus2023	87.34	2023	Campus router

by the suffixes. Table 3.1 provides a brief summary, including the total size of *pcap* files, collection period, and vantage point. Our data and code are publicly available[3].

Loss definition

According to the presence or absence of packet loss in a bin, we define: a **lossless time bin** is a bin devoid of any losses, while a **lossy time bin** denotes a bin in which at least one loss has occurred, as portrayed in Figure 3.1. More importantly, we also define two types of lossy time bins (as in Figure 3.2):

- **Sparse loss** indicates a lossy time bin, where there are no losses in its preceding and succeeding 5-s intervals, i.e., the 10 bins in the past and future of a sparse loss are lossless.
- **Concentrated loss** denotes a loss burst over a certain duration and is defined as a lossy time bin with at least one additional loss in its neighborhood of 5 s. That is to say, there is at least one lossy time bin in the 5 s in the past or future of a concentrated loss, and the additional lossy bin is, therefore, also a concentrated loss. Multiple consecutive concentrated losses form a group, wherein the leading lossy bin is deemed the starting-point loss, while the following ones are considered group members. Notably, there is no loss neither in the 5 s before the starting point nor in the 5 s after the last group member.

Selection of temporal parameters

On the one hand, the rationale behind 500 ms for a time bin stems from: it is neither excessively short, ensuring the inclusion of an adequate quantity of packets to yield

[3] https://mplanestore.polito.it:5001/sharing/4YNjiRhxW; additional information is available at https://smartdata.polito.it/real-time-prediction-of-packet-loss/

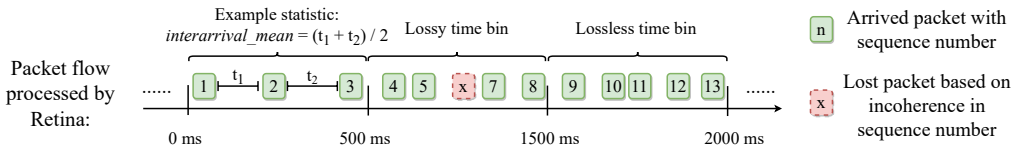


Fig. 3.1 How Retina works and the concept of time bins.

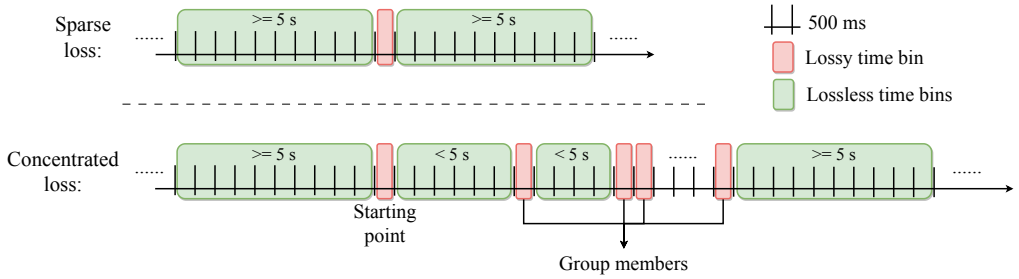


Fig. 3.2 The concept of sparse loss and concentrated loss.

representative statistics, nor too long, encompassing an excess of packets that dilute the impact of losses. On the other hand, the 5 s for different types of lossy bins is a relatively trivial choice. We aim to distinguish sporadic (e.g., non-congestion loss) and sustained losses in a straightforward manner, and 5 s is, therefore, deemed a conservative and empirically optimal value, mainly for two reasons: 1) the interval between most concentrated losses is less than 4 s, and 2) it is a safe value that a slight modification will not induce a significant impact.

3.2.2 Trend of concentrated loss

Table 3.2 presents the quantities of different time bins. In addition to the class imbalance issue (lossy bins are scarce), an imperative observation is that most lossy bins are concentrated losses regardless of datasets, which signifies a prevalent phenomenon of loss burst and suggests a higher likelihood of encountering more losses after the occurrence of a loss event. A similar observation is depicted in in Figure 3.3, which shows a shared pattern related to the duration of individual concentrated-loss groups across all datasets based on the Cumulative Distribution Function (CDF) plots. The majority (more than 80%) groups last for less than 10 s, and approximately 10% manifest a duration of 1 s, i.e., two adjacent lossy bins. That is to say, with a horizon of 10 s post a starting-point loss, it is capable to spot more than 80% concentrated groups for all concentrated losses. This, coupled with the fact

Table 3.2 Quantity of different types of time bins.

Dataset	Supervised	Campus2020	Campus2023
Total time bins	1537790	3100253	3897258
Lossy time bins	12330 (0.80%)	7009 (0.23%)	5745 (0.15%)
Sparse loss	4044 (32.80%)	1904 (27.17%)	2000 (34.81%)
Concentrated loss	8286 (67.20%)	5105 (72.83%)	3745 (65.19%)
Starting point	1741	815	814

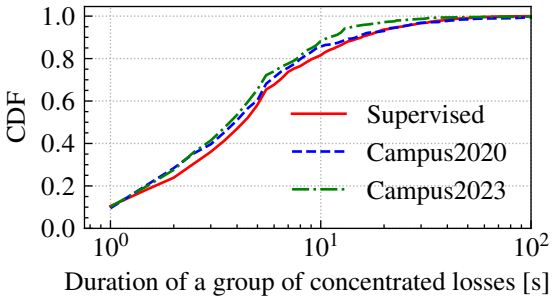


Fig. 3.3 CDF for the duration of concentrated losses (group).

that most lossy time bins are concentrated, implies the significance of starting-point lossy bins. Upon successful recognition of such losses, most subsequent losses (i.e., concentrated group members) are then inherently identified and addressed. The starting point marks the onset of a potentially severe network issue, and accurately predicting it can enable timely and effective network management. In contrast, sparse losses are theoretically difficult to predict due to their randomness and rarity [119, 120], and they are less detrimental compared to concentrated losses, which can cause continuous and lingering effects that disrupt traffic and impair QoE. Therefore, instead of targeting all types of losses, which is ambiguous and unnecessary, we focus on identifying starting-point lossy time bins. This choice is further motivated by a basic statistical observation from Table 3.2: the probability of another loss occurring after one has occurred is around 65–73%. In this context, it is unnecessary to apply a complex classifier after a starting-point loss to predict whether further losses will follow.

Furthermore, Figure 3.3 provides insight into how the monitoring system could operate. For example, upon successfully identifying a starting point, an alarm could be triggered for up to 10 s, as over 80% of concentrated groups last less than 10 s. In other words, the network issues causing packet losses are likely to persist for under 10 s, and using this strategy could potentially address more than 80% of such cases. However, this is a relatively conservative approach, since potential future losses may

not occur once system intervention begins — suggesting that a more aggressive response could also be feasible. Nonetheless, these considerations highlight the practical importance of starting-point lossy time bins and the need for an effective solution to detect them. On the other hand, both sparse losses and starting points share the concept of lossless in the previous 5 s, making it difficult to distinguish between them when a lossy time bin is observed for the first time.

We postulate an operating scenario in which the system begins at the start of an RTC stream and continues until either a lossy time bin is observed or a predicted starting-point bin is predicted. In both cases, whether a loss is detected on the fly or predicted in advance, an action lasting for a predetermined duration (e.g., 10 seconds) could be taken to optimize traffic and mitigate further losses. In the former case, it is crucial to identify whether the observed lossy bin is a starting point to avoid unnecessary actions for sparse losses without succeeding group members, and it is acceptable to make occasional mistakes, as we are dealing with only thousands of samples out of millions. However, the system may not be timely enough to address the observed loss or respond quickly to adjacent concentrated group members due to the time required for further operations. Fortunately, such situations account for only about 10% of all concentrated groups (as shown in Figure 3.3), confirming the approach's effectiveness in most cases. In the latter case, there is relatively sufficient time to react, even for predicted starting points, but the prediction operates across all time bins, which could lead to errors and unnecessary actions for lossless bins. Both approaches are proposed to accommodate different network conditions and requirements, offering a trade-off between timeliness and accuracy.

To enable successful prediction, we aim at identifying variations that can distinguish lossy time bins from lossless ones by analyzing the patterns of historical statistics prior to the target time bins. Figure 3.4 illustrates two examples of such statistics: the standard deviation and the minimum inter-arrival time of all packets in a time bin. For each statistic, the average value is computed in each time bin for the 5 s before all starting points and an equal number of randomly selected lossless time bins, respectively. Despite some fluctuations, the former statistic shows an increasing trend, while the latter experiences a slight decline irrespective of the dataset. More importantly, the statistics preceding lossless time bins remain substantially stable, as indicated by the relatively flat lines and narrow confidence intervals. This suggests that there are indeed discernible variations in the statistics leading up to packet loss events.

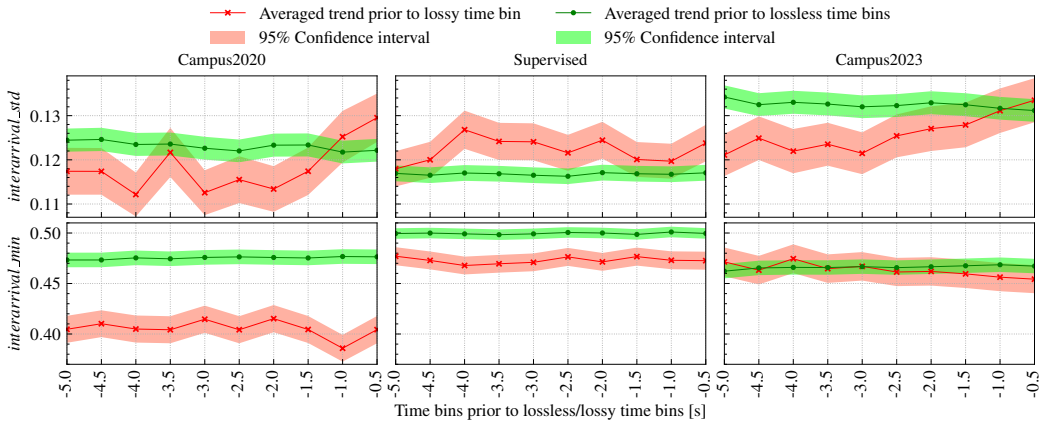


Fig. 3.4 The averaged patterns with 95% confidence interval of two example statistics (normalized value) in the past 5 seconds of starting-point lossy or lossless time bins.

Furthermore, differentiating between a starting point and a sparse loss also serves the goal of addressing concentrated group members once the target time bin is observed. However, this task is even more challenging due to the minimal distinction between the two categories. The only difference is the presence of additional losses following the starting point, but these may not significantly affect the statistics preceding the target time bin, especially if the subsequent losses occur near the end of the 5-s window. In such cases, we have to rely on ML models to detect the subtle patterns embedded in the data.

3.3 Problem statement

The overarching objective of our work is to develop a ML classifier to predict losses in RTP flows based on historical traffic statistics. Besides the potential optimization upon loss prediction, the underlying reason is manifold: 1) The acquisition of traffic statistics is seamless and convenience, and we only rely on packet headers, bypassing the complexity associated with packet encryption. 2) The reliance on statistics affords a degree of detachment from complicated network elements, avoiding the collection and analysis of pertinent information at each network hop. 3) The integration of our classifier into network live monitoring software is achievable, thus providing valuable insights into the network's status to support existing functionalities such as congestion control, bandwidth allocation, and more [121–124], as packet loss per se

stands as a fundamental performance metric, which is inherently factored into the majority of contemporary optimization technologies.

Alongside aforementioned observations, at a time instant t, we formulate two binary classification problems:

- **Problem 1, prediction of starting points of concentrated groups** – We utilize the previous 10 time bins (5 s of traffic) to predict whether the subsequent time bin is lossy or lossless. In this case, the current goal is impractical and meaningless because if we were to approach the problem of predicting losses in the immediate next time bin starting from the present moment, even if the occurrence of loss was correctly predicted, we would not have enough time to manage and instrument the network to react to this event. Therefore, we maintain the target time bin but intentionally skip the exact previous one, and consequently, a 500 ms gap is reserved for introducing a certain degree of freedom to execute operations. Formally:

$$\begin{aligned} &Y_t = f_1(\overline{x}_{t-2\Delta t}, ..., \overline{x}_{t-w\Delta t}, ...) \\ &\text{with } w \in [2, W_1],\ \overline{x} = (x_1, .., x_N),\ Y_t = \begin{cases} 0, \text{No loss} \\ 1, \text{Loss (starting point)} \end{cases} . \end{aligned} \tag{3.1}$$

- **Problem 2, classification of different losses** – We leverage the observed lossy time bin and its preceding 5 bins (6 in total, 3 s of traffic) to classify whether the observed lossy bin is a sparse or starting-point lossy time bin. Formally:

$$\begin{aligned} &T_{t-\Delta t} = f_2(\overline{x}_{t-\Delta t}, ..., \overline{x}_{t-w\Delta t}, ...) \\ &\text{with } w \in [1, W_2],\ \overline{x} = (x_1, .., x_N),\ T = \begin{cases} 0, \text{Sparse loss} \\ 1, \text{Starting point} \end{cases} , \\ &\text{s.t. } Y_{t-\Delta t} = 1. \end{aligned} \tag{3.2}$$

t is the current moment and Δt is the time bin size (500 ms). A time instant (e.g., $t - \Delta t$) represents the initial timestamp of a time bin. In problem 1, Y_t is a binary class label that denotes the presence of packet loss in the subsequent time bin starting

from the current moment t, and it takes value 1 if a loss/losses exist and 0 otherwise. In problem 2, $T_{t-\Delta t}$ is also a binary class label but represents the lossy bin type of the observed (current) lossy bin from $t-\Delta t$ to the current moment t, and it is 0 for sparse lossy bins and 1 for starting points. In both cases, $\bar{x}$ represents the input traffic statistics, and $\bar{x}_{t-w\Delta t}$ is the features vector in the past time bin at $t-w\Delta t$. In general, N is the number of different types of considered statistics (features) and W is the window size—how many time bins we use in the past. In our work, we have $N = 73$ statistics in total, and consider $W_1 = 10$ time bins (5 s in the past) for problem 1 and $W_2 = 6$ time bins (3 s including the present and the past) for problem 2. We aim at developing ML models to learn two individual functions $f_1(\cdot)$ and $f_2(\cdot)$, which map input feature vectors $\bar{x}$ to the binary classification target variables Y or T. Moreover, for problem 1, we exclude sparse losses[4], concentrated group members, and lossless time bins after them, because of both their irrelevance and the potentially misleading information introduced during model training. For problem 2, all lossless bins and concentrated group members are unneeded.

Additionally, to further clarify the selection of the 5-s time bin when defining the two loss types as well as formulating the problem, the goal is to strike a trade-off between a prolonged duration that encompasses extraneous information beyond the target time bin and a shorter interval that might overlook potentially crucial information in close proximity to the target bin. In essence, a 5-s duration is an empirically ideal temporal extent in the context of RTC, encapsulating the advantages of both breadth and precision. Meanwhile, we opt to reduce the designated duration for problem 2 to a span of 3 s, which is primarily based on the possession of statistical features within the target time bin, rendering the inclusion of temporally distant bins unnecessary. Nonetheless, the choice of time bin duration is deemed a trivial aspect, mainly due to our adoption of a technical approach that effectively selects informative features, thereby mitigating the inherent time constraints.

[4]We remove sparse losses during model training to avoid inadvertently learning pertaining patterns. However, the model may still predict a sparse loss and identify it as a starting point, which, nevertheless, presents no significant concern in practice, because (1) if a sparse loss is predicted as a starting point, it is not inherently erroneous, as a sparse loss indeed represents a lossy bin, and (2) if a sparse loss is not identified, its impact is likely negligible since there are no subsequent losses to compound the ramification. It is exactly due to the same reason that we also omit sparse losses during the testing phase without explicitly factoring them into the ultimate performance assessment.

Challenge

Notably, we identify multiple challenges: 1) For problem 1, we are obliged to bypass the hithermost time bin, which may contain significantly informative patterns. 2) Our datasets are extremely imbalanced (lossy bins only occupy 0.15% to 0.80%), not to mention the even less amount of our predicting target — starting points. In ML domain, models trained on such data typically exhibit a bias towards the majority class [125, 126]. 3) For problem 2, we have a limited number of available samples that are scant for a data-driven problem, let alone the same lossless situation in the past 5 s, which obfuscates the patterns and exacerbates the complexity of classification. 4) RTC is intrinsically dynamic and constantly evolving, influenced by extensive factors that generate diverse patterns. Meanwhile, we intentionally segment RTP flows for model training and testing rather than shuffling samples to resemble the reality, where the incoming traffic is always novel to the model, not to mention that the training and testing are performed separately on different datasets spanning three years.

3.4 Methodology

In the following, we illustrate the whole workflow of ML model development, as portrayed in Figure 3.5. In general, the raw *pcap* files are initially parsed by Retina to construct the structured dataset. Afterwards, we perform a model development process, undertaking a two-step feature selection process and examining different ML models and sampling strategies. Finally, we proceed to consolidate the model performance.

3.4.1 Dataset construction

The output of Retina (a sample) represents a time bin composed of the 73 traffic statistical features and two class labels for the two problems. To construct the structured dataset, we incorporate the previous 5 s of history for each time bin, turning each sample into 11-bin long (the current bin and the 10 preceding time bins), and generating 803 features (73 statistics $\times$ 11 time bins).

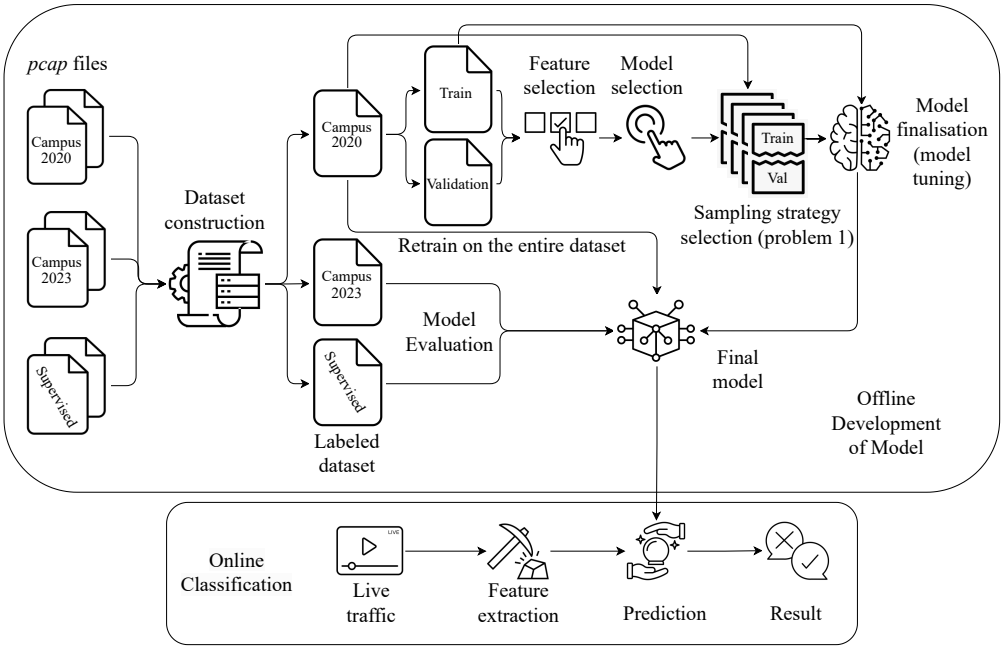


Fig. 3.5 Overview of the ML pipeline.

3.4.2 Model development

In order to justify the generalizability, versatility, and transferability of ML models, we explore the feasibility of training a model on an endpoint, e.g., campus router, observed three years ago, that can continue to identify relevant patterns despite the passage of time. In this context, "Campus2020" serves as a basis (training and validation sets) for the model development, and the entire "Supervised" and "Campus2023" datasets (test set) are used to evaluate the ultimate performance. And to ensure the independence of training and validation datasets, and to prevent data infiltration between RTP streams during the model training, the "Campus2020" dataset is split in a way that flows to which the data in training set belong to, would not present in the validation set. Specifically, we randomly shuffle the flows instead of individual samples, and the dataset is partitioned such that 70% of the data are allocated to the training set, while the remaining 30% are in the validation set[5]. Tailored to problem 1, we undertake a multiple-trial evaluation process to determine the optimal sampling strategy. Then, we refine the best choice by tuning hyper-

[5]In cases when early stopping is needed, one-seventh of the training set, equivalent to 10% of the entire dataset, is reserved.

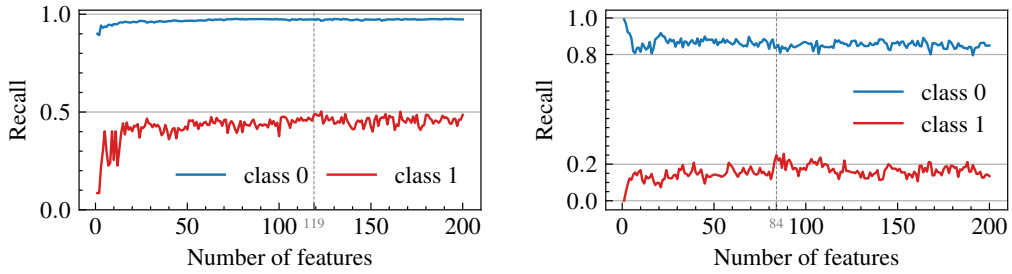
parameters of the selected model (with the selected sampling strategy) based on the performance on the validation set. Upon achieving the optimum configuration, we retrain and finalize the model on the entire "Campus2020" dataset.

Feature engineering

First, for problem 1, the current and the exact last time bins are removed, transforming the number of features in each sample to $73 \times 9 = 657$. For problem 2, the targets are the current time bins that we already observed, with unneeded time bins from the past 2.5 to 5 s (5 bins) discarded and $73 \times 6 = 438$ features left. Second, we perform feature engineering to remove redundant features. Consequently, the model development is initialized with a basic data splitting and then a two-step feature selection process as follows:

1. Correlation analysis: We evaluate the correlation between each pair of features and remove one of them randomly, if the Pearson correlation coefficient is greater than 0.9 (in absolute value). As a result, we retain 61 out of the 73 statistics in original time bins.

2. Feature refinement: We employ Recursive Feature Elimination (RFE) [127] approach to systematically examine the importance of features and iteratively remove insignificant ones based on scores provided by an eXtreme Gradient Boosting (XGB) [128] classifier. The whole procedure is recursively repeated to inspect the impact of number of features, refining and keeping the most informative ones for the problems. Figure 3.6 reports the result of feature selection process in terms of class recalls with respect to the number of features. For problem 1, class 0 quickly stabilizes and reaches a plateau, while class 1 undergoes dramatic changes in the early stages, followed by marginal improvements with minor fluctuations. We choose to include 119 features, which enables the recall for class 1 to reach 0.5 for the first time. For problem 2, both classes produce volatile variations even with more features. We select 84 features, which yields the best performance for class 1. Although we do not observe a prominent inflection point, known as a "knee", beyond which adding more features becomes unproductive, we adopt a conservative approach and include a slightly higher number of features. Nevertheless, we effectively remove a significant portion (more than 80%) for both problems.



(a) Problem 1. (b) Problem 2.

Fig. 3.6 The class recalls of the result for RFE process.

Model selection

For problem 1, two categories of models are considered: 1) ML classifiers that perform an unbalanced binary classification task in a supervised manner — 3 tree-based models of Decision Tree (DT) [129], Random Forest (RF) [130], eXtreme Gradient Boosting (XGB) and 2 DL models of Deep Neural Network (DNN) [131] and Long Short-term Memory (LSTM) NN [132]. 2) Unsupervised learning methods that frame an anomaly detection problem — Isolation Forest (IF) [133] and Autoencoder (AE) [134]. Specifically, to handle class imbalance in the case of classification, we implement the following techniques:

- Weighted sampling is introduced during the model training in two ways: 1) We construct the so-called Balanced Random Forest (BRF) [135], drawing bootstrap samples from the minority class and then randomly drawing the same number with replacement, from the majority class. 2) For DL approaches, the training batch is constructed by sampling data with given probabilities (i.e., minority samples have a higher chance to be selected).
- Assigning class weight can adjust ML models so that misclassifying a sample from the minority class is more heavily penalized than misclassifying a majority-class sample. During the training phase, the weight is set according to the multiplication between the majority and minority classes.

Since anomaly detection algorithms are specifically designed to cope with rare events, no additional workarounds are required. As for problem 2, since it is a relatively simple classification problem with comparable class quantities, we only implement the 3 tree-based models and the k-nearest neighbors (k-NN) algorithm [136].

Sampling strategy

To further address the imbalance issue in problem 1, with the best model, we resort to artificial sampling to intentionally manipulate the quantity of samples through oversampling (increase the amount of minority class), undersampling (decrease the amount of majority class), or hybrid-sampling (combination of oversampling and undersampling). In our case, we refer to Synthetic Minority Oversampling Technique (SMOTE) [137] for oversampling and hybrid-sampling, and a random selection for undersampling. Particularly, for each sampling strategy, we perform 50 trials of training and validation, each featuring data from different randomly selected RTP flows, i.e., a 50-fold cross-validation-like process that evaluates the general performance.

3.5 Experiment

Herein, we present the experimental results of each problem separately with a two-stage evaluation process:

1. **Preliminary model evaluation** focuses on the ML models themselves and aims to uncover the most promising candidate. Again, models are trained and validated exclusively on "Campus2020" dataset.

2. **Model performance finalization** aims to consolidate the model performance, by training the best model from the previous stage on the entire "Campus2020" dataset and testing the finalized model on other datasets.

The general procedure is similar for both problems, except for extra operations needed for problem 1. Importantly, due to the highly imbalanced nature of problem 1, many evaluation metrics for classification are biased, resulting in a distortion of the true performance. Therefore, we opt to use recall, to reflect the model sensitivity and evaluate the performance of each individual class[6].

[6]Unlike most imbalanced problems that devote to improving the performance of the minority class, we expect to avoid excessive false positives that could trigger unnecessary false alarms and thus squander network resources. It is, therefore, critical to guarantee a commendable performance for lossless samples, while simultaneously enhancing lossy ones, which can only be explicitly assessed by the class recall.

3.5.1 Experimental result of problem 1

Preliminary model evaluation

Table 3.3 showcases the class recalls on "Campus2020". All required configurations tackling class imbalance are implemented for each model, while no sampling strategy is utilized. Generally, anomaly detection approaches fail to fulfill the objective, as evidenced by their negligible recalls for class 1. DT and RF models perform marginally better but still fall short in effectively addressing the problem. DL models exhibit the capability of identifying abundant starting points but yield the worst performance for class 0. Remarkably, the advanced tree-based models, BRF and XGB, deliver substantially superior outcomes with correct predictions of approximately 50% of the target and without excessively penalizing class 0. To this end, we select XGB for further analysis, thanks to its distinguished performance and ease of training and tuning. Additionally, we still present class precision and macro average F1-score in Table 3.3 to provide supplementary insights into the performance evaluation. Evidently, all models exhibit a consistent behavior, characterized by unexceptionable precision for class 0, inferior precision for class 1, and mediocre F1-score, which reinforces again our choice of the evaluation metric. Only XGB and BRF that strike a relative balance between both classes, yield slightly higher precision for class 1, which is attributable to the dominance of the majority class, which overwhelms the minority in terms of the predicted positives, thus leading to a relatively constant F1-score of around 0.5. Notably, the diminished precision for class 1 is not deemed a significant issue, as this represents an inherent property for imbalanced ML [126], and the performance presented herein are merely from the preliminary model development.

An additional step of examining different sampling strategies is needed for problem 1. We perform multiple-trial evaluation on the best model (XGB) selected in the previous stage on "Campus2020" dataset, and test the model also on the entire "Campus2023" and "Supervised" datasets[7]. The box plots in Figure 3.7 depict class recalls of all experimental trials for each sampling strategy. Overall, all strategies share similar patterns, manifesting decent, stable performance for class 0 and varying degrees of overfitting for class 1. Moreover, the ranges of recalls for class 1 are wide regardless of sampling strategies, meaning that patterns extracted from training

[7]Notably, the "Campus2023" and "Supervised" datasets are included here just to verify the unified pattern, and they are not considered for the selection of sampling strategy.

Table 3.3 Basic results of all ML models on "Campus2020" for problem 1.

Category	Model	Recall		Precision		F1-score
		Class 0 (Lossless bin)	Class 1 (Starting point)	Class 0	Class 1	Macro Average
Classification (Tree-based)	DT	0.988	0.045	0.999	0.001	0.498
	RF	0.996	0.051	0.999	0.001	0.499
	BRF	0.962	0.486	0.999	0.003	0.494
	XGB	0.973	0.522	0.999	0.005	0.498
Classification (DL-based)	DNN	0.856	0.537	0.999	0.001	0.462
	LSTM	0.875	0.652	0.999	0.001	0.468
Anomaly Detection	IF	0.973	0.020	0.999	0.001	0.493
	AE	0.999	0.001	0.999	0.001	0.500

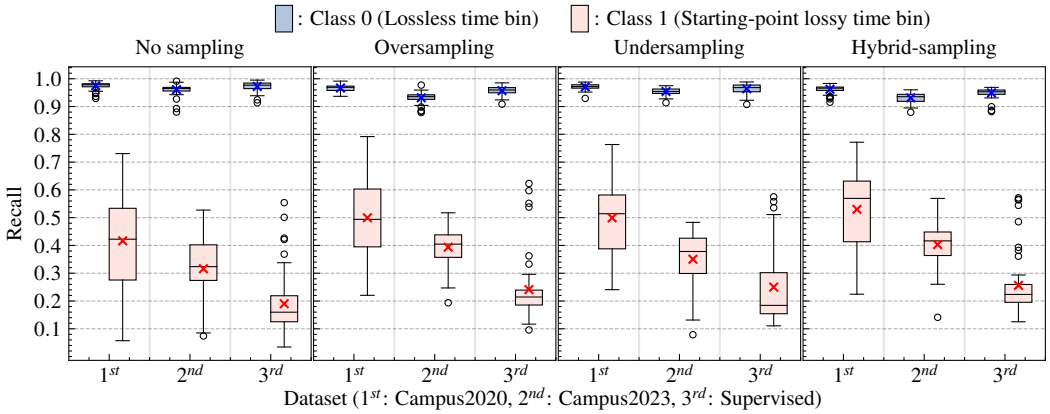


Fig. 3.7 Class recalls of results based on different sampling strategies for problem 1.

samples in different flows have a huge impact on the final performance, which necessitates the fine-tuning of models with all samples in the entire "Campus2020" in the next step. Regarding each sampling strategy, we observe that no sampling ranks the first for class 0, while hybrid-sampling generates comparatively improved performance for class 1. Therefore, we continue the process with both of them.

Model performance finalization

We now finalize the performance by training XGB on the entire "Campus2020" dataset with selected sampling strategies, as in Table 3.4. Accordingly, no sampling outperforms towards lossless bins by identifying around 96% of class 0, while hybrid sampling generates comparatively balanced results with higher recalls for class 1, reaching up to around 66% of starting points. In both cases, "Campus2023"

Table 3.4 Final result of models trained on the entire "Campus2020" dataset for problem 1.

Model		XGB (No sampling)		XGB (Hybrid-sampling)	
Test dataset		Campus2023	Supervised	Campus2023	Supervised
Recall for each dataset	**Class 0 (Lossless bin)**	0.960	0.956	0.903	0.915
	Class 1 (Starting point)	0.553	0.454	0.662	0.524
Recall for all test data	**Class 0 (Lossless bin)**	0.959		0.906	
	Class 1 (Starting point)	0.486		0.568	

Table 3.5 Basic results of all ML models for problem 2.

Category	**Model**	**Recall**	
		Class 0 (Sparse loss)	**Class 1 (Starting point)**
Classification (Tree-based)	DT	0.647	0.363
	RF	0.934	0.061
	XGB	0.825	0.184
Classification (Instance-based)	k-NN	0.762	0.310

outperforms "Supervised" dataset, which is reasonable since it was collected using the same tool at the same location as the training set. On top of that, by successfully predicting the starting point, we actually identify and tackle subsequent concentrated-loss groups of 49% to 57% across all test sets. Although the general prediction for lossy time bins is not optimal, the performance remains acceptable given the difficulty of the problem, let alone the prevention of discernible concentrated losses in the future, which could be more theoretically adverse.

3.5.2 Experimental result of problem 2

Preliminary model evaluation

Table 3.5 elucidates the class recalls of each model. In general, the overall performance is subpar, and there is a prevailing bias towards class 0 in all models, with their performances exhibiting an inverse relationship: a higher recall for class 0 correspond to a lower recall for class 1, and vice versa. To this end, we select XGB to take advantage of its flexibility in configurability and model tuning to achieve varying levels of performance.

Model performance finalization

The primary goal entails discerning starting points to the fullest extent possible, whilst ensuring a satisfactory identification for class 0. However, considering the relatively limited number of samples for class 0 across all datasets, misclassifications are moderately tolerable. This endows us with the opportunity to meticulously tune the model to achieve different performance levels, either prioritizing a balance between both classes or prioritizing class 1. Consequently, we deliberately train the model to target different optimization directions and present the finalized performance in Table 3.6.

For the main objective, the model effectively identifies the majority of sparse lossy bins and successfully recognizes 24% to 40% of starting points. When focusing on balanced performance, the model achieves a 60% recall for both classes in "Supervised" dataset, but only surpasses random guess for "Campus2023" dataset. In the last case, the performance is exactly opposite to the main objective, but the overall accuracy is below 50% due to the larger number of samples in class 0. Across all cases, the performance on "Supervised" dataset is substantially better than that on the "Campus2023" dataset, whose overall performance remains inferior with recalls barely reaching 0.8 on one end but merely achieving around 0.25 on the other end. In contrast, the results of "Supervised" dataset are sufficiently respectable, i.e., the model can identify 40% of starting points, while only yielding 15% misclassifications. To sum up, our approach demonstrates the prospect to satisfy different system requirements and the effectiveness in certain specific scenarios.

3.5.3 In-depth analysis

Herein, we further examine the outcomes, providing deeper insights into two aspects: 1) the reason behind the prediction difficulties, and 2) the distribution of classifications for starting points among RTP flows.

Analysis of classification

In order to understand the features which ML models (data-driven methods) generally rely on to distinguish different classes, we visualize their distribution through T-

Table 3.6 Final result of models trained on the entire “Campus2020” dataset for problem 2.

Model	Test dataset	Recall for each class	
		Class 0 (Sparse loss)	Class 1 (Starting point)
Main Objective			
XGB	Campus2023	0.818	0.236
	Supervised	0.849	0.400
Different Optimisation direction (Balanced performance)			
XGB	Campus2023	0.529	0.502
	Supervised	0.620	0.610
Different Optimisation direction (Better performance for class 1)			
XGB	Campus2023	0.263	0.824
	Supervised	0.336	0.862

distributed Stochastic Neighbor Embedding (t-SNE) [138][8], which transforms high-dimensional data to low-dimensional embeddings, that can be displayed in a 2D plot. For each finalized model and dataset, we extract features that correspond to the starting-point losses and the same amount of random lossless samples for problem 1, and to sparse losses and starting points for problem 2, which are then converted into 2D embeddings to make scatter plots.

For problem 1 (Figure 3.8), the light green (class 0) and red (incorrect predictions for class 1) points exhibit a wide distribution, intertwining with one another, while the blue dots (correct predictions for class 1) are dispersedly allocated along the periphery, isolatable in "Campus2023" dataset, yet tending to coalesce into "Supervised" dataset. Overall, the discernment between class 0 samples and misclassifications from class 1 proves arduous given their overlapping nature, further compounded by the inclusion of only a limited quantity of lossless time bins, thereby underscoring the exacerbation of overlap with an abundance of class 0 samples. For problem 2 (Figure 3.9), samples of either class/dataset are also scattered extensively across the plot, resulting in a highly overlapped pattern, which could originate from the shared characteristic of being lossless in the past for both classes. In this vein,

[8]It is important to acknowledge that t-SNE transcends dimensionality reduction, epitomizing a more sophisticated approach that strives to encapsulate and retain intricate patterns in the original high-dimensional space. Thus, the visual representation serves as a seamless conduit to reinstate the originality and authenticity of patterns, thereby facilitating an enhanced comprehension of different features in a visually intuitive manner.

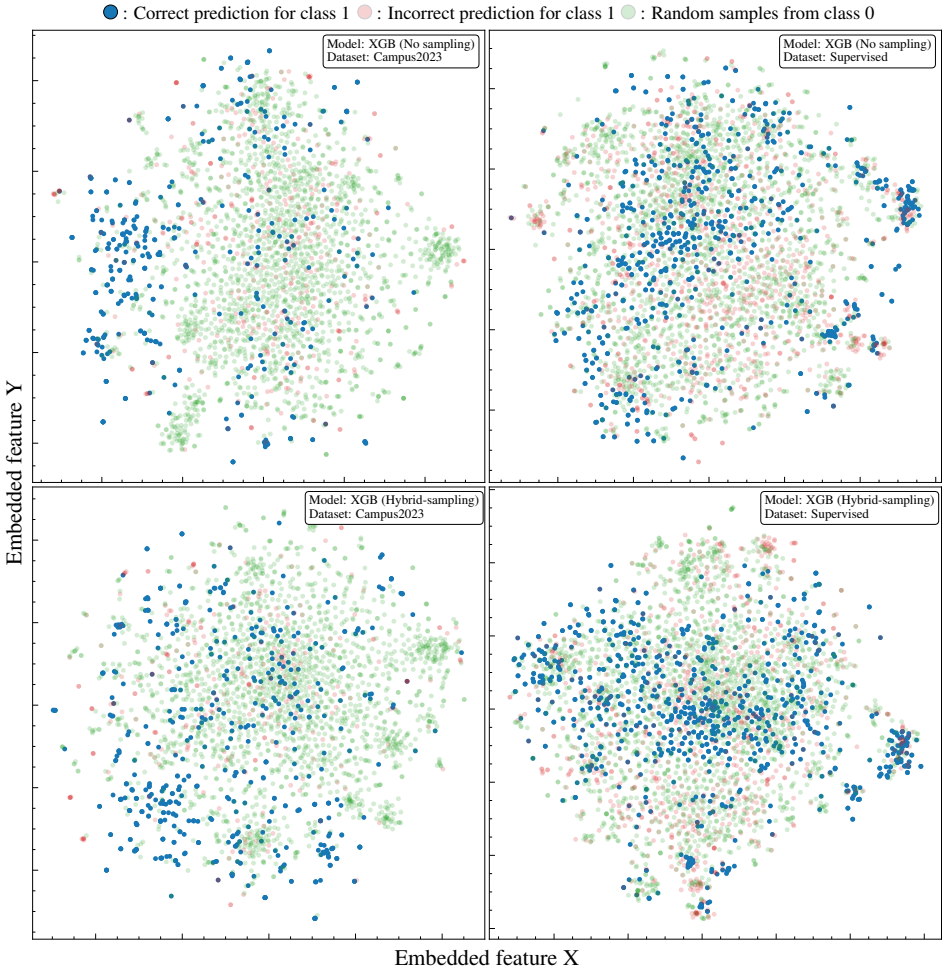


Fig. 3.8 Results of t-SNE for problem 1: the 2D plot for the embedding of correct prediction (blue dots) and incorrect prediction (light red dots) from class 1, and random samples from class 0 (light green dots).

the model tuning resembles the manipulation of decision boundary (hyperplane in multidimensional space), and due to the mixed nature of both classes, it is barely possible to accurately classify one class while effectively isolating the other one. All these observations collectively highlight the inherent challenges posed by the problem, where RTP traffic are dynamic and multifarious, engendering various nearly indistinguishable patterns.

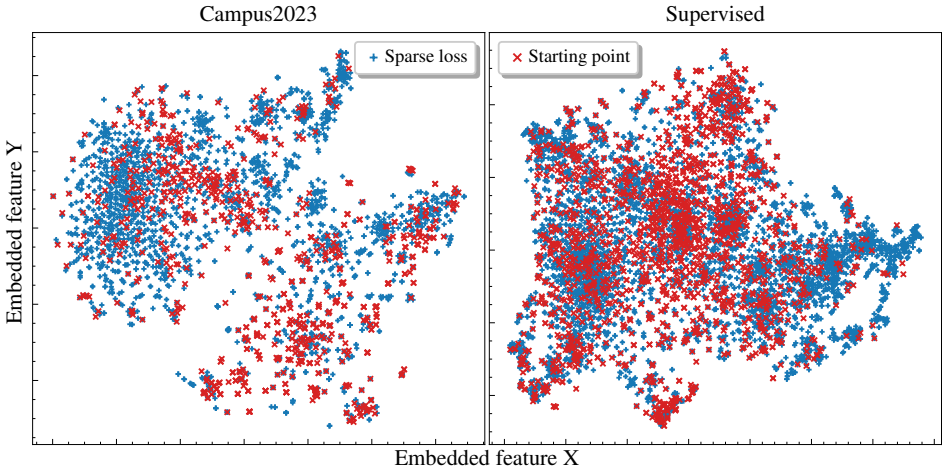


Fig. 3.9 Results of t-SNE for problem 2: the 2D plot for the embedding of class 0 (sparse lossy time bins, blue plus marker) and class 1 (starting-point lossy time bin, red cross marker).

Analysis of RTP flow

We now examine the distribution of correct and incorrect classifications across RTP flows for both problems, as depicted in Figure 3.10 and Figure 3.11, which illustrate the ratio of accurately predicted instances to the total number of starting points for individual RTP flows.

For problem 1, the prediction distribution exhibits a substantial bias for both datasets, with the model excelling in predicting some flows but performing poorly for others, and only several flows yielding mediocre results. This implies a unified pattern of packet loss in certain RTP flows, which the model can effectively capture and leverage to achieve reliable predictions consistently. This particularity may facilitate a management strategy that terminates predictions completely in the presence of multiple initial errors. Interestingly, the first several flows with a significant number of starting-point losses in both datasets experience exceptional performance, which could signify a specific capability of the model to efficaciously predict RTP streams heavily affected by packet loss. For problem 2, despite the first 4 flows in "Supervised" dataset with highly accurate classification, the overall distribution is relatively homogeneous. Different from problem 1, where flows with more losses are favored, the models in problem 2 are able to generalize across flows.

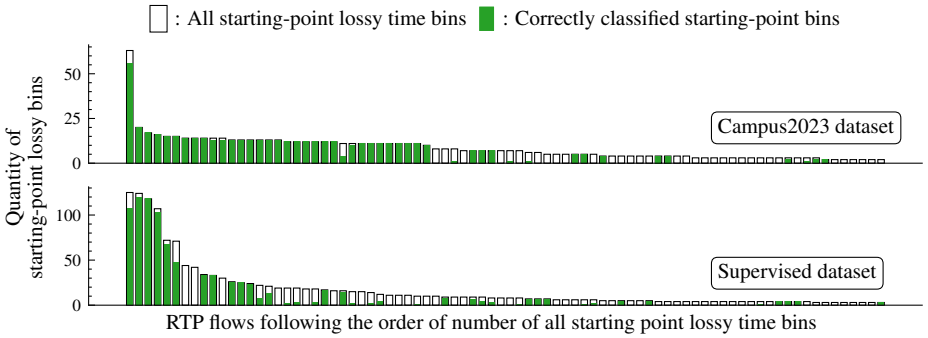


Fig. 3.10 Quantity of starting points in each individual RTP flow for problem 1: the overall amount (white bars) and the correctly predicted amount (green bars). For simplicity, only the two datasets examined by XGB with hybrid-sampling are presented and only the 80 flows with relatively more losses are presented. We notice that the other model and flows exhibit similar behavior.

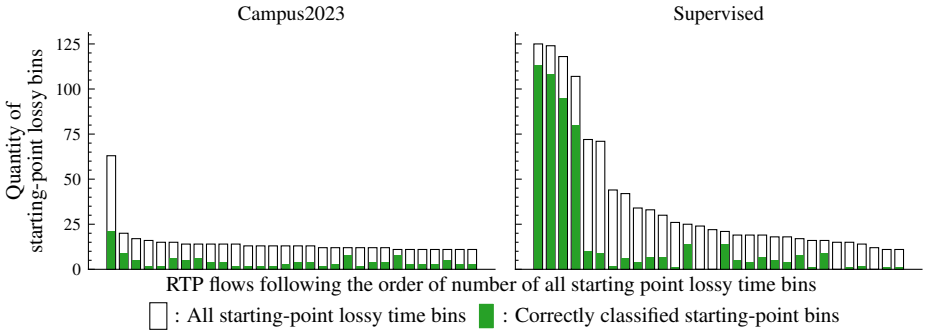


Fig. 3.11 Quantity of starting points in each individual RTP flow for problem 2. Note that, only the two datasets regarding the main objective of problem 2 and the first 30 flows are presented.

3.5.4 Supplementary material

In the original paper, for problem 1, we also analyze the false alarm for class 0 and the performance for sparse losses. In the former case, we observe a higher rate of misclassification for lossless bins when they are located close to lossy bins; however, this effect can be advantageous, as such bins may trigger alarms just a few seconds ahead of the actual losses. In the latter case, the proportion of sparse losses classified as starting points is evidently low, which is consistent with our objective of ignoring sparse losses that that do not herald subsequent concentrated losses. Additionally, we examine the impact of time-window length in the appendix to justify our choice. For further details, please refer to our paper [9].

3.6 Takeaways

This chapter presents a comprehensive study of packet loss in RTC, highlighting its bursty nature and exploring the use of ML models to predict its commencement. Our findings can be summarized as follows:

- Based on extensive RTP traffic (statistical observations) collected from diverse vantage points, time periods, and applications, packet losses generally occur in two forms: sparse and concentrated.
- Sparse losses are infrequent, arbitrary network events that only account for roughly 30% of all losses. In contrast, most losses are concentrated, occurring consecutively within a relatively short interval.
- This observation reminds us of the importance of the onset of concentrated losses. If such onsets can be predicted, the subsequent losses can be, therefore, addressed potentially in advance. Since concentrated losses dominate overall loss and sparse losses are comparatively less harmful, the identification of concentrated losses through onset prediction is particularly valuable.
- We formulate the problem in two ways, wherein the first one aims to predict the starting point of concentrated losses, while the second one aims to distinguish between start points and sparse losses. We examine multiple ML models and find that the advanced tree-based model, XGB, stands out as a promising candidate, delivering superior and balanced performance. For problem 1, XGB successfully forecasts approximately half of starting points while maintaining robust performance for lossless cases. For problem 2, the results are less favorable due to the inherent difficulty of the task, with overall performance being only modest.
- At first glance, the performance is not preeminent regardless of problems, but we still advocate for our solution because: 1) the model can be optimized towards different directions to meet varying requirements, and 2) our problems are framed in a naturally tough way, going beyond naive and common classification tasks while offering greater relevance and benefits for RTC scenarios.

- Since our attention is paid to end-to-end RTP traffic, a key defect of our work is the absence of root cause analysis for packet loss, which could further improve ML models and represent an important future direction.

Returning to the research questions outlined in the background, ML models can indeed facilitate loss prediction, albeit subject to several important constraints.

Chapter 4

Fine yet Single: Throughput Estimation with Emphasis on Traffic Extremes

This chapter corresponds to our work: *DeX: Deep Learning-based Throughput Prediction for Real-Time Communications with Emphasis on Traffic eXtremes* [10]. Throughput represents a critical performance indicator in RTC, as it directly reflects the total volume of traffic traversing network nodes. In general, higher throughput indicates better performance, particularly for RTP traffic that encapsulates multimedia content. A related term, goodput, refers to the subset of throughput that excludes functional packets, retransmissions, etc., representing the actual amount of effective traffic. Since RTP involves no retransmissions and functional packets typically account for only a negligible portion, throughput and goodput can be regarded as equivalent in RTC[1].

Building on the background in Chapter 2, this chapter discusses a DL solution for traffic prediction. Particularly, we aim at predicting the near-future throughput of received RTP traffic based on packet-level information, with special focus on traffic extremes, i.e., peaks, valleys, and abrupt changes. We consider the scenario of video-teleconferencing and frame a time-series problem, for which we develop a dedicated DL model named *DeX* and compare it with multiple ML/DL algorithms. In

[1]This equivalence does not always hold, as error correction flows may contribute a non-negligible portion in certain RTC applications. Nevertheless, we consistently use the term throughput for definitional clarity.

the following, Section 4.1 introduces the background, related work, and motivations. Section 4.2 formulates the problem, and Section 4.3 delineates our proposal. Afterwards, Section 4.4 showcases the experimental results. At last, several conclusive remarks are presented in Section 4.5.

4.1 Background

Bandwidth management plays a pivotal role, involving crucial functionalities in RTC, e.g., throughput measurement, bandwidth allocation, dynamic transmission adjustments, and traffic prioritization [139–141, 106]. In light of this, throughput prediction holds immense potential, providing a proactive mechanism with manifold merits: 1) bandwidth allocation becomes optimized based on precise throughput estimations to avoid either underutilization or over-provisioning, 2) QoE and content dissemination can be improved via prediction-informed adaptive streaming, transcoding, encoding settings, and more, 3) network congestion management can take advantage of throughput forecast, fostering traffic shaping, prioritization, routing, etc., and 4) resource planning and system scalability are more effective in accordance to the predicted throughput demands. Specifically, throughput metrics retrieved from end-users are frequently used as vital feedback in modern optimization approaches, including decision making agents, the software-defined networking (SDN) paradigm, congestion control mechanism, local integrated tools, and beyond [142–148], and the predicted throughput thereby transcends the observed value, assuming an instrumental role in preemptively informing network units and enabling proactive reactions.

Throughput prediction, akin to bandwidth or bitrate prediction, is a typical problem of traffic prediction and has gained considerable attentions in networking field. Authors in [149] tried to improve bitrate selection via throughput prediction based on a Hidden-Markov-Model (HMM) predictor, and analyzed millions of sessions, discovering similar and stateful patterns that are used to group sessions. [150] focused on predicting average throughput in cellular networks, utilizing a Random Forest (RF) regressor based on historical throughput measurements, alongside several radio channel metrics. The work was then extended [151] to examine the applicability of Support Vector Machine (SVM) [152] and Long Short-Term Memory (LSTM) NN. The work in [153] proposed a linear adaptive filter, Recursive Least Squares (RLS)

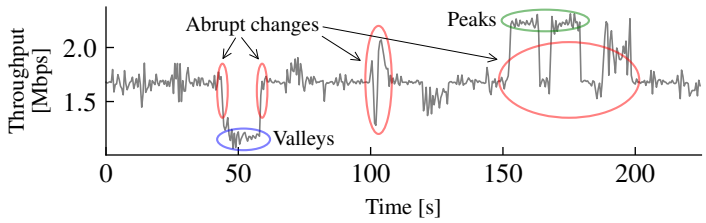


Fig. 4.1 Throughput time series of sample traffic.

to forecast forthcoming bandwidth based on historical values, aiming to enhance transmission quality of video calls in cellular networks. [154] introduced LinkForest, a RF-based framework, to predict cellular link bandwidth in 4G Long Term Evolution (LTE) networks, employing lower-layer metrics such as Reference Signal Received Power (RSRP) and previous throughput data as input features. The authors in [155] leveraged public datasets to perform short-term throughput prediction, and adopted multiple ML algorithms including tree-based models and DNNs, relying on aggregated features in time windows such as cumulative bitrate and number of packets. Moreover, both [156] and [157] incorporated the LSTM NN but with customized designs, where the former work explored real-time mobile throughput prediction, augmenting a pre-trained LSTM framework with model switching and Bayes model fusion for online deployment, and the latter one intended to enhance adaptive video streaming by proposing a RL paradigm, in which an LSTM combined with a Convolutional Neural Network (CNN) are responsible for bitrate prediction. Meanwhile, targeting Adaptive Bitrate (ABR) for HTTP-based video streaming, works in [158–161] employed various ML/DL or statistical tools to predict throughput, encompassing tree-based models, a DNN empowered by K-means clustering, a renowned RL-based engine, or a Kaufman's Adaptive Moving Average (KAMA) method, which are then integrated into ABR decision-making systems.

All aforementioned methods demonstrate different degrees of effectiveness and generally frame a time-series problem. However, due to their devoted yet irrelevant scopes or relatively naive problem formulation/model development, none of the works is able to provide methodological superiority in terms of prediction precision, especially struggling with the extreme conditions, which constitute critical facets in RTC traffic. Traffic extremes, i.e., **peak values**, **valley values**, and **abrupt changes**, represent the intricate nuances of network dynamics and significantly influence prediction accuracy. An time-series example with traffic extremities highlighted is

presented in Figure 4.1 to provide context. In particular, we underscore extreme values for the sake of three reasons: 1) Transmission peaks often coincide with network bottlenecks, providing invaluable insights into the prospective bandwidth availability, and their accurate predictions can, therefore, optimize resource allocation to avert potential pitfalls. 2) Valley values denote traffic idleness and underutilized network capacity, supporting energy-conservation strategies, resource redistribution, and load balancing. 3) Abrupt changes (i.e., sudden and transient network fluctuations), if anticipated correctly, could enhance adaptive streaming agility and expedited network management, ensuring a swift adaptation to rapid network oscillations.

Fine yet Single

Our previous work in Chapter 3 follows a typical feature engineering process of aggregating unitary entities into summarized representations from which features are derived. However, although effective in practice, this method is inherently a compromise, resulting in inevitable information loss. The term "fine" here means our shift from coarse, compressed features to fine-grained, packet-level features. The rationale behind is fourfold: 1) Packets represents the most fundamental and granular network units, epitomizing the rapidly changing dynamics and inherent characteristics of network traffic [162], which endows models built on such meticulous features with the capability to effectively discern underlying traffic patterns. 2) The acquisition of packet-level data incurs minimal effort regarding feature extraction, an particular advantage in RTC, where temporal and computational constraints are common. 3) We exclusively use header attributes to bypass issues from packet encryption, facilitating a more streamlined workflow with expeditious access to pertinent information. 4) Packets are ubiquitously available, extending beyond the confines of client sides, and thus affording a more holistic network observability, which enables throughput prediction to improve the overall performance within the network.

Regarding packet-level-related prediction, a limited corpus of research exists. The authors in [163] aimed to not only utilize packet-level information but also to predict packet-level characteristics (e.g., packet direction and payload length). They explored multiple ML/DL techniques, and arranged packets with three predicted and three exogenous parameters (e.g., TCP window size) sequentially to enable sliding window prediction. Both [164] and [68] applied the Transformer architecture [165]. The former work proposed FlowFormer to classify real-time network flow

types (video, conference, and download), by implementing layers of attention-based encoder to extract features that are then fed to a LSTM or a CNN model. The latter one sought to generalize network dynamics, introducing the Network Traffic Transformer (NTT) framework, wherein an additional hierarchical aggregation layer is added before the encoder to condense lengthy packet sequence. The authors initially pre-trained the model based on end-to-end delay predictions, and envisioned a replaceable decoder for other tasks.

To the best of our knowledge, our work is the first to employ packet-level information in throughput forecast with an emphasis on traffic extremes to bolster predictive performance. Although the method in [158] can indeed detect abrupt changes, the key feature of chunk size is unavailable in RTP traffic. Notably, the overall target of throughput prediction remains "single", as the current scope is on traffic extremes and model development.

4.2 Problem statement

4.2.1 Problem formulation

At any time instant t, we intend to predict traffic throughput in the forthcoming time window of Δt. For a comprehensive evaluation, we formulate a regression problem in a dual manner with different features but the same target: *i)* a conventional univariate time-series problem with historical samples as features, and *ii)* an irregular multivariate one using preceding packet-level features. Specifically:

- Problem *i* — univariate time series prediction:

$$\begin{aligned} \hat{R}_t &= f(X) \\ \text{with } X &= [r_{t-\Delta t}, r_{t-2\Delta t}, ..., r_{t-m\Delta t}, ...],\ m \in [1, M]. \end{aligned} \tag{4.1}$$

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

$$\begin{aligned} \hat{R}_t &= f(X) \\ \text{with } X &= [\underbrace{..., \bar{x}_{t,n}, ...}_{W}],\ n \in [1, N]. \end{aligned} \tag{4.2}$$

$\hat{R}_t$ is the predicted throughput in the subsequent time window from t to $t+\Delta t$. The input feature matrix X varies between the two problems. Problem i considers M historical samples, each of which (i.e., $r_{t-m\times\Delta t}$) denotes the previous throughput value in the window that commences at $t-m\times\Delta t$. For problem ii, we factor in a total of N packets in the past, while only selecting a subset of $W<N$ packets as features[2]. Should the n^{th} preceding packet antecedent to time t be designated as one of the selected packets, $\bar{x}_{t,n}$ represents its corresponding feature vector constituted by packet attributes. We aim to learn a function $f(\cdot)$ to undertake the regression task. Technically, the three extreme conditions during each individual video-call session are defined as follows:

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

$$\frac{|R_t - R_{t-\Delta t}|}{R_{t-\Delta t}} > \beta. \tag{4.3}$$

Note that α_p, α_v, and β are percentages.

4.2.2 Dataset

The traffic employed in this work is from the two major RTC applications, Jitsi Meet and Webex. As indicated by the problem formulation, we construct the dataset by generating the time-series of throughput samples for each individual session. Specifically, the predicted target of throughput value is calculated in successive time windows following chronological order, by aggregating the frame length of all packets within each window (i.e., the traffic amount per unit time). In this context, we select 3 elements (i.e., $\bar{x}_{t,n}=[x^{\text{FL}}_{t,n}, x^{\text{IAT}}_{t,n}, x^{\text{TS}}_{t,n}]$) in each RTP packet as features:

[2]They are not necessarily contiguous, and the remaining $N-W$ non-selected packets are simply discarded.

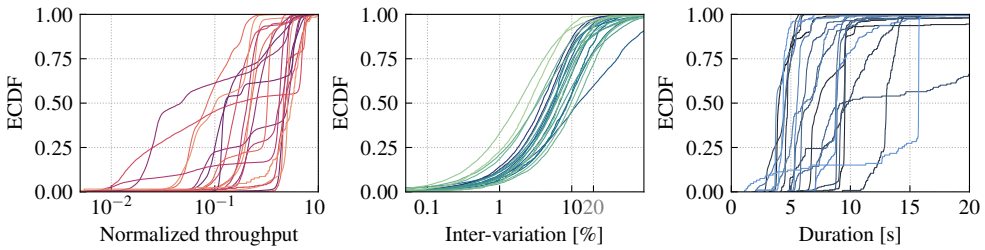
- **Frame length** ($x^{\text{FL}}_{t,n}$) is the packet total size including both its header and data, which directly represents the impact of transmitted bits in the past, endowing the model with the capability to operate in an autoregressive manner.
- **Inter-arrival time** ($x^{\text{IAT}}_{t,n}$) is the temporal gap between the arrivals of two consecutive packets, and it serves as a local granularity indicator to assess the frequency of packet flows.
- **Timestamp** ($x^{\text{TS}}_{t,n}$) denotes the relative timestamp at which the packet is received. It is the current absolute timestamp subtracted by the timestamp of the session start and it introduces global timing patterns.

With these features, we aim to incorporate both spatial and temporal patterns. Consequently, each time window (a data sample) is accompanied with M historical throughput samples for problem i, and the preceding N packets with aforementioned packet-level features for problem ii.

Initially, we set time windows to $\Delta t = 500\ ms$ and aim to select $W = 1024$ out of the previous $N = 2048$ packets for problem ii. Because of an average of roughly 7.6 s occupied by 2048 packets in the dataset, we resort to a comparable time span of 8 s ($M = 16$ prior windows) for problem i. Moreover, we devise that $\alpha_p = 10\%$, $\alpha_v = 10\%$, and $\beta = 20\%$, i.e., the highest and lowest 10% throughput samples during a session are considered peaks and valleys, respectively, and a sample with an inter-variation surpassing 20% is regarded as an abrupt change. We make our data and model publicly available[3].

To provide contextual insights, we illustrate traffic patterns from 20 randomly selected sessions in Figure 4.2, which depicts 3 sets of ECDF plots. The leftmost figure (4.2a, the ECDF of throughput values) demonstrates that nearly all values exhibit a steep ascent in the middle, gradually tapering into narrower tails for both ultra-low and high values, despite quantitative differences among traffic. The middle figure (4.2b, the percentage variations between successive throughput samples) reveals that the majority of inter-variations are below 20%, and in fact, 64.9% and 84.2% are lower than 10% and 20% for all traffic, respectively. Both observations suggest a generally and globally stationary evolution of traffic throughput, highlighting the significance of comprehending traffic extremes, and further rendering their prediction an interesting, substantial endeavor. Furthermore, we also investigate the

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



(a) Throughput values for individual sessions.

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

(c) Time span of 2048 packets.

Fig. 4.2 Traffic patterns of 20 randomly selected video-teleconferencing sessions.

duration of each set of 2048 packets for individual sessions in the rightmost figure (4.2c). As mentioned earlier, the average elapse is 7.6 s, but traffic exhibit different characteristics, whereby most sessions share a similar pattern with a duration of 3-10 s, as indicated by the rapid ascent in the ECDF, and a few manifest peculiar patterns, exemplified by either relatively uniform or irregular distributions. These observations actually conform to reality, where a video-call could be either active or quiescent with different frequencies of packets exchanged. And these also drive the selection of $N = 2048$ packets, as we strive to include the multifarious aspects of traffic transmitted in RTC, regardless of the degree of activity, which leads to a judicious trade-off between including extraneous, excessive information beyond the target and having insufficient packets.

4.3 Methodology

Our proposed DL framework *DeX* is composed of three components: a packet selection module, a feature extraction block, and a multi-task learning pipeline, as elucidated in Figure 4.3. Overall, we adhere to a moving window prediction, considering all $N = 2048$ preceding packets with three RTP elements as raw input, while learning an optimal subset as actual features during training. The feature extraction process is followed by a multi-task learning paradigm that incorporates various loss functions to optimize the prediction of traffic extremes.

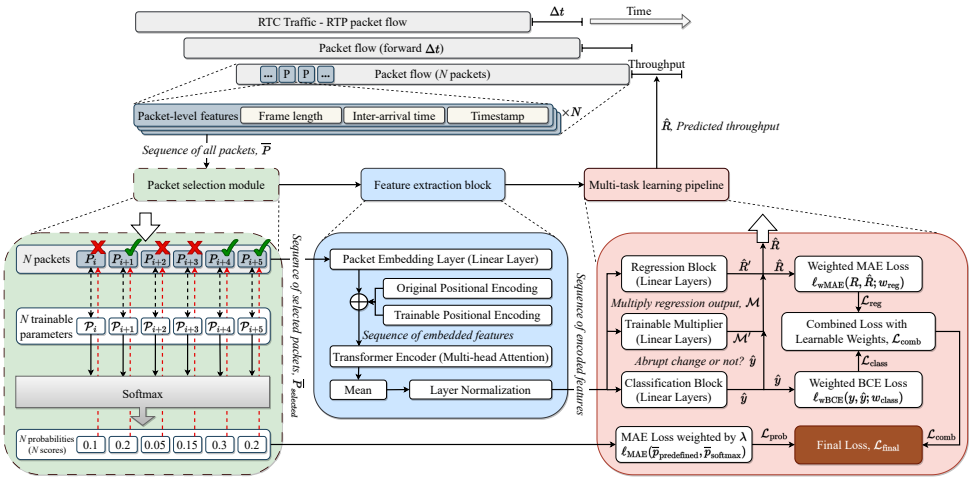


Fig. 4.3 Workflow, model architecture, and training strategy of *DeX*.

Packet selection module

This component reduces the input quantity while maintaining the performance by selecting an optimum portion out of all considered packets. This is particularly crucial for RTC services, which often encounter computational and temporal constrains, but simultaneously require low-latency real-time responses. On the one hand, RTC applications need swift, recurrent processing of media data and execution of various algorithms, which intensifies the computational demand. On the other hand, the delay-sensitive inherency of RTC naturally necessitates minimal time consumption of any intermediate process. Otherwise, escalated delays and synchronization discrepancies may emerge, thereby diminishing the QoE. Hence, we endeavor to curtail input packets to downsize the model complexity, enhancing memory efficiency and fostering a more computationally and temporally effective scheme.

For the sake of packet selection, we apply a straightforward logic, randomly initializing $N = 2048$ trainable parameters ($\mathscr{P}_i$), and assigning each to a packet (P_i). These parameters are then passed through a Softmax function to compute 2048 probabilities ($\overline{p}_{\text{softmax}}$) to opt for packets based on their values — the $W = 1024$ selected packets correspond to the highest 1024 probabilities[4]. Consequently. the model

[4] $\overline{P}_{\text{selected}} = \{P_i \mid \sum_{i \neq j} \prod_{i \neq j} (p_{\text{softmax},i} > p_{\text{softmax},j}) > 1024\}, \forall i, j \in [1, 2048]$. The outputs of Softmax are called probabilities simply because of the naming convention, and thus do not imply a stochastic process but are treated as deterministic scores for packet selection.

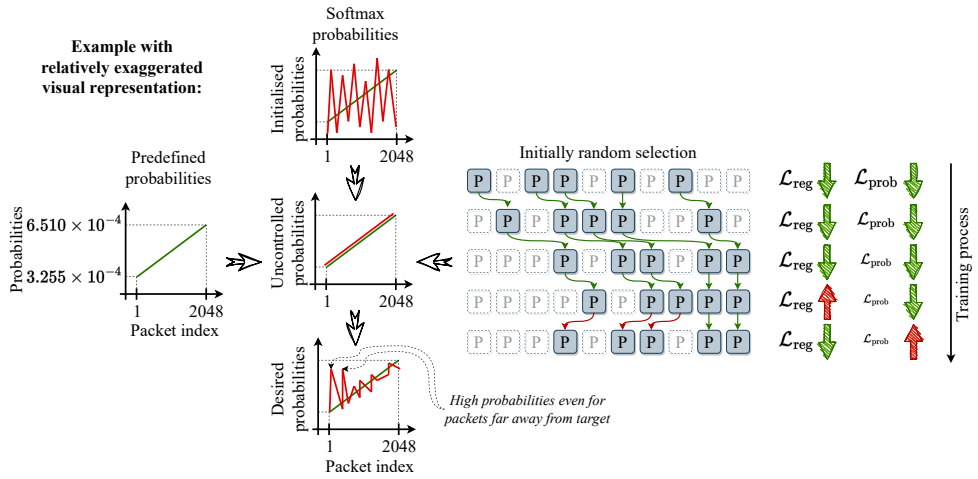


Fig. 4.4 How packet selection module works (note that $\mathscr{L}_{reg}$ in the figure is just a nominal representation instead of the actual regression loss in Equation 4.6).

learns and refines the trainable parameters in a way such that the derived probabilities (scores) are optimized to select the most suitable packets for the regression task.

However, this procedure is detached from the computational graph and the packet selection is not involved in the gradient flow, as no computations are triggered in the last step. To address the issue, we enforce the computational process by pre-defining a probability distribution ($\overline{p}_{\text{predefined}}$) to compare the output probabilities of Softmax. Meanwhile, we presume that the potentially optimal selections are the most recent 1024 packets — the closest ones to the target (current) sample. This hypothesis stems from the nature of time-series problems, where temporally proximate samples ought to reflect the latest trend in evolution and encompass the most salient features. Differently, packets other than those proximate to targets might also be critical and informative owing to the domain-specific irregularity of packet sequence in our specific problem. In this context, the preset probabilities (scores) serve as a guide to steer the learning process plus the selection towards the hypothetically optimal (closest) packets, while still tolerating a certain degree of flexibility to uncover potentially important packets located far from the targets.

To this end, we devise a simple predefined and monotonically increased linear distribution[5] as follows:

[5]At the first glance, such a distribution may appear somewhat arbitrary, but, by fine-tuning the hyperparameters, it is, in any case, possible for the NN to automatically cherry-pick beneficial packets, reaching optimal performance, irrespective of the initial distribution chosen. Therefore, the adopted distribution is inconsequential, as long as it satisfies our requirement. That is to say, the predetermined

$$\begin{aligned} \overline{p}_{\text{predefined}} &= [p_1, ..., p_i, ..., p_{2048}] \\ &\text{s.t. } p_i < p_{i+1},\ \sum p_i = 1,\ p_i = a \cdot i + b,\ i \in [1, 2048], \end{aligned} \tag{4.4}$$

where p_i is the predefined probability (score) for the i^{th} packet, and the summation of $\overline{p}_{\text{predefined}}$ equals 1, as the output of Softmax sums to 1. The term $a \cdot i + b$ indicates a linear relationship (details are in the next section). A visual example is portrayed in Figure 4.4, and the parameters, $\mathscr{P}$, are trained to produce Softmax probabilities that converge to the predefined ones, accomplished by computing the Mean Absolute Error (MAE loss function, $\ell_{\text{MAE}}(\cdot)$) between them:

$$\begin{aligned} \mathscr{L}_{\text{prob}} &= \ell_{\text{MAE}}(\overline{p}_{\text{softmax}}, \overline{p}_{\text{predefined}}) \\ \text{with } \overline{p}_{\text{softmax}} &= \text{Softmax}(\mathscr{P}_1, ..., \mathscr{P}_i, ..., \mathscr{P}_{2048}), \end{aligned} \tag{4.5}$$

in which, the loss of probability $\mathscr{L}_{\text{prob}}$ represents one of the optimized targets subjected to minimization. In consequence, the module operates in a manner that diligently forces the packet selection towards the target. Nevertheless, another problem arises — it always ends up with the closest packets because of the consistent gravitation towards the hypothetically optimal selection[6], diverging from our original goal of discovering informative packets in distance. To mitigate such an aggressive selection, we modify the MAE loss $\mathscr{L}_{\text{prob}}$, introducing a weighting factor, λ, to reduce its impact, refraining from permanently choosing the proximate packets (as illustrated in the right part of Figure 4.4). As the training process unfolds, both losses of regression and probability decrease synchronously, until reaching a certain point, where the packet selection is deemed non-optimal, leading to an upswing in the regression loss. Simultaneously, the MAE loss of probability continues to decrease, further abating its influence thanks to the weight, λ, and when the regression loss escalates because of certain selection, the selection revisits a prior relatively optimal position, given that the loss of probability exerts mere impact on the final loss. In consequence, the selection process stabilizes or hovers around the optimal locations, primarily due to the dominance of the regression loss.

probabilities function as a reference, and although different references will indeed influence the learning process and trainable parameters distinctively, the ultimate optimized objective of regression remains invariant, compelling the model to eventually ascertain the advantageous packet selection, regardless of the convergence trajectory.

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

As a result, we optimize the packet selection throughout the training process, significantly reducing the input quantity for downstream tasks. Importantly, the packet selection module will be discarded post-training without further perplexing the model, and from then on, we can directly channel the packets at already-known optimal positions into the next component.

Feature extraction block

We partially incorporate a Transformer model to systematically condense packet series, because packet flows resemble token sequences in NLP, which has been revolutionized by the groundbreaking game changer — Transformer. It enables our framework to demonstrate robust proficiency in learning the dynamic network nuances. We aim to harness the innate capabilities of the multi-head attention mechanism to automatically and intuitively apprehend the endogenous correlations interlacing the packet-level features and traffic throughput.

Firstly, we inject the sequence of selected packets, i.e., each set of the 3 features, into a packet embedding layer (linear layer) to create embedded features. The NN is anticipated to learn an apt mapping from the primordial packet attributes to latent embeddings, thereby enriching traffic features. Secondly, while each packet has the timestamp indicative of its order, we still lack of positional information for other packet entities. We, therefore, adhere to the original Transformer model, implementing *sine/cosine* positional encoding. More importantly, we augment the architecture by introducing an additional trainable positional encoding constituting of learnable parameters for two main reasons: 1) learning optimal positional and domain-related patterns to improve the task-specific adaptability, given that the original one is fixed and particularly designed for NLP, and 2) supplementing probably absent insights caused by inconsistent packet selection, since the original one operates on continuous sequence without any interstitial gap in between. Thirdly, both positional encodings are superimposed on top of the embedded features, and the resultant sequence is fed into a single layer of Transformer encoder, a component frequently employed for sequence representation learning [166, 167], to generate encoded features. Notably, we opt not to implement the Transformer decoder or additional stacks of encoders to avoid increasing the model complexity. Finally, we calculate the mean value for each set of encoded features in the output sequence, distilling feature quintessence

and deriving the ultimate feature vector, which is subjected to layer normalization to standardize the condensed features and stabilize the numerical pattern.

Multi-task learning pipeline

We employ a multi-task learning strategy with multifunctional weights to increase the model sensitivity to traffic extremes. Besides the primary regression task, we incorporate two auxiliary learning blocks: a binary classification component and a trainable multiplier. The former block predicts and identifies whether the target throughput signifies an abrupt change, a feat deemed attainable due to the granular and domain-specific packet-level features that are often absent in conventional time-series scenarios. The latter block serves as a calibrator to either amplify or attenuate the regression output based on the classification outcome, adjusting the final predictions to better accommodate dramatic variations. Each block (task) comprises a 2-layer feedforward neural network (FNN), which takes the finalized encoded features as input and produces a scalar value. Consequently, the NN undergoes meticulous training, ensuring that it consistently satisfies normal value expectations, while also compensating occasionally for abrupt changes. On top of that, we implement learnable weights [168] to systematically and optimally combine losses generated by different blocks:

$$\begin{aligned} \mathscr{L}_{\text{comb}} &= e^{-w_1} \cdot \mathscr{L}_{\text{class}} + w_1 + e^{-w_2} \cdot \mathscr{L}_{\text{reg}} + w_2 \\ \text{with } \mathscr{L}_{\text{class}} &= \ell_{\text{wBCE}}(y, \hat{y}; w_{\text{class}}),\ \mathscr{L}_{\text{reg}} = \ell_{\text{wMAE}}(R, \hat{R}; w_{\text{reg}}), \\ \hat{R} &= \hat{R}' \cdot \mathscr{M},\ \mathscr{M} = \begin{cases} \mathscr{M}', & \text{if } \hat{y} = 1 \text{ (abrupt change)} \\ 1, & \text{if } \hat{y} = 0 \text{ (normal transition)} \end{cases}, \end{aligned} \tag{4.6}$$

where the combined loss $\mathscr{L}_{\text{comb}}$ is computed by melding the classification ($\mathscr{L}_{\text{class}}$) and regression ($\mathscr{L}_{\text{reg}}$) losses via learnable weights, w_1 and w_2. Moreover, both regression and classification blocks are involved with weighted losses during training phase. On the one hand, to address class imbalance (only roughly 16% of throughput samples are abrupt changes), the classification loss is calculated by the weighted Binary Cross Entropy (BCE) loss function $\ell_{\text{wBCE}}(\cdot)$, with elevated weights w_{class} assigned to the minority samples. On the other hand, the weighted Mean Absolute Error (MAE) loss function $\ell_{\text{wMAE}}(\cdot)$ is employed for regression, with larger weights w_{reg} granted to peaks and valleys to incentivize the model to discern such scenarios.

y and R are the ground truths of classification and regression tasks, respectively. $\hat{y}$ denotes the predicted label of the classification block, while $\hat{R}$ represents the final forecasted throughput, determined by modulating the regression output ($\hat{R}'$) with the intervention of the trainable multiplier ($\mathscr{M} = \mathscr{M}'$). When the classification indicates a normal transition ($\hat{y} = 0$), the multiplier remains neutral ($\mathscr{M} = 1$), thus leaving the regression output unaltered.

Finally, the final loss for the entire model is calculated as:

$$\mathscr{L}_{\text{final}} = \mathscr{L}_{\text{comb}} + \lambda \cdot \mathscr{L}_{\text{prob}}, \tag{4.7}$$

wherein $\mathscr{L}_{\text{comb}}$ is the combined loss yielded by the multi-task learning pipeline in Equation 4.6, and $\mathscr{L}_{\text{prob}}$ corresponds to the loss of probability in Equation 4.5, tweaked by the hyperparameter, λ. The second term is deemed a regularizer, imposing constraints on the learning process, that are instrumental in preventing the model from becoming overly dependent on proximal packets, and thereby nudging the model towards solutions that are not only effective on the primary task but also exhibit a level of adaptability when faced with varying packet selections.

Model development

DeX is developed using the *Pytorch* [169] framework and is trained on a single GPU of NVIDIA Tesla V100-16GB. The implementation details are listed in Table 4.1. As for the parameters for the predefined probabilities, a and b are not initialized randomly but derived based on the following procedure:

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

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

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

$$\begin{gathered}\overline{p}_{\text{predefined}} = \Theta \leftarrow \frac{\Theta}{\sum \Theta}, \text{with} \sum \Theta = 3071.5 \\ \Downarrow \\ \sum \overline{p}_{\text{predefined}} = 1,\ p_1 = \theta_1 = 1/3071.5,\ p_{2048} = \theta_{2048} = 2/3071.5.\end{gathered}$$

Table 4.1 Implementation detail of *DeX*

Parameter	Value
Learning rate*, η	10^{-3}
Size of feature embedding, $N_{\text{embedding}}$	32
Size of positional encoding	1024×32
Number of heads	8
Number of encoder	1
Number of neurons for FNN in encoder	512
Activation function in encoder	*ReLU* [170]
Number of layers for multi-task learning pipeline	2
Number of neurons of the 1^{st} layer for a task in pipeline	512
Number of neurons of the 2^{nd} layer for a task in pipeline	1
Activation in multi-task learning pipeline	*ReLU*
Training optimizer	Adam [171]
Batch size	16
Weight for peaks and valleys, w_{reg}	2.0
Weight for abrupt changes, w_{class}	6.0
Weight for loss of probability, λ	4×10^{-4}
Parameters for the predefined probability, $\overline{p}_{\text{predefined}}$	$a = 1.59 \times 10^{-7}$ $b = 3.25 \times 10^{-4}$

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

As a result, a and b are computed accordingly. In fact, the actual controllers governing the linearity of the distribution are the values of the head and tail in Θ, i.e., θ_1 and θ_{2048}, for which we refer to a simple initialization of 1 and 2. We reiterate that the ablation study validating different model components and predefined probabilities is not included in this thesis. Further details can be found in our paper.

4.4 Experiment

4.4.1 Experiment setup

We randomly partition the 71 *pcap* files (video-calls) into 3 independent groups (50, 10, 11) to construct training (355,651 samples of throughput), validation (62,193), and test (65,061) datasets. We calculate based on the training set the statistics of mean value and standard deviation, which are then used to standardize validation and test sets. For each sample, the traffic extremes are marked based on the setting and its own session. Consequently, the model is trained on traffic collected under unique conditions different from other datasets, aiming to derive a generalized solution and preclude data infiltration among traffic.

Table 4.2 Model summary

Category	**Model**
Naive baseline*	Moving Average (MA)[1] [172]
Adaptive filter	Recursive Least Squares (RLS)[1] [173]
ML method	Random Forest (RF)[1] regressor [174] XGBoost (XGB)[1] regressor [128]
DL method	Multi Layer Perceptron (MLP)[1] [175] Long- and Short-term Time-series network (LSTNet)[2] [176] Long Short-Term Memory (LSTM)[1,2] [132] N-BEATS network[1] [177]

* It calculates the average value of past throughput samples as the prediction.
[1] Problem *i*, univariate time series prediction.
[2] Problem *ii*, multivariate packet level prediction.

We extend the comparison to a wide range of domains, implementing multiple other technologies appeared in the literature, as listed in Table 4.2. Notably, problem *i* has a total of 7 models, while 3 models including *DeX* are developed for problem *ii*[7]. Furthermore, we evaluate the performance of each model across various dimensions, including overall traffic, peaks, valleys, and abrupt changes, by assessing 4 typical regression metrics: Mean Squared Error (MSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the coefficient of determination (R^2 score).

4.4.2 Experimental result

Table 4.3 outlines the results independently for overall traffic, peak values, valley values, and abrupt changes. For the overall performance in the first part, *DeX* outshines other solutions across all quantitative measures. Although certain models (e.g., RF, XGB, N-BEATS, and LSTNet) are ostensibly on par (e.g. N-BEATS achieves a nearly identical MAPE of 10.6%), they invariably falter in other metrics (e.g., RF's declined MAPE of 11.844%). Additionally, the MA baseline and RLS filter manifest inferior performance (e.g., MSE > 0.07 and R^2 scores < 0.9), indicating the incompetence of simple statistical tools.

When it comes to peaks as well as valleys in the second and third parts in the table, *DeX* significantly outperforms its counterparts. On the one hand, our model stands out in terms of peak values, boasting the most minimal errors (e.g., the only MSE beneath 0.17 and the sole MAE lower than 0.25) and a preeminent R^2

[7]The other two models leverage the nearest 1024 packets as features without packet selection.

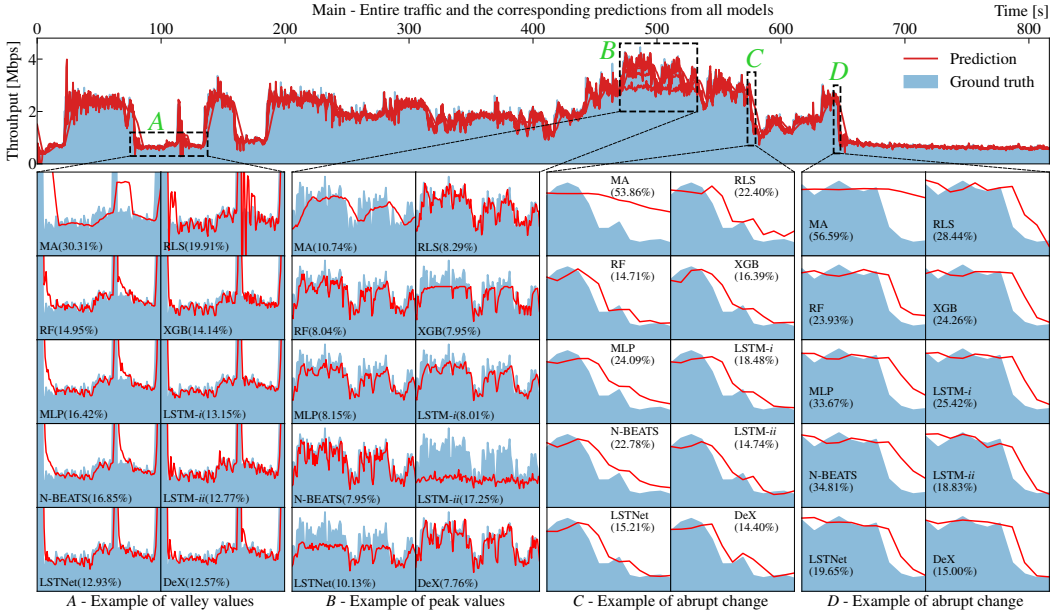
Table 4.3 Experimental result of all models regarding overall traffic, peaks, valleys, and abrupt changes.

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

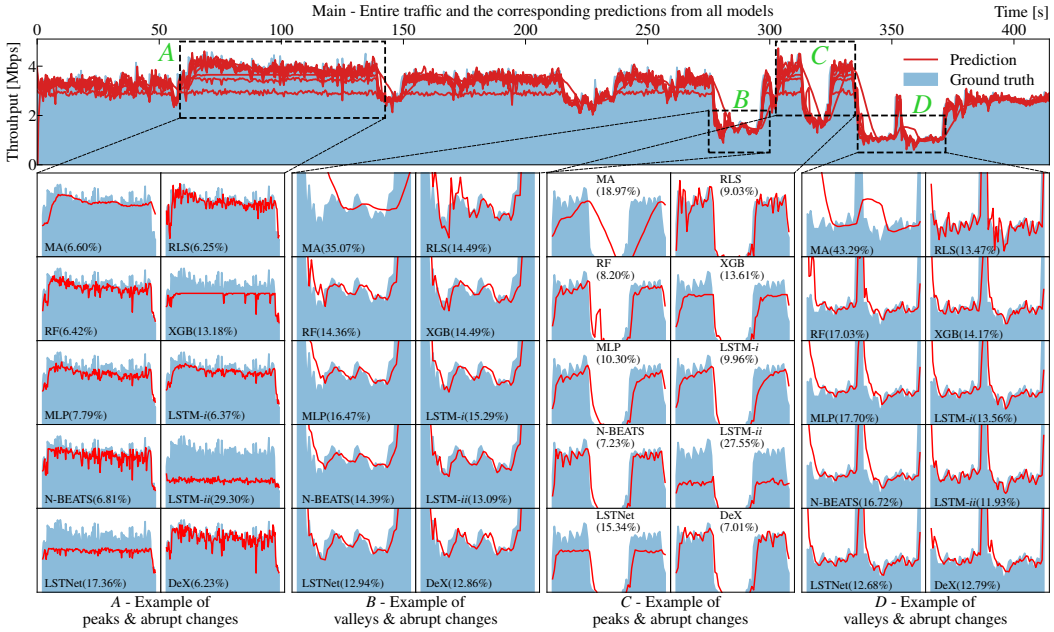
score (the only one exceeding 0.82), affirming its ability for precise forecasting rather than mere overestimation. On the other hand, the superiority becomes even more conspicuous for valley values. *DeX* delivers markedly less errors and decent coherence, as evidenced by a MAPE that is lowered by 1.285% and an R^2 score augmented by 0.0759 in comparison to their respective second-best values.

As for abrupt changes, *DeX* consistently excels other models, yielding optimal performance across the board (e.g., the only model with an MSE below 0.27). However, it remains inherently challenging to precisely predict such rapid and sudden transitions, resulting in relatively subpar performance regardless of the models, which originates from the intrinsic complexities entwined within the problem. Instantaneous fluctuations of RTC throughput could be induced by various factors, such as emergent traffic surges or network disruptions, which may not manifest prominently and timely in the packet flows received by end-users, thus rendering them elusive and difficult to be detected by the models. Yet, amidst this backdrop of hurdles, *DeX* proficiently harnesses granular packet-level insights in tandem with a meticulously designed model architecture, accommodating abrupt changes to a commendable extent.

To further illustrate the outcomes, Figure 4.5 showcases two examples in the test dataset, portraying the throughput time-series of ground truth and corresponding predictions, and highlighting traffic extremities in four sub-figures with MAE indicated



(a) Example traffic 1.



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

Fig. 4.5 Ground truth & predictions: traffic examples of throughput time series with highlights on peaks, valleys, and abrupt changes.

in parenthesis[8]. Evidently, all models can track the general traffic evolution, albeit to varying degrees of proficiency. *DeX* exhibits superiority characterized by the lowest error in most cases, while the MA baseline fails to quickly adapt the variations and RLS filter produces aggressive, volatile predictions. In particular:

- The first example in Figure 4.5a presents individual instances of critical traffic scenarios. For valley values (sub-figure *A*), *DeX* excels the others by deftly following the localized fluctuations, although performance disparities are relatively subtle with respect to certain models. Notably, some models, like RF and MLP, suffer from the sudden drop, resulting in unsatisfactory performance at the onset of troughs. Concerning peak values (sub-figure *B*), *DeX* is good at capturing and adapting to summits, whereas the majority of other models tend to generate underestimated predictions. Moreover, *DeX* also outperforms its peers regarding abrupt changes (sub-figures *C* & *D*), by swiftly and precisely accommodating the instantaneous declines. Remarkably, *DeX* can even anticipate the precipitate rise after a serial descents in example *C*, while yielding an optimal error, 3.83% lower the penultimate one in example *D*. However, each model exhibits a latency in response to the initial abrupt transformation, reaffirming the notion that it is barely possible to predict drastic transitions.

- The second example in Figure 4.5b depicts mixtures of traffic extremes. Regarding peaks coupled with abrupt changes (sub-figures *A* & *C*), *DeX* shows the smallest errors, whereas the tendency of underestimation reappears for most of other models. More importantly, *DeX* also demonstrates how swift and effective it is to transition from a prior precise prediction of critical values to another subsequent critical point in the opposite direction. Meanwhile, similar capability is also applied to valleys with abrupt changes (sub-figures *B* & *D*). An evident instance can be observed at the initiation and the culmination of the example *D*, where *DeX* responses rapidly to the sudden transitions, while all the others exhibit a lingering effect of either a tail for the descent from high values to valleys or a delay for the ascent from valleys to high values. Although *DeX* only ranks the third place in terms of MAE with a slight performance dip, all of the aforementioned irreplaceable merits still serve to validate its uniqueness and overall distinction.

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

Indeed, the prediction of traffic extremes ends up with underwhelming performance in contrast to the totality, which could emanate from the prevalence of relatively stable throughput samples, that limits the model's ability to effectively learn patterns associated with critical values, echoing the dilemma in imbalanced ML scenarios [126].

4.4.3 Model explainability

We herein showcase several examples regarding the packet selection module and the multi-task learning pipeline, elucidating their operational mechanisms.

Packet selection module

Figure 4.6 depicts the resultant packet selections in various scenarios of default 1024 packets, modified 1024 packets with different predefined probabilities, and less amount of 512 and 256 packets[9]. The uppermost 3 graphs portray 2 hypothetically optimal selections as baselines. The quartet of reddish figures below illustrate the smoothed distribution of derived selection probabilities, i.e., the average probability value for every 8 packets. The bottom 4 bluish figures elucidate the density of the corresponding packet selections, with darker regions signifying heightened packet density.

Apparently, the convergence towards the hypothetically optimal probabilities persists consistently across the various scenarios, despite minor divergences in alignment, and most selections are still allocated in temporal proximity to the prediction target, notwithstanding the broader distribution. The packet selection module enables *DeX* to strike a balance between optimally choosing input features and modulating the trainable parameters to generate Softmax probabilities towards the predefined ones, resulting in higher scores even for packets situated distally from the target. Moreover, the three cases of 1024 and 512 packets share a similar pattern of picking a portion of packets in the farside, while no remote packets are elected in the case of 256 packets, with even the 59 packets absent in the target's immediate vicinity (256 nearest ones) still positioned closed to the boundary. Of noteworthy concern, the packet selections in both instances of 1024 packets are different, which

[9]The descriptions and experimental results of the last three scenarios are in the ablation study and parametric analysis in the original paper.

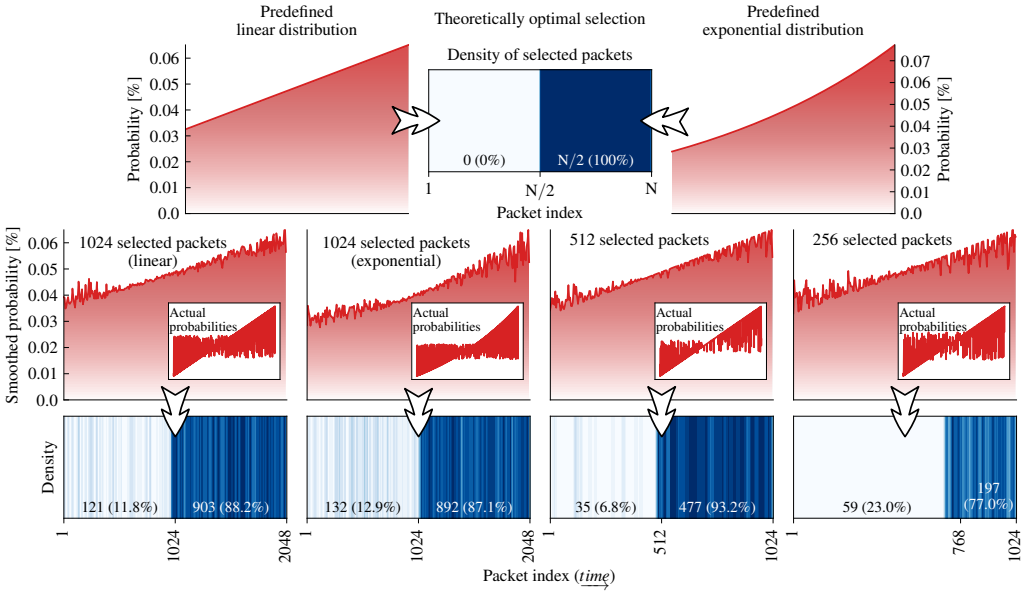


Fig. 4.6 Outcomes of packet selection. Notes: 1) The two-sided figures on top present the hypothetically optimal probabilities (linear or exponential), and consequently, the closest half packets ($N/2$) to the target are selected (top central figure). 2) The smoothed probabilities are for the sake of a clearer visualization, while the original ones are portrayed in the sub-figures. 3) The lowermost figures also indicate specific numbers of selected packets and their occupations, which are marked based on the target packet selection number, e.g., in the case of selecting 256 packets from a pool of 1024, 768 packets are allocated to the farside while 256 packets are in the nearside.

could stem from the fact that the module does not seek to pinpoint the absolutely and deterministically optimal locations of packets, but aim to uncover the practically effective packets that are suitable for a specific training trial.

Multi-task learning pipeline

To demonstrate the multi-task learning pipeline in improving the regression outputs for abrupt changes, we showcase 4 traffic throughput time-series of both ground truth and the corresponding predictions, with and without intervention of trainable multiplier, in Figure 4.7. Evidently, trainable multipliers can either magnify or compress the primordial regression outcomes, bringing them into closer alignment with the ground truth, especially in example 1, where the adjusted prediction is almost identical to the true value. Another more profound instance is demonstrated at the tail end of traffic example 4, where the throughput experiences a sudden increase, but the unaltered prediction veers in the opposite direction. Remarkably, the trainable multiplier discerns such an erroneous behavior and the abrupt surge, calibrating the prediction to attain a performance level deemed acceptable. Although

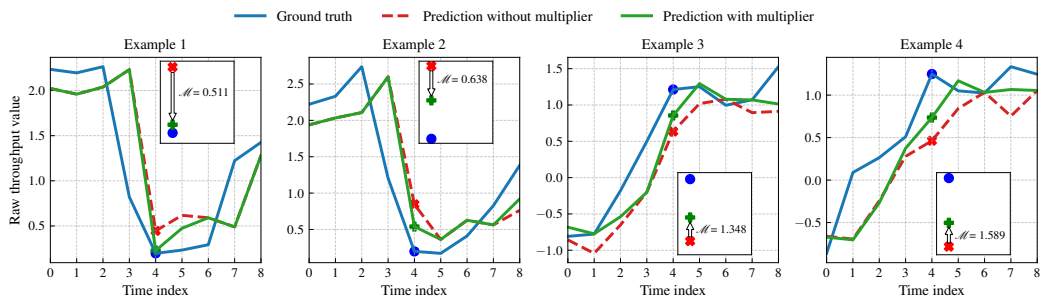


Fig. 4.7 Examples of how the trainable multiplier operates. Notes: 1) The negative throughput samples are raw values output by the NN, which are subsequently subjected to an inverse scaling to replicate the actual throughput. 2) The magnified sub-figures explicitly indicate the magnitude of the multiplier, that modulates the original regression output to align with the ground truth.

Table 4.4 Time consumption for *DeX* to make a prediction in CPU environments.

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

* The original model with 1024 selected packets.
** The modified model with 512 selected packets.

it is barely possible to response instantaneously to the very first abrupt change, the prompt and precise adaptations still demonstrate the effectiveness of *DeX*.

4.4.4 Model practicability

Due to the real-time nature and the restricted time window, the time consumed by *DeX* for predictions stands as a critical consideration. To provide context, we investigate the time needed for two model versions to execute a single prediction across three different levels of CPU environments devoid of GPU acceleration. According to Table 4.4, both models demonstrate acceptable consumption irrespective of the CPU, with 512 packet merely requiring 23 ms even in the worst case scenario. Hence, we envision sufficient temporal margin to employ possible optimization strategies, thereby validating the model practicability.

Moreover, we also provide insights into the model complexity regarding the number of parameters. In fact, the majority (93.3% and 80.8%) of parameters are

Table 4.5 Experimental result of *DeX* with less parameters.

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

occupied by the FNNs in the multi-task learning pipeline, which can be readily curtailed to enhance efficiency. In this regard, we reduce the size of the multi-task learning pipeline, decreasing the number of neurons in the first layer of the FNNs in all tasks from 512 to 256, and consequently obtaining a reduction of roughly 47% in the total parameter count. According to the evaluation result in Table 4.5, the trimmed model successfully achieves comparable performance to the original *DeX*, with only a slight decline for valley values, illustrating the potential and possibility for further improvement of model efficiency. Notably, the aforementioned analyses do not factor in any existing technical optimizations, including well-established pruning technologies [178] and efficient Transformer architectures [179], which could further bolster the model efficacy and practicality.

4.4.5 Supplementary material

In the original paper, we conduct a comprehensive ablation study considering five alternative scenarios: (i) using all 2048 packets as input features; (ii) training the model with the 1024 packets closest to the targets without packet selection; (iii) training the model with 1024 packets under a different predefined probability distribution for packet selection; (iv) replacing the Transformer encoder with an LSTM NN for feature extraction; and (v) removing the multi-task learning pipeline while retaining only the regression block. The results are in line with our expectations, thereby justifying our strategic choices and architectural design. Moreover, we also perform a parametric analysis under different configurations in terms of the input packet quantity (256 & 512), the prediction window duration (300 & 1000 ms), and the traffic-extreme threshold ($\alpha_p, \alpha_v = 15\%$ or 20% & $\beta = 10\%$ or 15%). Across all settings, our model consistently remains top-performing. See our paper for full details [10].

4.5 Takeaways

In this chapter, we aim at throughput prediction of video-conferencing traffic in RTC, framing a time-series problem and proposing *DeX* with a dedicated focus on traffic extremes. Our work can be summarized as follows:

- Traffic prediction, framed as a time-series problem, constitutes a critical endeavor bolstering various functionalities in RTC. However, existing works, despite differing levels of efficacy, fall short in prediction performance, especially for estimating extreme conditions, which is the typical long-tailed effect in statistics — a phenomenon often overlooked.
- We identify and define three types of traffic extremes, namely peaks, valleys, and abrupt changes, which, while relatively infrequent, are of particular importance in RTC traffic, symbolizing transmission limits and network oscillations.
- Our proposed model *DeX* consists of triple components: (i) a packet selection module to reduce model complexity and improve efficiency, (ii) a Transformer-based architecture with tailored designs to extract features and capture network "fate", and (iii) a multi-task learning paradigm to better accommodate traffic extremities.
- As demonstrated by the experimental results, *DeX* outstrips all comparative methods and deliver distinguished performance regarding traffic extremes, accurately forecasting traffic plateaus/troughs without over/under-estimation and promptly adapting to dramatic variations.
- Despite the enhanced outcome, it is still arduous to precisely predict traffic extremities, especially for their initial periods that are nearly unpredictable, because of the complicated nature of network systems and the elusive root causes of such events.

Returning to the research questions discussed in the background, packet-level information proves to be instrumental, and although traffic extremes pose significant challenges, they can be effectively detected using dedicated models.

Chapter 5

Fine and Multiple: Multi-objective QoS Prediction

This chapter presents our work: *One is Enough: Efficient Modelling of RTP Traffic for QoS Predictions in Real Time Communications* [11]. Serving as the objective and direct performance indicator, Quality of Service (QoS) has been extensively studied in RTC over a long history, including its analysis that attempts to unearth valuable insights from records, measurement/monitoring that focuses on system/application-level QoS retrieval/computation, estimation that approximates inaccessible metrics based on available information, prediction that anticipates future QoS for preemptive purposes, and beyond.

Building on the background in Chapter 2, this chapter aims to unify various QoS predictions into a single framework, achieving efficiency, comprehensiveness, and prediction precision at the same time. We propose *Oh*, asserting that *o*ne model is enoug*h* to forecast multiple QoS metrics of all concurrent RTP flows in a single shot. In the following, Section 5.1 introduces the background with necessary literature review and motivations. Section 5.2 expatiates the problem, Section 5.3 delineates our solution, and Section 5.3 presents the experiments. At last, Section 5.4 concludes with a discussion of key findings.

5.1 Background

QoS metrics are quantitative measurements used to assess network/application performance and reliability in RTC. They are indispensable components for stakeholders, including both consumers and network operators, enabling full-stack observability and providing critical feedback to inform decision-making strategies. Existing quality-reporting or runtime-monitoring technologies are limited to historical observations, which can quickly become outdated. Therefore, QoS prediction could play an instrumental role in strengthening network observability, enabling real-time network optimizations, and fostering proactive strategies [180, 181]. These predictions can anticipate quality degradation and network oscillations to allow for preemptive corrective actions and mitigate adverse effects, facilitating various network functionalities: providing beforehand valuable feedback for congestion control [99], enabling anticipated QoE mapping [72, 182], improving dynamic adaptive streaming or encoding [142, 143], contributing to SDN as key indicators for network management, such as bandwidth allocation [145], and more.

Substantial research has explored QoS prediction using ML/DL technologies in RTC [180, 181, 84, 183], in addition to literature that have been reviewed in previous chapters. The work in [184] examined various ML algorithms and developed a group of DL models like LSTM to predict QoS metrics including latency, received/sent packet counts, and download/upload traffic rates for real-time videos. The authors in [185] focused on video QoE estimation, using a bidirectional LSTM-based CNN to predict 5 key indicators such as average bitrate within 3 different scenarios. Multiple QoS metrics (e.g., jitter and packet count) are predicted in [186, 187], in which the former formulated a data mining problem, proposing a Support Vector Regressor (SVR) with a series of network features as input, and the latter one worked on the video quality under time-varying loads, introducing a load-adjusted learning scheme, wherein 4 conventional regression methods are utilized. Regarding individual metrics, more specialized models have emerged. For traffic prediction, [157, 188, 189] adopted various technologies to forecast bitrate. The first integrated LSTM into a RL framework as the bitrate predictor to optimize QoE, the second discussed an online ML system to predict future average bitrate and resolution for encrypted video streaming, and the third employed autoregressive integrated moving average (ARIMA) to predict the rate of upcoming packet sequences for video streaming in wireless networks. Moreover, delay prediction has also been

studied by [190] with LSTM NN, and packet loss prediction have been investigated in [191, 117] using multiple ML approaches, including tree models, artificial NN, etc.

However, a fundamental flaw persists: existing solutions, while effective, are task-specific, delivering a model/models that are limited only to single objectives of an individual QoS metric and RTP flow. First, each QoS metric has its own usefulness, and multiple metrics, rather than a single one, can operate either separately or cooperatively, contributing to numerous applications and unlocking additional possibilities. Second, during an RTC session, multiple RTP flows coexist concurrently, each originating from an individual sender, encapsulating a unique media type, and traversing possibly different network paths. Besides the fact that many QoS metrics are generally derived independently for each flow (e.g., the sequence number consistency for assessing packet loss is valid for each individual flow, and the bitrate of a single video stream is determined by the encoding conducted on each distinct source), per-flow QoS can provide nuanced insights into the network status and performance from varied granular standpoints, enabling both local and global optimization. More importantly, notwithstanding the disparities among entities, different QoS metrics are inherently interdependent, and different RTP flows may still share a portion of common sources and network conditions. These interconnected relationships can be factored in to optimize predictions, which, on the other hand, requires a unified consideration of multiple QoS metrics across all coexisting flows.

Unfortunately, in order to gain comprehensive insights into network performance through multiple metrics of all flows, one is compelled to perform sequential or parallel computing, which, though feasible, is inefficient and resource-consuming. Contemporary RTC applications often confront temporal and computational restrictions because of their real-time nature and constrained control over devices: 1) the execution of predictions must be prompt to provide sufficient time left for responsive operations, otherwise the QoS can be already observed, and 2) the computational resources available for predictions are limited and undetermined, whereas parallel flows may be dozens. To this end, our objective is to develop a model capable of efficiently predicting unlimited metrics and flows in a single shot, without consuming excessive time and extra computational threads.

Fine and Multiple

Having explored various aspects of QoS in RTC through different ML/DL model developments and specific focuses, we progressively recognize the limitations of previous works, which lays the groundwork for the innovations presented here, where we continue the usage of "fine"-grained packet-level features[1], but shift targets from single and limited to "multiple" and comprehensive. To the best of our knowledge, this work represents the first endeavor to efficiently and simultaneously predict multiple QoS metrics on a per-flow basis. In essence, we propose a single, integrated, and versatile alternative that can replace (initially partially, and ultimately entirely) existing approaches, circumventing the redundancy and incompatibility associated with implementing several standalone prediction techniques.

5.2 Problem statement

5.2.1 Problem formulation

Our objective is to leverage previous packets to predict multiple near-future QoS metrics for each of concurrent RTP flows. At a time instant t and for RTC traffic with M flows, we predict N QoS metrics per flow ($M \times N$ metrics in total) in the ensuing time window of length Δt, based on the preceding L packets for each flow ($M \times L$ packets in total) and K packet-level features for each packet. We formulate an irregular multivariate multi-objective time-series problem as:

$$\hat{\mathbf{Q}}^t = f(\mathbf{X}^t), \hat{\mathbf{Q}}^t \in \mathbb{R}^{M \times N}, \mathbf{X}^t \in \mathbb{R}^{M \times L \times K}, \tag{5.1}$$

in which the predicted QoS matrix in the window from t to $t + \Delta t$ is $\hat{\mathbf{Q}}^t$, whose elements $\hat{\mathbf{Q}}^t_{m,n}$ represent the n^{th} type of QoS metric for the m^{th} flow. The feature matrix $\mathbf{X}^t$ comprises packets precedent to time t, where each entry, $\mathbf{X}^t_{m,l,k}$, denotes the k^{th} packet-level attribute of the l^{th} preceding packet in the m^{th} flow. We aim to develop a model that learns a function $f(\cdot)$ to map input features to the estimated

[1] For the sake of efficiency, another key advantage of relying on packet-level features is the minimal effort required for feature extraction, which enables a streamlined end-to-end prediction directly from raw packets without intermediate procedures.

QoS metrics that converges closely to ground truths, $\mathbf{Q}^t$. In our work, we target N=4 QoS metrics:

1. **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.

2. **Average jitter** ($\hat{\mathbf{Q}}^t_{m,2}$=$\overline{R}_{jitter}$) — the mean value of all packet jitters, each computed according to the protocol[2].

3. **FPS** ($\hat{\mathbf{Q}}^t_{m,3}$=R_{fps}) — the number of video frames per second for video flows, computed by counting the unique RTP timestamps in the time window.

4. **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 a flow.

In summary, we formulate a combination of 3 regression problems and 1 binary classification problem. We follow a moving-window schema, forwarding a step of Δt each time after a prediction, and assimilating new packets with old ones discarded if unneeded.

Initially, we refer to a duration of $\Delta t = 500$ ms for the predicted time window, and define that 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 the start of the time window (t). Simply put, we ignore inactive or interrupted traffic flows. Moreover, we resort to $K = 7$ packet-level attributes as features:

1. x_1, absolute inter-arrival time (IAT) — the interval between the arrivals of two successive packets.

2. x_2, relative IAT — the time elapsed from the first packet in the L-packet series to a given packet.

3. x_3, lead time — the duration between a packet's arrival and the start of the predicted window (time t).

[2]The calculation of jitter in RTP specification differs from the general network jitter definition (variance of inter-arrival time). It requires the clock rate (packet sampling frequency) of the RTP flow with a particular payload type, which is accessible in our session logs. If unavailable, the clock rate can also be inferred from the difference in RTP timestamps between consecutive packets.

Table 5.1 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

4. x_4, packet length — the total size of an entire packet, including both header and payload.

5. x_5, RTP timestamp — a 32-bit value that provides specific functions such as aiding to maintain the correct timing order of incoming packets.

6. x_6, RTP marker — a 1-bit flag in the RTP header indicating significant events, such as frame boundaries.

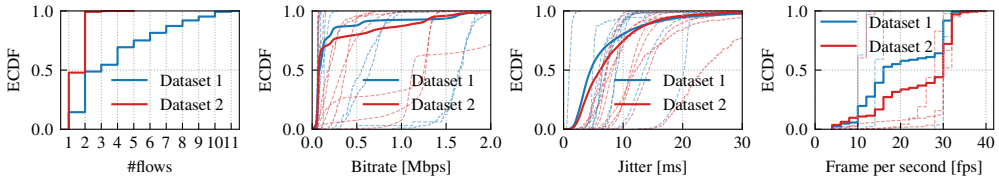
7. x_7, sequence difference — a binary label that is 0 if the sequence number pattern between two consecutive packets is regular, and 1 otherwise.

The first 3 features capture both local and global timing behaviors, the fourth feature directly reflects the volume of transmitted bits in the past, and the remaining ones are RTP-event factors, bringing in potential influence arising from specific network events. Collectively, they incorporate manifold facets from temporal, spatial, and RTP-related components.

5.2.2 Dataset

The traffic employed in this work is divided into two parts: one collected from the two major VCAs, and one collected using the other applications, as mentioned in Section 2.1.2. Table 5.1 lists the details[3]. Both datasets are constructed by creating multivariate time series of continuous time windows for each session. Specifically,

[3]The dataset associated with this work is not currently available.



(a) Number of flows per time window.

(b) Bitrate of individual RTP flows.

(c) Jitter of individual RTP flows.

(d) FPS of individual video flows.

Fig. 5.1 Traffic patterns of both datasets (light-colored, dashed lines in the 3 sub-figures on right-side correspond to 10 randomly selected individual flows for each dataset).

we identify active flows for each window, and then for each flow, we extract preceding packets as features, and calculate corresponding QoS metrics in the window. In consequence, each time window generates one or more samples (M active flows), with each sample composed of 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 $128 \cdot M$ packets). On the other hand, to comprehensively evaluate the performance, our problem can also be framed as a traditional regular time-series prediction task, using historical observations as features, e.g., QoS metrics from previous time windows. In our case, 128 packets merely occupy 2.97 ± 0.06 seconds (roughly 6 windows), which is generally inadequate for typical time-series problems. Hence, we extend to 10 s (20 500-ms windows), crafting 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). As for the classification task of packet loss, we refer to our previous work in Chapter 3, creating datasets with traffic statistical features and formulating a multivariate problem. In a word, for each session, we produce two similar datasets with identical prediction targets but different features of either preceding packets or historical information.

To provide context, we present the traffic patterns via 4 ECDF plots in Figure 5.1. Figure 5.1a illustrates the number of concurrent RTP flows in each time window for both datasets. Apparently, dataset 2 exhibits a significantly lower number of parallel flows, with nearly half of the windows containing only a single flow. This is attributed to the fact that the traffic was collected for the sake of a basic analysis of VCAs [1], and thus the focus was on gathering data from various apps with minimal participant

involvement, resulting in fewer streams generated. With this in mind, we use dataset 2 as an additional test set only to provide supplementary performance evaluation. 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 approximately 4 concurrent flows per time window, with only around 10% of the windows engaging 1 single flow, further underscoring our motivation. Moving forward to the patterns of the predicted targets, Figure 5.1b, 5.1c, and 5.1d outline the three QoS metrics for both datasets, with both the entirety and several (10 for each dataset) 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 to the standard setting of common video encoders in VCAs. 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, 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.

5.3 Methodology

5.3.1 Background

We first elaborate on the necessary background before delving into the topic. In generally, two key concepts underpin our solution:

- Knowledge distillation: The core idea is to achieve elevated efficiency in terms of time consumption without compromising prediction accuracy, by transferring knowledge from the best-performing, computationally intensive model to a simpler yet more efficient one. Knowledge distillation is a technique frequently adopted in ML/DL domains for this purpose, and it typically involves two models: the teacher, a competent pre-trained large model, and the student, a sufficiently smaller model that learns from the teacher by, for example, incorporating auxiliary losses. The complexity of the teacher model hinders efficiency, and thus it is 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.

- Attention-based LSTM: Following the success and conception in previous works, it becomes natural to stay with Transformer in the teacher model, and we harness the core component instead of the intact architecture, i.e., the multi-head self-attention mechanism, to fathom the inherent and intricate correlations interweaving packet-level features, RTP flows, and QoS metrics. The attention mechanism serves a pivotal role in sequence modelling, involving a series of computation of the query (Q), key (K), value (V) matrices derived from the input sequence:

$$\text{Attention}(Q,K,V) = \text{Softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{5.2}$$

  However, 2 inherent issues persist for our specific problem. First, the vanilla Transformer supports only a fixed input length, whereas we deal with an unfixed number of RTP flows, which necessitates architectural customization[4]. Second, it has been argued that the applicability of Transformer to time-series problems remains contentious [193, 194]. To this end, rather than simply using Transformer, we opt for a hybrid approach, integrating attention with LSTM, thus combing the strengths of both architectures. Although LSTM, which demonstrates efficacy in time-series tasks [195, 196], has lost its competitiveness against other more sophisticated ensemble models, it remains an auspicious option as an individual component for sequence modelling [197, 198], and has been widely proved to be effective when in conjunction with attentions [199, 200]. Notably, we first apply attention to refine packet sequence and then inject them into LSTM layers, unlike most approaches that leverage attention mechanism to assign weights to LSTM's output.

[4]Notice that while certain Transformer variants can handle unlimited input [192], they are not applicable to our case.

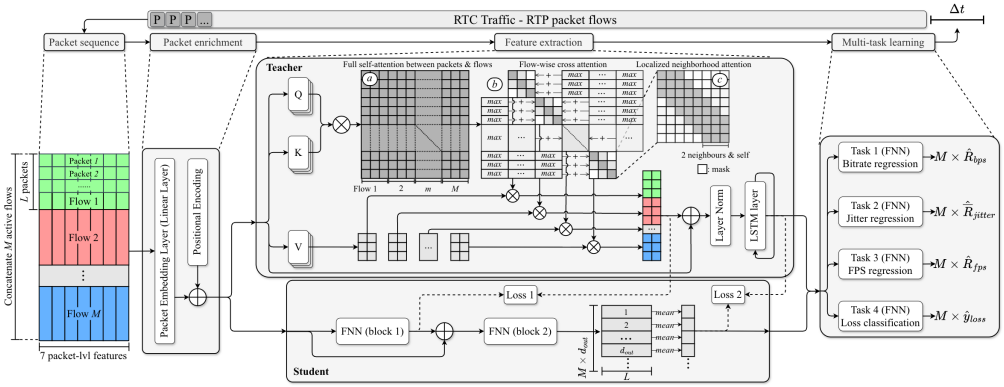


Fig. 5.2 *Oh*: model architecture.

5.3.2 Proposed framework

Overview

We propose *Oh*, a DL framework that incorporates teacher-student scheme, as depicted in Figure 5.2. Generally, we first extract preceding valid traffic flows, passing packet-level features to the packet enrichment process. The output is subjected to a feature extraction module responsible for sequence (RTP flow) modelling and then a multi-task learning block to predict various QoS metrics.

Packet enrichment

For a targeted time window, we arrange active RTP flows with their respective preceding packets in chronological order. All valid flows are concatenated, forming a $128{\cdot}M{\times}7$ matrix as the raw input for the packet enrichment process. Next, we implement a packet embedding layer (a linear layer) to convert 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. Again, we follow the good practice of the original Transformer, stacking *sine/cosine* positional encoding on top of the embedded features to infuse positional information. Consequently, we derive the traffic sequence of embedded packet features, and the whole process is needed for both teacher and student.

Teacher model — Preliminary

We devise a dual attention architecture named the cross and neighborhood multi-head self-attention, aiming to eliminate the restriction on input quantity (variable M) and correlate concurrent RTP flows. The first step of multiplying the query and key matrices ($W{=}QK^T$) in Equation 5.2 forms the attention weight matrix[5]. As exemplified by M flows with $K{=}3$ packets for each (the top-left large grayish matrix marked by a in Fig. 5.2), the full (regular) self-attention yields a square matrix (attention weights) with dimension $3M{\times}3M$. Specifically, $W_{3M\times3M}(1{:}3,1{:}3)$ represents the attention weights within flow 1, e.g., $W_{3M\times3M}(1,1)$ and $W_{3M\times3M}(1,2)$ 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. On the other hand, $W_{3M\times3M}(1,4{:}6)$ is the attentions paid by the 1^{st} packet in flow 1 to packets from flow 2, and similarly, $W_{3M\times3M}(1,3m{-}2{:}3m)$ shows the attentions paid by the same packet to m^{th}-flow packets. Accordingly, entries in $W_{3M\times3M}(1{:}3,4{:}3M)$ indicate the attention weights assigned by flow 1 to other flows.

Teacher model — Flow-wise cross attention

As portrayed by the collection of matrices marked by b, the 3-by-3 diagonal sub-matrices 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 other words, 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. Hence, by treating packets from all flows as a collective input while processing them separately for attention weight computations, it is viable to scale up to a theoretically infinite amount of concurrent RTP flows, thereby overcoming the obstacles associated to input limitations. Furthermore, the flow-wise cross attention also potentially discovers relationships among concomitant flows, which is achieved by augmenting the targeted diagonal matrices with refined information from other flows (off-diagonal square matrices). Particularly, the attention weights between different flows (flow-

[5]For a matrix W, let $W(i,j)$ denote 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.

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, adding 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 preserve only the diagonal sub-matrices, discarding off-diagonal ones completely. In summary, the flow-wise cross-attention prioritizes and confines packet interactions to those localized within the same flow, while still taking into account a certain degree of influence from other flows.

Teacher model — Localized neighborhood attention

We further reform each sub-matrix of attention weights on the diagonal, inspired by [201]. In the context of the $L = 128$ packets per flow that may span over several seconds, we argue that the correlations are substantial between adjacent packets, while minor between temporally distant ones. For RTC, where network dynamics oscillate drastically, characteristics of, for instance, early and late packets with a relatively long duration in between may differ significantly, thereby demonstrating 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 learn such a pattern at a risk of failure. An example for a flow (sub-matrix) of 8 packets and a neighborhood degree of k=2 is depicted by the rightmost zoomed-in 8-by-8 matrix with gray and white mixed (marked by c in Fig. 4.3). We apply masks[6] to redundant entries (distant attention weights), only retaining the diagonal region. In other words, the kept 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 dispensable. Consequently, the localized neighborhood attention ensures the focus of the model

[6]Masking is a common technique in DL used to ignore certain values. In our case, it sets an attention weight to $-\infty$, so the corresponding Softmax output is approximately 0.

on closely related packets, mitigating 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. The finalized attention weight for each individual flow then multiplies its corresponding value matrix (V) to obtain the attention score (a $128\times N_{embedding}$ matrix), and we concatenate scores of all M flows to derive the final output, a $128M\times N_{embedding}$ matrix. Subsequently, we apply a residual connection to add the original input (the sequence of enriched packet features) on top of the result of our customized attention, whose values are stabilized via layer normalization. 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.

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 comprising an identical FNN and corresponding to one of the targets. 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, we once again leverage learnable weights [168] during the training phase, dynamically adjusting the importance of different tasks and systematically blending losses:

$$\begin{aligned}
\mathscr{L}_{\text{teacher}} &= e^{-w_1}\cdot\mathscr{L}_{\text{bps}} + w_1 + e^{-w_2}\cdot\mathscr{L}_{\text{jitter}} + w_2 + e^{-w_3}\cdot\mathscr{L}_{\text{fps}} + w_3 + e^{-w_4}\cdot\mathscr{L}_{\text{loss}} + w_4 \\
\text{with } \mathscr{L}_{\text{bps}} &= \ell_{\text{MAE}}(R_{bps},\hat{R}_{bps}),\ \mathscr{L}_{\text{jitter}} = \ell_{\text{MAE}}(\overline{R}_{jitter},\hat{\overline{R}}_{jitter}), \\
\mathscr{L}_{\text{fps}} &= \ell_{\text{MAE}}(R_{fps},\hat{R}_{fps}),\ \mathscr{L}_{\text{loss}} = \ell_{\text{wBCE}}(y_{loss},\hat{y}_{loss},w_{\text{class}}),
\end{aligned} \tag{5.3}$$

where $\mathscr{L}_{\text{teacher}}$ is the final loss for the teacher model, computed by combining individual losses of tasks ($\mathscr{L}_{\text{bps}}$, $\mathscr{L}_{\text{jitter}}$, $\mathscr{L}_{\text{fps}}$, $\mathscr{L}_{\text{loss}}$) 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 address class imbalance.

Student model

The feature extraction component in *Oh*-student has a substantially simplified architecture with solely fully connected layers, while other parts are unchanged. We conform to the feature-based paradigm [202], fetching knowledge from intermediate layers of the teacher model. Specifically, we construct 2 FNN blocks to learn the feature representation from the attention mechanism and the LSTM NN, respectively. First, the enriched packet sequence of each flow is fed into block 1, whose output is directly compared with the final attention score produced by the teacher model. Similarly, a residual connection is applied, superimposing the block's output back on the original input sequence. Next, the sequence is processed by block 2, generating a matrix with dimension of $M \cdot d_{out} \times L$, where d_{out} represents both the LSTM hidden state size in the teacher model and the dimension of the last layer in the FNN, 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 subjected to contrast the LSTM output in the teacher model.

Student model — Loss function

The multi-task learning block in *Oh*-student remains the same, with an exception of 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. Thus, we minimize the MAE losses between these entities alongside the original task-related losses:

$$\begin{aligned} \mathscr{L}_{\text{student}} &= \mathscr{L}_{\text{teacher}} + e^{-w_5} \cdot \mathscr{L}_{\text{b1}} + w_5 + e^{-w_6} \cdot \mathscr{L}_{\text{b2}} + w_6 \\ \text{with } \mathscr{L}_{\text{b1}} &= \ell_{\text{MAE}}(out_{attention}, out_{b1}),\ \mathscr{L}_{\text{b2}} = \ell_{\text{MAE}}(out_{LSTM}, out_{b2}), \end{aligned} \tag{5.4}$$

in which $\mathscr{L}_{\text{student}}$ is the ultimate loss of *Oh*-student, derived by combing the loss of *Oh*-teacher ($\mathscr{L}_{\text{teacher}}$ from Equation 5.3) with the losses from the two FNN blocks ($\mathscr{L}_{\text{b1}}, \mathscr{L}_{\text{b2}}$) 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 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.

Model training

Table 5.2 lists the implementation details regarding hyperparameters. We adopt a three-stage 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 is frozen, serving as a pre-trained inference-only model to generate intermediate outputs. We subsequently abandon completely the teacher model during the third stage, continuing the training of the student model based solely on the four task-specific losses. In this context, it adheres to a self-learning paradigm, transcending the typical knowledge distillation scheme, 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, act 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.

5.4 Experiment

5.4.1 Experiment setup

As discussed previously, we construct 2 dataset packs, called dataset 1 and 2, with the same structure but different traffic sources. For dataset 1, we follow the same

Table 5.2 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* [203]
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* [171]
Batch size	8

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

strategy in previous works, randomly splitting traffic based on individual sessions (*pcap* files) 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, while the entire dataset 2 is used as another test set. The intention remains the same, where we ensure that *Oh* is trained on traffic manifesting distinctive conditions different from other datasets to avoid data leakage and to obtain a more universal and generalized solution.

To comprehensively evaluate the performance, we compare *Oh* against an array of ML/DL algorithms, either appearing in the literature or delivering SOTA performance in time-series problems. However, recent trends of long-term forecasting [204] and LLMs [205] are ill-suited to our context, certain DL approaches are tailored to canonical datasets that markedly diverge from our scenario [206, 207], and some time-series models do not align well with our specific problem [193, 208, 209]. Given the rapid evolution and sheer volume of deep time-series models emerging in the community, identifying the most appropriate solution for our problem is non-trivial. Therefore, we refer to [210], selecting and implementing several top-performing and applicable models. According to Table 5.3, 5 models are implemented for time-series features, while 6 are applied to packet-level features. Notably, an individual model is required for each task (QoS metric), meaning that 4 independent models are built for each comparative algorithm. Quantitatively, we still adopt the 4 evaluation metrics for regression problems: RMSE, MAE, MAPE, and R^2 score. As for the

Table 5.3 Considered models for comparison.

Category	Model
Time-series	Random Forest (RF) [174] eXtreme Gradient Boosting (XGB) [128] Multi-layer Perceptron (MLP) [175] Long Short-Term Memory (LSTM) [132] N-BEATS [177]
Packet-level	LSTM Long- and Short-term Time-series network (LSTNet) [176] Non-stationary Transformer (NST) [211] PatchTST [212] TimesNet [213] TSMixer [214]

classification one with imbalanced classes, we again only employ class recalls and the F1-score to measure both individual class accuracy and overall performance, as what we have done in previous works.

5.4.2 Experimental result

Quantitative results

Table 5.4 provides a summary for all models and both datasets. Overall, *Oh*-teacher delivers superior performance on dataset 1, albeit slight overfitting on dataset 2, and *Oh*-student achieves comparable outcomes, with even enhanced performance for dataset 2.

For dataset 1, *Oh*-teacher outstrips predominantly other competitive models and demonstrates exceptional performance especially for bitrate and FPS. As for average jitter, all 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, exclude the preceding jitter value for each packet, relying on models to discern underlying relationships between targeted jitters and considered packet features. We intentionally do so to avoid the overhead incurred by jitter computation, as jitter is derived iteratively for each packet. While this limits jitter prediction, we still advocate for our current approach instead of simply injecting jitter into packet features, since the primary goal is to improve efficiency. Even though, *Oh*-teacher still outperforms other packet-level models, exhibiting comparable performance with respect to time-series models, exemplified by the second-best MAE. Regarding

Table 5.4 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	Re.0	Re.1	F1
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.
[3] In the last column, Re.0 and Re.1 mean class recalls, and F1 is the F1-score.

packet loss predictions, *Oh*-teacher may not be the prominent choice at the first glance, but it reaches a balance between both classes, identifying most lossy time windows without compromising lossless ones, and thereby attaining a commendable F1-score. On the contrary, the top rank (XGB) suffers a 13% lower recall for class 1, and RF misclassifies excessive lossless time windows, yielding a subpar recall for class 0. Meanwhile, *Oh*-student inherits advantages of the teacher, managing to achieve comparable performance even with certain slightly better results.

When it comes to dataset 2, *Oh*-teacher fails to output the premier performance (though it remains among top ranks), manifesting marginal overfitting. This 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 deemed a minor issue, because 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

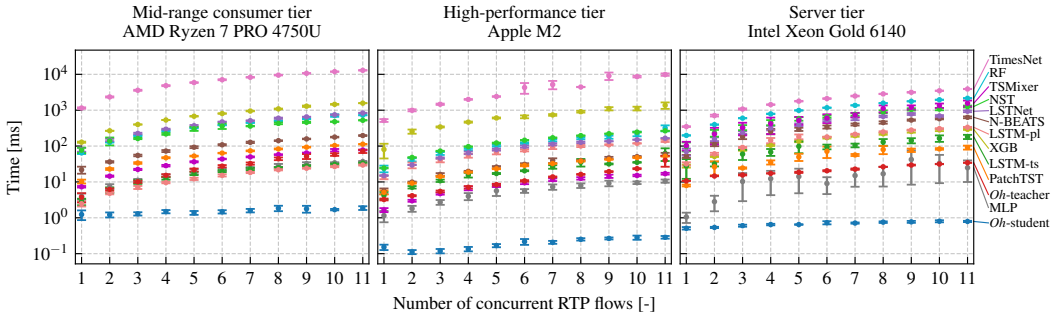


Fig. 5.3 Time consumption of different number of flows for all models in various computational environments.

alleviates the issue, producing outstanding performance, especially for bitrate and FPS. Although we still suffer from jitter prediction and a slight decline for packet loss, the overall performance remains satisfactory.

To summarize, *Oh*-teacher stands as the optimal candidate for knowledge distillation, while *Oh*-student maintains robust performance even with its simplicity.

Temporal overhead evaluation

Our primary objective lies on the model efficiency, and we thereby 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 concurrent RTP flows from 1 to 11, as in Figure 5.3.

All models experience a relatively linear growth (notice the logarithm scale) as the number of flows increases. Remarkably, thanks to the simplest FNN-only architecture and the mutual features for all tasks, *Oh*-student demonstrates substantially lower time consumption, with a negligible magnitude of 1 ms irrespective of computational environments. On top of that, the consumption does not scale up dramatically with 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 much more RTP flows. Moreover, while *Oh*-teacher is supposed to be temporally inefficient due to its sophisticated structure, it still generates relatively lower time consumption, owing to the simultaneous processing of input in one shot and the shared features, which unlocks the potentiality of utilizing *Oh*-teacher when performance is prioritized. Again, the performance herein might be further improved,

as we do not factor in any other potential optimizations including well-known NN pruning techniques [178] and other effective NN architectures [179]. 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 considering the 500-ms time window. When certain models (e.g., PatchTST, LSTM-ts, and MLP) yield acceptable time consumption, their prediction performance is 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 timely and promptly enact network optimization operations.

Generalizability

The two distinct datasets from ample video-call traffic may not offer exhaustive coverage for evaluating model generalizability, and thus, we also consider a random hour (around 200,000 samples) of the general RTP traffic collected at the campus router[7].

Both teacher and student models attain satisfactory performance for the majority of traffic (roughly 70%), while encountering abnormal behaviors in a minor subset (30%), as reported in Table 5.5. Given the broad coverage of RTC applications across the entire campus, the competitive performance achieved from abundant new traffic validates the model generalizability. Noteworthily, the abnormal traces, that experience significant performance decline, manifest peculiar patterns upon closer inspection. For example, the oddly high MAPEs (around 40% higher than generalized traffic) with 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 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. In a word, the anomalous traffic deviates from standard RTP implementations, fundamentally 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

[7]Notes: 1) As it includes the entire population and all incoming traffic of the university, an hour of traffic is deemed representative and voluminous. 2) It was not included previously because QoS prediction for all traffic at the ingress node is relatively less meaningful, and we thereby only aim to take a peek at the performance.

Table 5.5 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	Re.0	Re.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

underlying low-bitrate RTC applications generally do not necessitate sophisticated QoS predictions. Although performance degradation is often inevitable ascribed to the arduous challenges in model generalizability and domain shift, *Oh* still manages to deliver satisfactory results for numerous instances.

5.4.3 Supplementary material

The ablation study in the original paper demonstrates the necessity of the various model components and the teacher-student scheme by evaluating performance under several modified settings: employing a standard attention mechanism; restoring the vanilla Transformer encoder with a linear layer in place of the LSTM; discarding the Transformer architecture altogether; training the student model devoid of guidance from the teacher model; and training different QoS tasks independently. Furthermore, we elaborate on parametric analysis (i.e., model sensitivity) to inspect different packet quantities per flow ($L = 64$ or 256) and predicted window sizes ($\Delta t = 300$ or 1000 ms), with results demonstrating consistent performance. For further details, please refer to our original paper upon its publication.

5.5 Takeaways

In this chapter, we present our finalized unified solution for QoS prediction with a simultaneous focus on forecasting precision, comprehensiveness, fine granularity, and model efficiency. In particular, our findings/contributions can be summarized as follows:

- QoS prediction represents a fundamentally important task in RTC and has attracted substantial popularity in both academia and industry. Existing works

employ various technologies, including statistical tools and artificial intelligence, and focus on specific targets, including different individual QoS metrics or operational pipelines, which urges us to rethink the current scope of QoS prediction — since all QoS metrics and their predictions are critical, an ultimate futuristic RTC system ought to be exhaustive, incorporating as many various and detailed predictions as possible.

- A naive way is to implement and integrate off-the-shelf solutions, which is feasible yet impractical because of multidimensional issues related to temporal efficiency, computational constrains, operation incompatibility, system redundancy, etc. To this end, we visit such a dilemma that has been overlooked or deemed irrelevant, aiming to develop a versatile quasi-foundation model that unites QoS prediction while addressing unique challenges in RTC.

- In response, we propose *Oh*, adopting knowledge distillation for efficiency, designing a customized attention mechanism to capture multifaceted correlations, and employing multi-task learning to generate multiple QoS predictions. As verified by the experiments, our solution shows superior prediction efficacy while being capable of forecasting numerous entities, including four metrics for all concurrent RTP flows, within a negligible amount of time. 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.

- A key limitation of our work is the consideration of only four QoS metrics, which mainly originates from the dataset constraints, i.e., the unavailability of certain critical information. The traffic was collected using general-purpose tools to enable large-scale data acquisition rather than in a controlled environment. Nevertheless, given that the strong correlations among various network patterns can also be applied to other QoS metrics and our model relies solely on general packet properties instead of task-specific features, we foresee an "effortless" integration of additional task-oriented NNs to predict more metrics once complete and specialized datasets become available.

Returning to the research questions outlined in the background, multiple per-flow QoS metrics can be (and should be) predicted efficiently to provide comprehensive network observability.

Chapter 6

Theoretical to Practical: Bandwidth Estimation for Congestion Control

In this chapter, we discuss our work: *Debunking Reinforcement Learning for Bandwidth Estimation in Real-Time Communications: When a Simple Regressor Suffices*. As indicated by the title, this study revolves around two key concepts: reinforcement learning (RL) and bandwidth estimation (BWE). The former, usually referred to as deep RL (DRL), is a renowned learning paradigm designed to improve performance through interaction with real-world/simulated environments. The latter aims to probe the network bottleneck capacity (mainly) for the purpose of congestion control (CC).

Building on the background presented in Chapter 2, this chapter explores a simple regressor as an alternative to complex RL algorithms for CC in RTC. In particular, we revisit the current trend of utilizing RL for BWE, contending that RL is not strictly necessary, while BWE accuracy is the determinant, which renders a simple regressor enough. In the following, Section 6.1 presents the background of BWE for CC and the role of RL. Section 6.2 illustrates our observation that justifies the importance of accurate BWE. Section 6.3 introduces our proposal, followed by Section 6.4 that describes the experiment. Section 6.5 concludes the chapter with key takeaways.

6.1 Background

CC in common transmission protocols, primarily TCP, relies on a shared principle of regulating the sending rate to avoid excessive traffic. In contrast, RTP itself born without CC, depending instead on application-level functionalities to handle network congestion, whose underlying mechanisms are, however, fundamentally different from, for example, TCP, due to the operational vantage point/layer, the feedback granularity, the required traffic statistics, the latency/loss tolerance, among others [99, 215]. This explains why researchers and engineers consistently develop new CC techniques tailored for RTC, rather than directly adapting TCP-based schemes. Therefore, it is both unreasonable and infeasible to (at least directly) apply the theories, adopt the assumptions, or leverage existing SOTA CC solutions (e.g., Orca [216], MOCC [217] and Sage [58]) from TCP.

Specifically for RTP, CC is realized via BWE-governed rate adaptation at the application layer, coupled with a feedback loop mechanism, as portrayed in Figure 6.1. During an RTC session, the receiver uses received traffic statistics to consistently estimate the bandwidth availability of the bottleneck link, which is then fed back to the sender through real-time control protocol (RTCP) messages [30]. Upon receiving the estimated bandwidth, the sender[1] adjusts the traffic sending rate accordingly, thereby preventing overwhelming network links and accommodating bandwidth variations.

Traditionally, RTC employs heuristic techniques such as NADA [218] and Google Congestion Control (GCC) [99], implementing delay- and loss-based controllers to address congestion, which, however, have been thoroughly proved to be deficient [111, 219]. As one of the most important optimization components, CC inexorably ascends as a focal point of advancement, with BWE experiencing a prevalent shift to learning-based approaches [220, 221], among which RL asserts dominance [222, 216] and DNNs commonly operate as bandwidth estimators to learn optimal strategies [223]. Capitalizing on its aptitude to exploit massive data-driven explorations and its conceptual alignment with CC in RTC, RL has achieved exceptional progress, exceeding conventional approaches in almost every way.

[1]The sender simply consumes the estimation information to guide the encoder to conform to the available bandwidth, i.e., the BWE and rate adjustment are normally technique-decoupled.

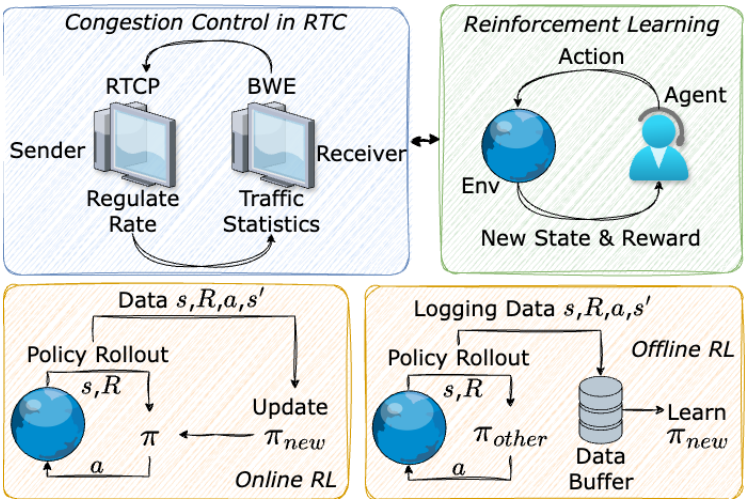


Fig. 6.1 RL & BWE in congestion control for RTC.

In general, RL follows a discrete-time stochastic control process, wherein an agent (a deep bandwidth estimator) interacts with the environment (an RTC session or traffic traces). At time t, the agent resides in a state S_t (received traffic statistics) and executes an action a_t (estimating the current bandwidth), which affects the environment (causing the sender to regulate the bitrate) to grant a reward R_t (reflecting network performance) and transition to a new state S_{t+1} (updated statistics of incoming traffic). RL aims to train the agent to learn an optimal policy π (i.e., how the agent predicts bandwidth based on varying traffic statistics) in order to maximize long-term rewards. Importantly, it does not nor is it designed to necessarily and explicitly pursue an accurate BWE. As depicted in Figure 6.1, applying RL to CC is somewhat intuitive because of the inherent alignment between CC's iterative decision-making loop and RL's sequential learning paradigm. Generally, there are two learning paradigms: 1) the online RL requires the agent to continuously interact with the environment through a trial-and-error process, upgrading its policy incrementally upon receiving new traffic statistics each time, and 2) the offline one, also referred to as data-driven RL, exploits pre-collected data generated by existing policies (e.g., heuristic BWE experts) and learns a refined policy that chases higher cumulative rewards [58].

The "hype" of RL-driven BWE in RTC dates back to the *ACM MMSys2021 Grand Challenge* [224], which provides *OpenNetLab* alongside its built-in *AlphaRTC* and *Gym* environments [223] — an open platform and testing ground that emulate real video-call sessions and enable/assess plug-in custom bandwidth estimators. As a

result, numerous RL-based solutions have emerged [225–229], which engage either a hybrid scheme that coordinates RL models and heuristic methods, or purely DL-based agents of various NN architectures, e.g., CNNs, trained via different RL paradigms, e.g., the Actor-Critic (AC) method [230] and Proximal Policy Optimization (PPO) algorithm [231]. These nascent approaches, despite their notable advantages over heuristics, are quickly surpassed by newly released RL solutions, with the trend further proliferating fueled by the 2^{nd} *Bandwidth Estimation Challenge* in *ACM MMSys2024* [232], which concentrated on offline RL models and open-sourced an ample corpus of traces. In response, the challenge winners [233, 88, 234] demonstrated promising performance, outperforming relatively earlier offline models like Merlin [235]. They employed different more advanced RL techniques, such as implicit Q-learning (IQL) [236] and Twin Delayed Deep Deterministic policy gradient algorithm with behavior clone (TD3+BC) [237].

Notwithstanding their remarkable success, the fundamental disadvantages of RL are intentionally or otherwise disregarded: inferior generalizability, reliance on imperfect expertise, and excessive training time [238, 239]. Meanwhile, many studies are unpersuasive, comparing only with simplistic baselines [78, 240–242]. Specifically:

1. Online RL models are highly environment-dependent, and thus their generalizability to new scenarios remains controversial.

2. Offline models hinge on (and are thus constrained by) the knowledge acquired from known experts (existing BWE policies), which tend to be inherently defective.

3. RL models typically yield overlong training time, diminishing the efficiency when retraining is frequently needed in today's evolving RTC landscape (also given the two above-mentioned drawbacks).

Indeed, certain works [243, 244] have partially identified (and attempted to tackle) such limitations, but they remain entrenched within the box of RL — the workarounds are anyhow confined without breakthrough in RL theory per se. The aspiration for an optimal BWE algorithm seems to be unduly tethered to DRL, a powerful yet opaque black box, while the elementary root-cause analysis is largely ignored, rendering the purportedly distinguished performance unreliable, which significantly discredits the

implementation feasibility of RL-based BWE in practice, and underpins why most of contemporary RTC applications are still heuristic-driven.

Theoretical to Practical

All three previous works have investigated the "theoretical" feasibility of predicting QoS to achieve specific objectives, which remain data-driven solutions that can potentially (but are not strictly justified to) enhance real-world performance. In this chapter, an underlying goal is to understand the extent to which ML can deliver "practical" performance gains. In other words, we implement ML technologies in an actual RTC system, evaluating the improvement on session quality (that is indirectly related to the output of ML models) instead of on the prediction (model output) itself.

6.2 Observation

6.2.1 Preliminary

The *Gym* framework

We base our work on *OpenNetLab Gym*, a plug-and-play tool that uses *ns-3* and WebRTC to establish one-way RTC. It provides an interface for implementing customized bandwidth estimators, and generate/transmit quasi-real traffic at a discrete pace in emulated RTC sessions (environments), which are initialized with predefined multimedia content (e.g., a sample video) and given network traces. Each trace comprises a time-series of bandwidth profile along with other (presumable) auxiliary information including round-trip-time and loss rate, which determines the available link capacity that governs the transmitted traffic. At each time step, the receiver estimates the bandwidth based on statistics of received traffic, and afterwards, the sender adjusts the bitrate based on the estimation, with the newly generated traffic subjected to the actual bandwidth availability, reflecting the current network conditions in the updated traffic statistics. At the end of a session, we can observe the logged data/metrics to evaluate the network performance of the employed bandwidth estimator.

QoE score

The performance is assessed using a metric promoted by the *Grand Challenge* [224] and adopted by most of existing works, encompassing three networking aspects: receiving rate, delay and packet loss. Specifically, the overall QoE score is calculated as an average:

$$\begin{aligned} QoE &= \frac{QoE_{rr} + QoE_{delay} + QoE_{loss}}{3} \\ \text{with } QoE_{rr} &= 100 \times U,\ QoE_{delay} = 100 \times \frac{d_{max} - d_{95th}}{d_{max} - d_{min}},\ QoE_{loss} = 100 \times (1 - L). \end{aligned} \tag{6.1}$$

QoE_{rr} is the QoE score for receiving rate and U is the median bandwidth utilization. QoE_{delay} represents the delay QoE score, where d_{max}, d_{min}, d_{95th} are specific values in the distribution of all delays observed during the session. QoE_{loss} denotes the loss QoE with L indicating the mean of all loss ratios.

6.2.2 BWE: the deterministic factor

The critical role of network bandwidth in RTC, particularly for RTP traffic that encapsulates multimedia content and normally exhibits high throughput, have been corroborated in substantial studies [215, 245, 30, 246]. To substantiate that BWE accuracy stands for the most decisive factor, we train as well as test an online RL model on five randomly selected traces with low, medium, or high bandwidth availability[2]. As a result, we deduce the upper performance limit of RL, as the model discovers the optimum strategy specifically in environments matching the test set. Meanwhile, we employ a "perfect" bandwidth estimator, i.e., Prophet, which knows the actual bandwidth a priori — it informs the bandwidth ground-truth as its estimation to the sender at each time step. We notice that an utterly-precise BWE could incur relatively aggressive traffic generation to flood the network when encountering bandwidth fluctuations. With this in mind, we intentionally introduce a

[2]Notes: 1) When it comes to environment-specific performance, online models naturally surpass offline ones, as they directly interact with test environments during training, so there is no need to implement or evaluate offline RL herein. 2) The 5 traces are presented for the sake of concise illustration, while the result remains valid for all traces.

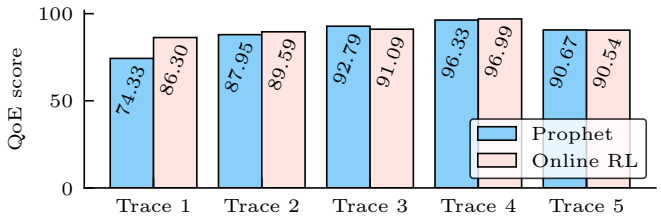


Fig. 6.2 QoE scores for Prophet and online RL model.

margin ratio to relax BWE instead of using the intact value (e.g., $BW_{actual} \times 0.9$)[3], thereby proffering a certain degree of tolerance towards bandwidth variations.

Figure 6.2 showcases the results in terms of QoE scores of both estimators on each trace. Except for trace 1, where the online RL model performs better, the score disparities are generally trivial for other traces. This suggests that identifying bandwidth availability alone is sufficient to achieve decent performance on par with RL models' ceiling, which is, however, a theoretical yet practically unattainable superiority, as estimators are invariably deployed in new scenarios distinct from their training environments. While a perfect estimator like Prophet does not exist, it implies that prioritizing precise bandwidth estimation can already deliver satisfactory outcomes. Intuitively, with accurate BWE, the sender can manage the bitrate to deliver an appropriate amount of traffic that respect the bottleneck capacity, ensuring neither underutilization nor overuse of the link, and resulting in improved network/application performance.

6.3 Methodology

As previously illustrated, CC in RTC can be optimized solely through accurate BWE without the need of RL, re-framing the problem as a supervised learning task (regression) — predicting the current available bandwidth based on traffic statistics. Nonetheless, the standard supervised learning paradigm, wherein predicted targets and features form fixed pairs, does not align with the dynamic nature of CC. At a step in an RTC session, the current BWE is based on the most-recent traffic statistics, which are subjected to the previous BWE and meanwhile influences future traffic and BWE. Given that BWE is never perfect, this recursive dependency (a propagation

[3]We observe that a ratio between 0.90 and 0.92 can yield equivalent and optimal performance, i.e., any values within this range only lead to negligible differences.

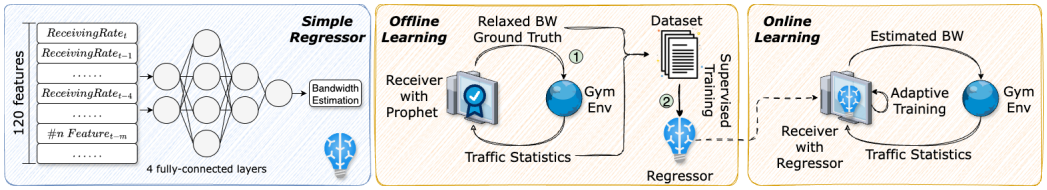


Fig. 6.3 The simple regressor with dual training stages.

Table 6.1 Regressor details

Parameter	**Value**
Initial learning rate	1×10^{-3}
Number of neurons per layer	120, 240, 120, 1
Activation function	*Leaky ReLU* [203]
Training optimizer	*Adam* [171]
Loss function	Mean Absolute Error (MAE)
Batch size (offline)	32
Batch size (online)	8

loop) results in varying traffic statistics, but the ground truth bandwidth is always fixed. In short, the supervised-learning assumption of a bijective feature (traffic statistics)-target (bandwidth) mapping in a constant data pool does not hold. To overcome this conflict, we propose a dual-stage training process, as delineated in Figure 6.3.

Simple regressor

We adopt a simple pure MLP-based NN with four fully connected layers as the base model. The output is basically the estimated bandwidth, and the input consists of 120 values (a feature vector), including 12 types of traffic statistics, such as receiving rate and queuing delay, in the five most-recent short (60 ms) and long (600 ms) monitoring intervals (10 in total)[4]. During training, we intentionally refrain from implementing any sophisticated tricks (coherent to "a simple regressor") except for a cosine annealing schedule [247] for the learning rate. Model training/tuning follow typical and standard DL paradigms (the model itself is a simple MLP after all), and the hyperparameters are enumerated in Table 6.1.

[4]We adopt the features proposed by the *MMSys2024 Challenge* [232] (https://www.microsoft.com/en-us/research/academic-program/bandwidth-estimation-challenge/data/). However, we opt to exclude certain features (e.g., packet probability metrics) that provide negligible utility for our regressor. This aligns with the challenge guidelines and some participants also omitted features for their own interests.

Offline learning

The first training stage is analogous to imitation learning or behavior cloning. The initial step involves a data collection campaign, where for each training trace, we interact with the *Gym* environment while assuming perfect knowledge of the bandwidth (i.e., employing Prophet). At each step, the receiver transmits a flawless yet relaxed "estimation" (i.e., the predicted target of bandwidth ground truth multiplied by the margin ratio) to the sender and record the corresponding spawned traffic statistics (features), forming a sample in the dataset. In consequence, we reproduce the behavior of the ideal bandwidth estimator, and the fixed BWE and its resulting statistics consistently match in the determinate environment, enabling the regressor to be trained following the conventional supervised learning procedure. In a word, we construct a high-quality dataset, aiming for the regressor to learn a mapping between the trajectory that Prophet follows and the theoretically optimal BWE.

Online learning

During the second training stage, we deploy the pretrained regressor in an online fashion, actively and recurrently interacting with the environment. Upon receiving statistics at each time step, the regressor produces an estimation and compare with the ground truth to calculate the regression loss, akin to standard supervised learning. The difference lies in the fact that the training step coincides with the environment emulation step, where each BWE is conditioned based on the dynamically varying statistics that, in turn, depend on previous estimations. Because of the imperfect nature of estimations that propagate and compound errors during inference, the offline pretrained regressor is insufficient on its own. Thus, we aim to adapt the regressor on the fly, mastering the nuanced relationship between optimal BWE and traffic dynamics. Additionally, to avoid overly relying on one specific training trace, we establish and communicate with eight different environments (i.e., a batch size of 8) simultaneously — juxtaposing various traffic traces and producing a BWE for each to foster a generalized solution (the loss is calculated collectively for all environments). Each environment is initiated with a random trace drawn from the training trace pool, and upon consuming a trace, a new trace is randomly selected to initialize the next environment, with a higher probability assigned to those not previously chosen.

It is imperative to reiterate that our solution is solely dedicated to precise BWE, bypassing the complexity and constraints associated with RL. On the other hand, RL algorithms are diametrically opposed to regression, and do not prioritize accurate BWE but instead strive to learn an optimal policy based on the multidimensional reward, which may be completely unrelated to bandwidth. For instance, the offline RL models in [232] are specifically designed to optimize audio and video quality.

6.4 Experiment

6.4.1 Experiment setup

Traffic & Dataset

The employed traces in our work are from multiple sources[5] and meticulously segmented into training and test sets to circumvent potential commonalities of environments between datasets, as a model is always trained on historical data but deployed in novel scenarios. For the training set, we consider the extensive data (a total of 28,264 traces) provided by the *MMSys2024 Challenge* [232], collected from massive video-calls using *Microsoft Teams* between testbed nodes or during emulated sessions. This dataset is selected for training because: 1) it is the only one that offers various BWE policies alongside audio/video qualities as the required rewards, i.e., experts from which offline RL models can learn, and 2) the corresponding test traces referenced in the Challenge were not publicly released (so it cannot be used as the test set). Importantly, the training set is designed to cover a diverse network conditions with a wide range of bandwidth characteristics ranging from low/moderate/high availability to stable/periodic/volatile variations, thereby enabling the model trained on it to exhibit generalizability. Regarding test set, we consider a diverse collection of traces [224, 248–250] (163 in total) recorded from real-world scenarios across different network connections (wireless/wired), times, and locations. As such, the test set utterly distinguishes from the training set, aiming to evaluate the model transferability/generalizability over diverse real-world networks.

[5]We employ public datasets (traces). References are available at https://www.microsoft.com/en-us/research/academic-program/bandwidth-estimation-challenge/links/ (training set) and https://github.com/denama/ReCoCo/tree/master/traces, https://github.com/denama/ReCoCo/tree/master/new_data (test set).

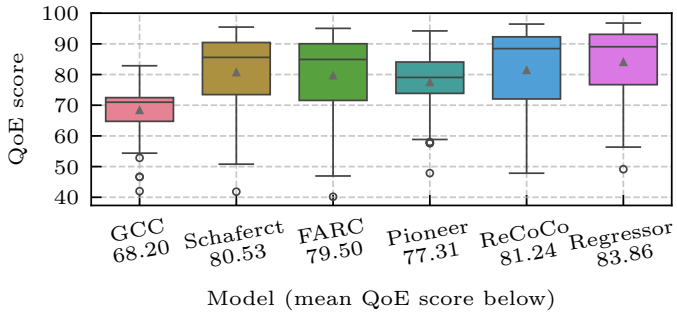


Fig. 6.4 Performance evaluation: QoE scores of all models.

Comparative method

In order to comprehensively validate the effectiveness of our regressor, we compare with multiple SOTA RL-based bandwidth estimators[6], that outshine the majority of existing works — one online RL model that is tailored for CC and demonstrates heightened performance: ReCoCo [86], and three offline RL models that win the 2^{nd} *ACM MMSys2024 BWE Challenge*: Schaferct [254], FARC [88], Pioneer [234]. We also consider GCC, the most well-known heuristic approach widely adopted by modern applications (e.g., Google Meet), as a simple baseline. Notably, it is unneeded to consider other regressors, as the MLP already represents the simplest one capable of online learning. During the training phase, RL models have access to the full set of training traces, while our regressor only entails a randomly-selected trace subset. During the test phase, each trained model functions as the bandwidth estimator, interacting with the *Gym* environment instantiated with each test trace, and afterwards, we compute the QoE score to evaluate the performance.

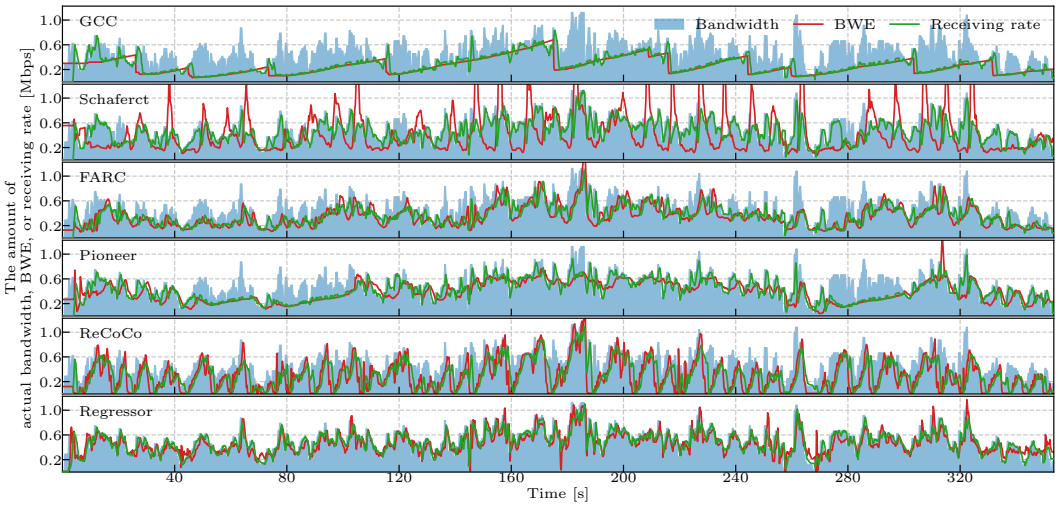


Fig. 6.5 Behaviors of different bandwidth estimators for an example traffic trace.

6.4.2 Experimental result

Figure 6.4 outlines the performance, with the traditional heuristic GCC approach, as anticipated, struggling to compete against ML-based models and delivering an average QoE score of mere 68.2. As for the online RL model (ReCoCo), we observe a slightly enhanced result compared with offline ones, potentially benefiting from its ability to interact dynamically with the *Gym* platform. Among offline RL models, Pioneer, despite producing the lowest QoE score, demonstrates relatively steady behavior, signified by its narrower distribution with only few instances scoring below 60, while Schaferct and FARC manifest comparable performance, both yielding mean scores around 80. Remarkably, our regressor achieves the best performance with the highest mean QoE score of nearly 84 and without sacrificing certain traces (one few scores lower than 60). Beyond the fact that the performance disparity can be validated by statistical tests, we still advocate for our regressor owing to its unique advantages, such as temporal efficiency, even if it does not yield superior performance but merely remains on par with RL-based approaches.

[6]Notes: 1) Certain existing models like BoB [227], although commonly used for comparison in many prior works, are relatively outdated and thus omitted hereafter. 2) Some RL-based frameworks, such as Loki [87] and OnRL [251], are ensemble solutions (not just for BWE) and target different objectives, e.g., wireless scenarios. 3) Few seemingly competitive BWE solutions [252, 253] exist but are close-sourced and thereby non-reproducible. 4) Other ML/DL models that are designed for bandwidth estimation [154, 156] formulate a fundamentally different problem with pure supervised learning, where predictions generally do not affect the traffic sending rate, and therefore, they are not involved in the feedback loop discussed earlier.

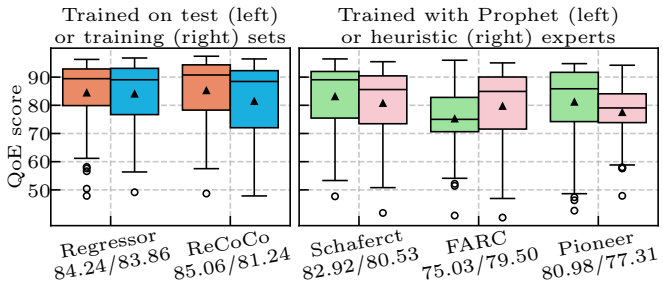


Fig. 6.6 Performance comparisons in different scenarios (the right one in each box plot duo is the original performance indicated in Figure 6.4).

To be intuitive, we showcase the traffic evolution, i.e., the time-series of actual bandwidth, BWE, and receiving rate, of each method for an example trace in Figure 6.5. Typically, GCC exhibits a sawtooth pattern, gradually increasing its estimation until congestion occurs. RL-based solutions manifest either aggressive, conservative, or volatile behaviors, which, despite tracking generally the network oscillation, remain evidently bandwidth-unbounded. In contrast, our regressor rigidly conforms to the bandwidth profile, striving to neither underutilize nor exceed the available link capacity.

Furthermore, we also examine the bandwidth overshooting ratio, defined as the percentage of samples with a BWE greater than the ground truth, which represents a critical factor for performance assessment, as excessive traffic induced by overestimation can directly trigger network congestion. Unsurprisingly, GCC obtains the top rank with a overshooting ratio at only 8.8%, attributed to its cautious strategy that avoids saturating the network bandwidth, but this restraint, in turn, results in mediocre QoE performance. As for all RL models, the resulting ratios are generally above 20%. Remarkably, our regressor, as the runner-up, yields a ratio of 13.8% thanks to its deliberate compliance with the constraint-loosened bandwidth targets.

6.4.3 Drawback of RL

Herein, we elaborate on the three aforementioned flaws of RL-based solutions.

Model generalizability

Inherently, Online RL models are environmentally dependent, as they are trained to earn high rewards in the specific environments (emulation spaces and training traces), which invariably differ from the deployment scenarios (diverse real-world services and network conditions). This discrepancy is particularly pronounced in today's RTC landscape, wherein emerging applications and evolving network infrastructures with frequent reconfigurations introduce continuous variability. To compare the generalizability of different models to novelties, we retrain both our regressor and ReCoCo particularly on the test set to evaluate the performance gap between identical and differing training/testing environments. As suggested by the results in Figure 6.6 (left), ReCoCo undergoes an ascent with a performance gain of roughly 4 in the average QoE score when able to access the testing environments. On the contrary, our regressor delivers nearly equivalent outcomes, with only a slight improvement compared to its original performance, eliminating overfitting and signifying robust behavior irrespective of environments. These arise from the fact that online RL models seek optimum strategies most suitable to the training set that may not translate effectively to unseen scenarios, while the regressor only employs a single, environment-agnostic strategy of accurately estimating bandwidth, ensuring unbiased treatment of varying network conditions to augment generalizability.

Limitation of experts

Offline RL approaches rely on existing policies, aiming to mimic and refine expert behaviors. They are devised to traverse a variety of state-action pairs and prioritize long-term cumulative returns, thereby exploiting high-reward trajectories and transcending the experts behind the scene. In this regard, the offline models and experts employed are strongly tied, and thus the potential of RL is determined by the quality of the expert. However, existing BWE policies in training traces, such as Kalman-filtering-based estimators, are inherently suboptimal (otherwise, the development of new approaches would be unnecessary), which fundamentally restricts the capacity of RL-based solutions from the outset. To verify this, we retrain[7] the three offline

[7]When generating the dataset for the regressor during offline learning, we need to compute a reward [86] for each sample based on the optimal BWE (ground truth) to derive the Prophet-based training traces, in which the experts' behaviors are the ideal estimations with their corresponding traffic statistics, and the rewards are network-related qualities (i.e., QoE scores).

models using an advanced expert, Prophet, i.e., the perfect bandwidth estimator, which, as illustrated previously, proves to be a promising candidate capable of approaching the theoretical performance cap of RTC CC. Figure 6.6 (right) shows the results in comparison to the models trained on original experts. Except for FARC, which experiences a performance decline, suggesting an incompatibility between the model and Prophet, likely stemming from FARC's architectural design tailored to the original dataset, both Schaferct and Pioneer achieve notable improvements, with an approximate increase of 3 in the mean QoE score. This outcome underscores the critical role of accurate BWE, as justified by the superior expertise of Prophet, and reinforces the premise that a more proficient expert enhances the capacity of RL models to acquire refined policies.

Time consumption

In contemporary RTC, continual model retraining for ML-based solutions to compensate for new data environments is indispensable to conquer the challenges posed by the ever-evolving networking landscape. Nonetheless, RL-based approaches suffer from prohibitively lengthy training times, which may span days in some cases. This is because they commonly necessitate extensive data to explore a satisfactory policy and continuous monitoring of highly fluctuating evaluation metrics to allocate a convergence point, which significantly impedes their practicality and scalability. In our case, we leverage a GPU of NVIDIA Tesla V100-16GB, but it still takes roughly 6 hours to train offline models and more than twice that duration for the online one. Our regressor, on the other hand, demonstrates an acceptable temporal overhead, with offline learning requiring a negligible amount (several minutes) and online learning spending around half an hour to converge, owing to the streamlined framework architecture of a simple NN. Additionally, the minimal complexity also facilitates the model inference speed with respect to other RL models characterized by sophisticated structures, supporting a more swift and timely CC, as BWE can be computed promptly.

6.4.4 Supplementary material

In the original paper, we conduct an ablation study to illustrate the necessity of offline learning. On the one hand, it is intuitive that online learning is necessary,

since the model is trained in a hard-coded way during offline learning. On the other hand, the result (the evolution of training losses) indicates that the offline stage helps expedite and stabilize the learning process. See our paper [12] for details. Moreover, our journal extension was not yet completed at the time of this thesis submission. Nevertheless, our preliminary results—including the comparison against another SOTA RL model from a different learning paradigm, two additional ablation tests, and the evaluation of video quality generated under real traffic—substantiate a consistently enhanced performance of our solution.

6.5 Takeaways

In this chapter, we rethink the role of RL for BWE in RTC CC, demonstrating that the network optimization problem can be framed as a regression task, with a simple regressor rivaling SOTA RL-based BWE solutions. Our findings can be summarized as follows:

- RL has gained substantial attentions in optimizing BWE for CC in RTC and demonstrated varying degrees of effectiveness. However, its primary disadvantages of inferior generalizability, reliance on defective experts, and long training time are somehow overlooked. Meanwhile, we posit that the reward with multifarious aspects (e.g., network metrics or multimedia content qualities) may introduce misleading noises. The reward typically integrates diverse and holistic components to avoid being biased toward specific directions. However, RL models are propelled to derive reward-wise optimal policies, yet the reciprocally constraining factors could lead to suboptimal policies that fail to align with the intended objective.
- We observe that it is not necessary to optimize various aspects as RL is designed to do, but sufficient to simply accurately estimate bandwidth availability, reducing a complex problem to a regression task, which is naturally more tractable than RL-based optimization tasks. Simply put, a regressor is easier to perform accurate BWE than an RL model to discover the optimal policy. With the single primary objective of BWE in mind, the regressor leverages extensive data to capture the intricate relationships between bandwidth and

traffic statistics across diverse network conditions, thereby ensuring reliable performance and generalizing over unseen environments.

- One might argue that if BWE were the sole factor in the reward, RL could still be leveraged to prioritize bandwidth. However, while plausible, this approach essentially boils down to supervised learning disguised as RL — employing RL-based algorithms to perform regression, which merely replicates what we are already doing. In this case, the RL model is trained to approximate a direct mapping from network statistics to BWE, duplicating the functionality of a regressor but with unnecessary complexity, and this redundant formulation contradicts the intended purpose of RL.
- For future work, we identify three directions: improving performance with a more sophisticated regressor, extending the framework to other protocols, and incorporating root-cause analysis to leverage prior domain knowledge.

Returning to the research questions outlined in the background, RL is not the necessary path for BWE in RTC; even when employing RL approaches, incorporating domain knowledge is essential rather than relying solely on data-driven methods.

Chapter 7

Conclusion

The research presented in this thesis involves a series of systematic development of ML/DL-based solutions for QoS prediction and system management in RTC, with the ultimate objective of enhancing QoE and network/application performance. We focus on massive RTP traffic and network traces collected from various applications (e.g., VCAs) and vantage points (e.g., border routers), analyzing traffic patterns, identifying potential problems, and uncovering unique phenomena. In this final chapter, we conclude the thesis by summarizing our work and discussing future directions.

7.1 On ML applied to RTC

We are witnessing an unprecedented and flourishing advancement of Artificial Intelligence, catalyzed by the rapid development of LLMs dating back to the emergence of ChatGPT[1], which revolutionized both academia and industry in early 2022. Today, with the advent and progress of more sophisticated and competent technologies such as generative and agentic AI, both individuals and businesses are confronting profound changes, with numerous nations/companies/organizations shifting their strategies to develop as well as exploit AI tools. At the core, ML/DL algorithms underpin the cornerstone of AI technologies, leveraging abundant data, discovering subtle patterns/correlations/rules, and facilitating decision-making. Before this, the application of ML to networking had already become feasible and increasingly

[1] https://chat.openai.com/

popular, exemplified by RTC propelled by the pandemic, where ML/DL approaches play important roles in traffic classification, performance monitoring, network management, and more. Amidst such atmospheres, we spontaneously wonder for our interests: what is the convergence between AI and RTC?

In this vein, we unfold our research of applying ML to RTC, gradually pinpointing challenges, identifying defects, and proposing solutions. We commence our work by adopting common feature engineering approaches to detect the onset of bursty losses, followed by an endeavor to estimate throughput extremes based on packet-level information. Both works lay the foundation for our subsequent attempt to unify QoS prediction in an efficient way. Finally, we turn our interests to BWE in CC, debunking the role of RL and demonstrating the sufficiency of a simple regressor. Throughout the works, not only do we substantiate the effectiveness of our introduced ML algorithms, we also deliberately enhance methodological aspects, transitioning from coarse to fine features that provide granular information and effortless extraction, from single to multiple objectives that are comprehensive and efficient, from theoretical check to practical implementations that explore real-world scenarios, and (when needed) from simple to advanced models that are more capable and effective.

Conclusively, we discover that: 1) Packet losses tend to occur in bursts, making the identification of their onset particularly valuable, though inherently challenging due to data imbalance and network dynamics. 2) Traffic extremes in network throughput are critically important and can be better estimated through dedicated model design, except in cases of drastic variations. 3) Multiple per-flow QoS metrics can be predicted accurately in an efficient manner, bypassing the complexity of existing solutions and providing thorough and fine-grained network observability. 4) RL may not be the destination of BWE for CC; more precisely, it should take into account additional domain-related factors instead of being solely data-driven.

7.2 Future direction

Our work marks not an endpoint but a starting point for the application of ML in RTC. Specifically, we identify three promising directions for future research: 1) The first three studies discussed in this thesis primarily provide theoretical validation of model feasibility and efficacy, which is of limited value unless implemented in practice.

Therefore, it is crucial to evaluate the real-world deployments and application cases of our proposed frameworks, understanding the gap between purely data-driven problems and actual RTC scenarios. 2) Although we always take efficiency into account when formulating problems and devising ML models, it remains insufficient as our experiments run in controlled/simulated/offline environments. Due to the real-time nature and the application constraints, it is important to further improve data retrieval, model efficiency, etc., a series of efforts that combine research and engineering difficulties. 3) While our focus is on RTC, by analyzing the similarities and differences with other protocols, we envision the prospect of extending/transferring our solutions to a broader field to facilitate the more robust and effective AI-powered networking.

References

[1] Antonio Nistico, Dena Markudova, Martino Trevisan, Michela Meo, and Giovanna Carofiglio. A comparative study of rtc applications. In *2020 IEEE International Symposium on Multimedia (ISM)*, pages 1–8. IEEE, 2020.

[2] Andrea Di Domenico, Gianluca Perna, Martino Trevisan, Luca Vassio, and Danilo Giordano. A network analysis on cloud gaming: Stadia, geforce now and psnow. *Network*, 1(3):247–260, 2021.

[3] Amritpal Kaur and Sarbjeet Singh. A survey of streaming protocols for video transmission. In *Proceedings of the International Conference on Data Science, Machine Learning and Artificial Intelligence*, pages 186–191, 2021.

[4] Mahbub Hassan, Alfandika Nayandoro, and Mohammed Atiquzzaman. Internet telephony: services, technical challenges, and products. *IEEE Communications magazine*, 38(4):96–103, 2000.

[5] Caroline Omoanatse Alenoghena, Henry Ohiani Ohize, Achonu Oluwole Adejo, Adeiza James Onumanyi, Emmanuel Esebanme Ohihoin, Aliyu Idris Balarabe, Supreme Ayewoh Okoh, Ezra Kolo, and Benjamin Alenoghena. Telemedicine: a survey of telecommunication technologies, developments, and challenges. *Journal of Sensor and Actuator Networks*, 12(2):20, 2023.

[6] Mowei Wang, Yong Cui, Xin Wang, Shihan Xiao, and Junchen Jiang. Machine learning for networking: Workflow, advances and opportunities. *Ieee Network*, 32(2):92–99, 2017.

[7] Mohammad Azmi Ridwan, Nurul Asyikin Mohamed Radzi, Fairuz Abdullah, and YE Jalil. Applications of machine learning in networking: A survey of current issues and future challenges. *IEEE access*, 9:52523–52556, 2021.

[8] Dena Markudova. Machine learning for quality of experience in real-time applications. 2023.

[9] Tailai Song, Gianluca Perna, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. Packet loss in real-time communications: Can ml tame its unpredictable nature? *IEEE Transactions on Network and Service Management*, 2024.

[10] Tailai Song, Paolo Garza, Michela Meo, and Maurizio M Munafò. Dex: Deep learning-based throughput prediction for real-time communications with emphasis on traffic extremes. *Computer Networks*, 249:110507, 2024.

[11] Tailai Song, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. One is enough: Efficient modeling of rtp traffic for qos predictions in real-time communications. *IEEE Transactions on Networking*, 34:2589–2604, 2025.

[12] Tailai Song and Michela Meo. Debunking reinforcement learning for bandwidth estimation in real-time communications: When a simple regressor suffices. In *2025 21st International Conference on Network and Service Management (CNSM)*, 2025.

[13] Tailai Song, Dena Markudova, Gianluca Perna, and Michela Meo. Where did my packet go? real-time prediction of losses in networks. In *ICC 2023-IEEE International Conference on Communications*, pages 3836–3841. IEEE, 2023.

[14] Tailai Song, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. Throughput prediction in real-time communications: Spotlight on traffic extremes. In *2024 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–7. IEEE, 2024.

[15] Tailai Song, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. Modelling concurrent rtp flows for end-to-end predictions of qos in real time communications. In *2024 International Symposium on Multimedia (ISM)*, pages 63–70, 2024.

[16] Tailai Song, David Lopez, Michela Meo, Nicola Piovesan, and Daniela Renga. High altitude platform stations: the new network energy efficiency enabler in the 6g era. In *2024 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2024.

[17] Tailai Song, Gianluca Perna, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. Bitformer: Transformer-based neural network for bitrate prediction in real-time communications. In *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, pages 65–70. IEEE, 2024.

[18] Tailai Song, Paolo Garza, Michela Meo, and Maurizio Matteo Munafò. Towards the detection of unobservable losses in real-time communications. In *2024 IEEE 30th International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pages 21–26. IEEE, 2024.

[19] Tailai Song, Mukharbek Organokov, Lennart Gulikers, Giulio Grassi, Giovanna Carofiglio, and Michela Meo. Advancing cloud-native cyber threat detection with graph-based feature engineering. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 4291–4297. IEEE Computer Society, 2025.

[20] Tailai Song, Pedro Casas, and Michela Meo. Do we really need reference-based phishing detection? unleashing the power of gnn. In *2025 9th Network Traffic Measurement and Analysis Conference (TMA)*, pages 1–4. IEEE, 2025.

[21] Anja Feldmann, Oliver Gasser, Franziska Lichtblau, Enric Pujol, Ingmar Poese, Christoph Dietzel, Daniel Wagner, Matthias Wichtlhuber, Juan Tapiador, Narseo Vallina-Rodriguez, Oliver Hohlfeld, and Georgios Smaragdakis. The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic. In *Proceedings of the ACM Internet Measurement Conference*, IMC '20, page 1–18, New York, NY, USA, 2020. Association for Computing Machinery.

[22] Thomas Favale, Francesca Soro, Martino Trevisan, Idilio Drago, and Marco Mellia. Campus traffic and e-learning during covid-19 pandemic. *Computer networks*, 176:107290, 2020.

[23] Nicholas Bloom, Ruobing Han, and James Liang. How hybrid working from home works out. Technical report, National Bureau of economic research, 2022.

[24] Andrew Fecheyr-Lippens. A review of http live streaming. *Internet Citation*, pages 1–37, 2010.

[25] Thomas Stockhammer. Dynamic adaptive streaming over http– standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144, 2011.

[26] Ron Frederick, Stephen L. Casner, Van Jacobson, and Henning Schulzrinne. RTP: A transport protocol for real-time applications. RFC 1889, January 1996.

[27] David P. Reed and Jon Postel. User datagram protocol. https://tools.ietf.org/html/rfc768. Accessed: 2023-11-16.

[28] J Uberti, C Jennings, and S Murillo. Rfc 9335: Completely encrypting rtp header extensions and contributing sources, 2023.

[29] Bill Marczak and John Scott-Railton. Move fast and roll your own crypto: A quick look at the confidentiality of zoom meetings. April 2020.

[30] Zaheduzzaman Sarker, Colin Perkins, Varun Singh, and M Ramalho. Rtp control protocol (rtcp) feedback for congestion control. *Internet RFC*, (8888), 2021.

[31] Tim Dierks and Christopher Allen. The tls protocol version 1.0. Technical report, 1999.

[32] Eric Rescorla and Nagendra Modadugu. Datagram transport layer security version 1.2. Technical report, 2012.

[33] Jonathan Rosenberg, Joel Weinberger, Christian Huitema, and Rohan Mahy. Stun-simple traversal of user datagram protocol (udp) through network address translators (nats). Technical report, 2003.

[34] Rohan Mahy, Philip Matthews, and Jonathan Rosenberg. Traversal using relays around nat (turn): Relay extensions to session traversal utilities for nat (stun). Technical report, 2010.

[35] Branislav Sredojev, Dragan Samardzija, and Dragan Posarac. Webrtc technology overview and signaling solution design and implementation. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 1006–1009. IEEE, 2015.

[36] Mark Baugher, David McGrew, Mats Naslund, Elisabetta Carrara, and Karl Norrman. The secure real-time transport protocol (srtp). Technical report, 2004.

[37] Gianluca Perna, Dena Markudova, Martino Trevisan, Paolo Garza, Michela Meo, and Maurizio M Munafò. Retina: An open-source tool for flexible analysis of rtc traffic. *Computer Networks*, 202:108637, 2022.

[38] Yevgeniy Dodis, Daniel Jost, Balachandar Kesavan, and Antonio Marcedone. End-to-end encrypted zoom meetings: Proving security and strengthening liveness. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 157–189. Springer, 2023.

[39] Henning Schulzrinne, Steven Casner, R Frederick, and Van Jacobson. Rfc3550: Rtp: A transport protocol for real-time applications, 2003.

[40] Giovanna Carofiglio, Giulio Grassi, Enrico Loparco, Luca Muscariello, Michele Papalini, and Jacques Samain. Characterizing the relationship between application qoe and network qos for real-time services. In *Proceedings of the ACM SIGCOMM 2021 workshop on network-application integration*, pages 20–25, 2021.

[41] Boni García, Micael Gallego, Francisco Gortázar, and Antonia Bertolino. Understanding and estimating quality of experience in webrtc applications. *Computing*, 101(11):1585–1607, 2019.

[42] ITU-T Recommendation G.107: The E-model, a Computational Model for Use in Transmission Planning, 2005. Accessed: 2025-12-28.

[43] Markus Fiedler, Tobias Hossfeld, and Phuoc Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *Ieee Network*, 24(2):36–41, 2010.

[44] Yevgeniya Sulema, Noam Amram, Oleksii Aleshchenko, and Olena Sivak. Quality of experience estimation for webrtc-based video streaming. In *European Wireless 2018; 24th European Wireless Conference*, pages 1–6. VDE, 2018.

[45] Tobias Hossfeld, Markus Fiedler, and Thomas Zinner. The qoe provisioning-delivery-hysteresis and its importance for service provisioning in the future internet. In *2011 7th EURO-NGI Conference on Next Generation Internet Networks*, pages 1–8. IEEE, 2011.

[46] Gianluca Perna, Dena Markudova, Martino Trevisan, Paolo Garza, Michela Meo, Maurizio Matteo Munafò, and Giovanna Carofiglio. Real-time classification of real-time communications. *IEEE Transactions on Network and Service Management*, 19(4):4676–4690, 2022.

[47] Marco Mellia, Andrea Carpani, and Renato Lo Cigno. Tstat: Tcp statistic and analysis tool. In *Quality of Service in Multiservice IP Networks: Second International Workshop, QoS-IP 2003 Milano, Italy, February 24–26, 2003 Proceedings*, pages 145–157. Springer, 2003.

[48] Jinliang Fan, J Xu, M Ammar, and S Moon. Cryptography-based prefix-preserving anonymization. *URl: http://www. cc. gatech. edu/computing/Telecomm/projects/cryptopan/(cit. on p. 58)*.

[49] Michael Maruschke, Oliver Jokisch, Martin Meszaros, and Viktor Iaroshenko. Review of the opus codec in a webrtc scenario for audio and speech communication. In *Speech and Computer: 17th International Conference, SPECOM 2015, Athens, Greece, September 20-24, 2015, Proceedings 17*, pages 348–355. Springer, 2015.

[50] Jean-Marc Valin, Koen Vos, and Timothy Terriberry. Definition of the opus audio codec. Technical report, 2012.

[51] Wikipedia contributors. Machine learning. https://en.wikipedia.org/wiki/Machine_learning#cite_note-Samuel-8, 2025. Accessed: 2025-07-14.

[52] Wikipedia contributors. Timeline of machine learning. https://en.wikipedia.org/wiki/Timeline_of_machine_learning, 2025. Accessed: 2025-07-14.

[53] Koosha Sharifani and Mahyar Amini. Machine learning and deep learning: A review of methods and applications. *World Information Technology and Engineering Journal*, 10(07):3897–3904, 2023.

[54] Mohammad Mustafa Taye. Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. *Computers*, 12(5):91, 2023.

[55] Yutaka Matsuo, Yann LeCun, Maneesh Sahani, Doina Precup, David Silver, Masashi Sugiyama, Eiji Uchibe, and Jun Morimoto. Deep learning, reinforcement learning, and world models. *Neural Networks*, 152:267–275, 2022.

[56] Raouf Boutaba, Mohammad A Salahuddin, Noura Limam, Sara Ayoubi, Nashid Shahriar, Felipe Estrada-Solano, and Oscar M Caicedo. A comprehensive survey on machine learning for networking: evolution, applications

and research opportunities. *Journal of Internet Services and Applications*, 9(1):1–99, 2018.

[57] Danish Rafique and Luis Velasco. Machine learning for network automation: overview, architecture, and applications [invited tutorial]. *Journal of optical communications and networking*, 10(10):D126–D143, 2018.

[58] Chen-Yu Yen, Soheil Abbasloo, and H Jonathan Chao. Computers can learn from the heuristic designs and master internet congestion control. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 255–274, 2023.

[59] Guillermo Bernárdez, José Suárez-Varela, Albert López, Bo Wu, Shihan Xiao, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Is machine learning ready for traffic engineering optimization? In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pages 1–11. IEEE, 2021.

[60] Yang Xiao, Jun Liu, Jiawei Wu, and Nirwan Ansari. Leveraging deep reinforcement learning for traffic engineering: A survey. *IEEE Communications Surveys & Tutorials*, 23(4):2064–2097, 2021.

[61] Amin Shahraki, Mahmoud Abbasi, Amir Taherkordi, and Anca Delia Jurcut. Active learning for network traffic classification: A technical study. *IEEE Transactions on Cognitive Communications and Networking*, 8(1):422–439, 2021.

[62] Ola Salman, Imad H Elhajj, Ayman Kayssi, and Ali Chehab. A review on machine learning–based approaches for internet traffic classification. *Annals of Telecommunications*, 75(11):673–710, 2020.

[63] Song Wang, Juan Fernando Balarezo, Sithamparanathan Kandeepan, Akram Al-Hourani, Karina Gomez Chavez, and Benjamin Rubinstein. Machine learning in network anomaly detection: A survey. *IEEe Access*, 9:152379–152396, 2021.

[64] Gilberto Fernandes Jr, Joel JPC Rodrigues, Luiz Fernando Carvalho, Jalal F Al-Muhtadi, and Mario Lemes Proença Jr. A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70(3):447–489, 2019.

[65] Fatima Hussain, Syed Ali Hassan, Rasheed Hussain, and Ekram Hossain. Machine learning for resource management in cellular and iot networks: Potentials, current solutions, and open challenges. *IEEE communications surveys & tutorials*, 22(2):1251–1275, 2020.

[66] Javad Hassannataj Joloudari, Roohallah Alizadehsani, Issa Nodehi, Sanaz Mojrian, Fatemeh Fazl, Sahar Khanjani Shirkharkolaie, HM Dipu Kabir, Ru-San Tan, and U Rajendra Acharya. Resource allocation optimization using artificial intelligence methods in various computing paradigms: A review. *arXiv preprint arXiv:2203.12315*, 2022.

[67] Gianluca Perna, Dena Markudova, Martino Trevisan, Paolo Garza, Michela Meo, Maurizio M Munafò, and Giovanna Carofiglio. Online classification of rtc traffic. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2021.

[68] Alexander Dietmüller, Siddhant Ray, Romain Jacob, and Laurent Vanbever. A new hope for network model generalization. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pages 152–159, 2022.

[69] Guangwu Hu, Xi Xiao, Meng Shen, Bin Zhang, Xia Yan, and Yunxia Liu. Tcgnn: Packet-grained network traffic classification via graph neural networks. *Engineering Applications of Artificial Intelligence*, 123:106531, 2023.

[70] Xi Jiang, Shinan Liu, Aaron Gember-Jacobson, Arjun Nitin Bhagoji, Paul Schmitt, Francesco Bronzino, and Nick Feamster. Netdiffusion: Network data augmentation through protocol-constrained traffic generation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 8(1):1–32, 2024.

[71] Radoslav Gatev. Observability: Logs, metrics, and traces. In *Introducing distributed application runtime (dapr) simplifying microservices applications development through proven and reusable patterns and practices*, pages 233–252. Springer, 2021.

[72] Suying Yan, Yuchun Guo, Yishuai Chen, Feng Xie, Chenguang Yu, and Yong Liu. Enabling qoe learning and prediction of webrtc video communication in wifi networks. In *Proceedings of the ICC*, volume 2017, 2017.

[73] Taveesh Sharma, Tarun Mangla, Arpit Gupta, Junchen Jiang, and Nick Feamster. Estimating webrtc video qoe metrics without using application headers. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, pages 485–500, 2023.

[74] Doreid Ammar, Katrien De Moor, Lea Skorin-Kapov, Markus Fiedler, and Poul E Heegaard. Exploring the usefulness of machine learning in the context of webrtc performance estimation. In *2019 IEEE 44th conference on local computer networks (LCN)*, pages 406–413. IEEE, 2019.

[75] Jing Zhao, Bin Li, Jiahao Li, Ruiqin Xiong, and Yan Lu. A universal optimization framework for learning-based image codec. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(1):1–19, 2023.

[76] Amal Punchihewa and Donald Bailey. A review of emerging video codecs: Challenges and opportunities. In *2020 35th International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6. IEEE, 2020.

[77] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. {GRACE}:{Loss-Resilient}{Real-Time} video through neural codecs. In

21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24), pages 509–531, 2024.

[78] Nikita Smirnov and Sven Tomforde. Real-time rate control of webrtc video streams in 5g networks: Improving quality of experience with deep reinforcement learning. *Journal of Systems Architecture*, 148:103066, 2024.

[79] Yucheng Jing and Sang-Chul Kim. Adapt compression rate and improve network packet utilization based on rtp video stream. pages 127–130, 2024.

[80] Petr Matousek, Ondrej Rysavy, and Martin Kmet. Fast rtp detection and codecs classification in internet traffic. *Journal of Digital Forensics, Security and Law*, 9(2):9, 2014.

[81] Dena Markudova, Martino Trevisan, Paolo Garza, Michela Meo, Maurizio M Munafo, and Giovanna Carofiglio. What's my app? ml-based classification of rtc applications. *ACM SIGMETRICS Performance Evaluation Review*, 48(4):41–44, 2021.

[82] Rashid Amin, Elisa Rojas, Aqsa Aqdus, Sadia Ramzan, David Casillas-Perez, and Jose M Arco. A survey on machine learning techniques for routing optimization in sdn. *IEEE Access*, 9:104582–104611, 2021.

[83] Bin Dai, Yuanyuan Cao, Zhongli Wu, Zhewei Dai, Ruyi Yao, and Yang Xu. Routing optimization meets machine intelligence: A perspective for the future network. *Neurocomputing*, 459:44–58, 2021.

[84] Georgios Kougioumtzidis, Vladimir Poulkov, Zaharias D Zaharis, and Pavlos I Lazaridis. A survey on multimedia services qoe assessment and machine learning-based prediction. *Ieee Access*, 10:19507–19538, 2022.

[85] Ymer Gurra. *Estimating QoE from QoS in real-time traffic: a Machine learning approach*. PhD thesis, Politecnico di Torino, 2021.

[86] Dena Markudova and Michela Meo. Recoco: Reinforcement learning-based congestion control for real-time applications. In *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*, pages 68–74. IEEE, 2023.

[87] Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. Loki: improving long tail performance of learning-based real-time video adaptation by fusing rule-based models. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, pages 775–788, 2021.

[88] Ekrem Çetinkaya, Ahmet Pehlivanoglu, Ihsan U Ayten, Basar Yumakogullari, Mehmet E Ozgun, Yigit K Erinc, Enes Deniz, and Ali C Begen. Offline reinforcement learning for bandwidth estimation in rtc using a fast actor and not-so-furious critic. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 388–393, 2024.

[89] Jaiden Fairoze and Peitong Duan. Information leakage through packet lengths in rtc traffic. In *International Workshop on Security*, pages 277–296. Springer, 2024.

[90] Petra Gabriela, Doyoung Lee, Nguyen Van Tu, and James Won-Ki Hong. Machine learning-based auto-scaler for video conferencing systems. In *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, pages 142–150. IEEE, 2021.

[91] Jason Gerard, David C Bonilla, Abdelhak Bentaleb, and Sandra Céspedes. Optimizing quality and energy efficiency in webrtc with ml-powered adaptive fec. In *2024 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2024.

[92] Senem Tanberk, Volkan Dağlı, and Mustafa Kağan Gürkan. Deep learning for videoconferencing: A brief examination of speech to text and speech synthesis. In *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pages 506–511. IEEE, 2021.

[93] Lei Yan, Shiji Zhou, Wenxuan Zheng, and Jingyi Chen. Deep reinforcement learning-based resource adaptive scheduling for cloud video conferencing systems. 2024.

[94] Dunja Vucic and Lea Skorin-Kapov. The impact of packet loss and google congestion control on QoE for WebRTC-based mobile multiparty audiovisual telemeetings. In *International Conference on Multimedia Modeling*, pages 459–470. Springer, 2019.

[95] Wenyu Jiang and Henning Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proc. NOSSDAV*, pages 1–10, 2000.

[96] Pablo Perez, Jesus Macias, Jaime J Ruiz, and Narciso Garcia. Effect of packet loss in video quality of experience. *Bell Labs Technical Journal*, 16(1):91–104, 2011.

[97] Asif Ali Laghari, Rashid Ali Laghari, Asif Ali Wagan, and Aamir Iqbal Umrani. Effect of packet loss and reorder on quality of audio streaming. *EAI Endorsed Transactions on Scalable Information Systems*, 7(24), 2020.

[98] Qin Dai and Ralf Lehnert. Impact of packet loss on the perceived video quality. In *2010 2nd International Conference on Evolving Internet*, pages 206–209. IEEE, 2010.

[99] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–12, 2016.

[100] Michael Luby, Lorenzo Vicisano, Jim Gemmell, Luigi Rizzo, M Handley, and Jon Crowcroft. Forward error correction (FEC) building block. Technical report, 2002.

[101] Matteo Sacchetto, Yuen Huang, Andrea Bianco, and Cristina Rottondi. Using autoregressive models for real-time packet loss concealment in networked music performance applications. In *Proceedings of the 17th International Audio Mostly Conference*, pages 203–210, 2022.

[102] Atsushi Miyamoto, Kazuho Watanabe, and Kazushi Ikeda. Packet loss rate estimation with active and passive measurements. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–4. IEEE, 2012.

[103] Paul Barford and Joel Sommers. A comparison of probe-based and router-based methods for measuring packet loss. *submission,(see http://www. cs. wisc. edu/pb/publications. html)*, 2003.

[104] Joel Sommers, Paul Barford, Nick Duffield, and Amos Ron. A geometric approach to improving active packet loss measurement. *IEEE/ACM Transactions on Networking*, 16(2):307–320, 2008.

[105] Zhiguo Hu and Qiqiang Zhang. A new approach for packet loss measurement of video streaming and its application. *Multimedia Tools and Applications*, 77:11589–11608, 2018.

[106] Bart Jansen, Timothy Goodwin, Varun Gupta, Fernando Kuipers, and Gil Zussman. Performance evaluation of webrtc-based video conferencing. *ACM SIGMETRICS Performance Evaluation Review*, 45(3):56–68, 2018.

[107] Matteo Varvello, Hyunseok Chang, and Yasir Zaki. Performance characterization of videoconferencing in the wild. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 261–273, 2022.

[108] Ashkan Nikravesh, David Ke Hong, Qi Alfred Chen, Harsha V Madhyastha, and Z Morley Mao. Qoe inference without application control. In *Proceedings of the 2016 workshop on QoE-based Analysis and Management of Data Communication Networks*, pages 19–24, 2016.

[109] Albert Choi, Mehdi Karamollahi, Carey Williamson, and Martin Arlitt. Zoom session quality: A network-level view. In *International Conference on Passive and Active Network Measurement*, pages 555–572. Springer, 2022.

[110] Oliver Michel, Satadal Sengupta, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. Enabling passive measurement of zoom performance in production networks. In *Proceedings of the 22nd ACM Internet Measurement Conference*, pages 244–260, 2022.

[111] Kyle MacMillan, Tarun Mangla, James Saxon, and Nick Feamster. Measuring the performance and network utilization of popular video conferencing applications. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 229–244, 2021.

[112] Hyunseok Chang, Matteo Varvello, Fang Hao, and Sarit Mukherjee. Can you see me now? a measurement study of zoom, webex, and meet. In *Proceedings of the 21st ACM Internet Measurement Conference*, pages 216–228, 2021.

[113] João Victor Costa Carmona, Edemir Marcus Carvalho de Matos, Bruno Souza Lyra Castro, Fabrício José Brito Barros, Miércio Cardoso de Alcântara Neto, and Evaldo Gonçalves Pelaes. Video loss prediction model in wireless networks. *Plos one*, 14(3):e0212407, 2019.

[114] Lopamudra Roychoudhuri and Ehab S Al-Shaer. Real-time analysis of delay variation for packet loss prediction. In *IFIP/IEEE International Conference on Management of Multimedia Networks and Services*, pages 213–227. Springer, 2004.

[115] P Ganvir Manish and Dr SS Salankar. Time series forecasting of packet loss rate using artificial neural network based on particle swarm optimization. *International Journal of Engineering Research and General Science*, 3(2):466–472, 2015.

[116] Fernando Silveira Filho and E de Souza e Silva. A method for predicting packet losses with applications to continuous media streaming. In *Proceedings of the Brazilian Symposium on Computer Networks*, 2006.

[117] Anna Giannakou, Dipankar Dwivedi, and Sean Peisert. A machine learning approach for packet loss prediction in science flows. *Future Generation Computer Systems*, 102:190–197, 2020.

[118] Kalpana Saha Roy and Tune Ghosh. Study of packet loss prediction using machine learning. *Int. J. Mob. Commun. Netw*, 11:1–11, 2020.

[119] Mario Hock, Roland Bless, and Martina Zitterbart. Experimental evaluation of bbr congestion control. In *2017 IEEE 25th international conference on network protocols (ICNP)*, pages 1–10. IEEE, 2017.

[120] Yiran Zhang, Yifan Liu, Qingkai Meng, and Fengyuan Ren. Congestion detection in lossless networks. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, pages 370–383, 2021.

[121] Bong-Hwan Oh, Serdar Vural, Ning Wang, and Rahim Tafazolli. Priority-based flow control for dynamic and reliable flow management in sdn. *IEEE Transactions on Network and Service Management*, 15(4):1720–1732, 2018.

[122] Nwe Thazin, Khine Moe Nwe, and Yutaka Ishibashi. End-to-end dynamic bandwidth resource allocation based on qos demand in sdn. In *2019 25th Asia-Pacific Conference on Communications (APCC)*, pages 244–249. IEEE, 2019.

[123] Rana Fareed Ghani and Laith Al-Jobouri. Packet loss optimization in router forwarding tasks based on the particle swarm algorithm. *Electronics*, 12(2):462, 2023.

[124] Anusha Kannan, Sumathi Vijayan, Manikandan Narayanan, and Monesh Reddiar. Adaptive routing mechanism in sdn to limit congestion. In *Information Systems Design and Intelligent Applications: Proceedings of Fifth International Conference INDIA 2018 Volume 1*, pages 245–253. Springer, 2019.

[125] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[126] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys (CSUR)*, 52(4):1–36, 2019.

[127] Fan Li and Yiming Yang. Analysis of recursive feature elimination methods. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 633–634, 2005.

[128] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[129] Barry De Ville. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6):448–455, 2013.

[130] Gérard Biau and Erwan Scornet. A random forest guided tour. *Test*, 25(2):197–227, 2016.

[131] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[132] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[133] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[134] Pengzhi Li, Yan Pei, and Jianqiang Li. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, 138:110176, 2023.

[135] Chao Chen and Leo Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley*, 01 2004.

[136] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[137] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[138] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[139] Haakon Riiser, Tore Endestad, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Video streaming using a location-based bandwidth-lookup service for bitrate planning. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 8(3):1–19, 2012.

[140] James R Wilcox. *Videoconferencing: The whole picture*. Taylor & Francis, 2017.

[141] Chao Liang, Miao Zhao, and Yong Liu. Optimal bandwidth sharing in multiswarm multiparty p2p video-conferencing systems. *IEEE/ACM Transactions On Networking*, 19(6):1704–1716, 2011.

[142] Siqi Huang and Jiang Xie. Dave: Dynamic adaptive video encoding for real-time video streaming applications. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2021.

[143] Libin Liu, Jingzong Li, Hong Xu, Kaiwen Xue, and Jason Chun Xue. Efficient real-time video conferencing with adaptive frame delivery. *Computer Networks*, 234:109918, 2023.

[144] Yueheng Li, Zicheng Zhang, Hao Chen, and Zhan Ma. Mamba: Bringing multi-dimensional abr to webrtc. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9262–9270, 2023.

[145] Eliseu Torres, Rafael Reale, Leobino Sampaio, and Joberto Martins. A sdn/openflow framework for dynamic resource allocation based on bandwidth allocation model. *IEEE Latin America Transactions*, 18(05):853–860, 2020.

[146] Riza Arda Kirmizioglu and A Murat Tekalp. Multi-party webrtc services using delay and bandwidth aware sdn-assisted ip multicasting of scalable video over 5g networks. *IEEE Transactions on Multimedia*, 22(4):1005–1015, 2019.

[147] Netlimiter. https://www.netlimiter.com/. Last accessed on March 6, 2024.

[148] Netbalancer. https://netbalancer.com/. Last accessed on March 6, 2024.

[149] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 272–285, 2016.

[150] Darijo Raca, Ahmed H Zahran, Cormac J Sreenan, Rakesh K Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello. Empowering video players in cellular: Throughput prediction from radio network measurements. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 201–212, 2019.

[151] Darijo Raca, Ahmed H Zahran, Cormac J Sreenan, Rakesh K Sinha, Emir Halepovic, Rittwik Jana, and Vijay Gopalakrishnan. On leveraging machine and deep learning for throughput prediction in cellular networks: Design, performance, and challenges. *IEEE Communications Magazine*, 58(3):11–17, 2020.

[152] Marti A. Hearst, Susan T Dumais, Edgar Osuna, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[153] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. Real-time bandwidth prediction and rate adaptation for video calls over cellular networks. In *Proceedings of the 7th International Conference on Multimedia Systems*, pages 1–11, 2016.

[154] Chaoqun Yue, Ruofan Jin, Kyoungwon Suh, Yanyuan Qin, Bing Wang, and Wei Wei. Linkforecast: Cellular link bandwidth prediction in lte networks. *IEEE Transactions on Mobile Computing*, 17(7):1582–1594, 2017.

[155] Maxime Labonne, Jorge López, Claude Poletti, and Jean-Baptiste Munier. Short-term flow-based bandwidth forecasting using machine learning. *arXiv preprint arXiv:2011.14421*, 2020.

[156] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifan Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using lstm neural network and bayesian fusion. *Computer Networks*, 182:107515, 2020.

[157] Anirban Lekharu, K.Y. Moulii, Arijit Sur, and Arnab Sarkar. Deep learning based prediction model for adaptive video streaming. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 152–159. IEEE, 2020.

[158] Gerui Lv, Qinghua Wu, Weiran Wang, Zhenyu Li, and Gaogang Xie. Lumos: Towards better video streaming qoe through accurate throughput prediction. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 650–659. IEEE, 2022.

[159] Bo Wei, Hang Song, Shangguang Wang, Kenji Kanai, and Jiro Katto. Evaluation of throughput prediction for adaptive bitrate control using trace-based emulation. *IEEE Access*, 7:51346–51356, 2019.

[160] Jiaoyang Yin, Yiling Xu, Hao Chen, Yunfei Zhang, Steve Appleby, and Zhan Ma. Ant: Learning accurate network throughput for better adaptive video streaming. *arXiv preprint arXiv:2104.12507*, 2021.

[161] Ashkan Sobhani, Abdulsalam Yassine, and Shervin Shirmohammadi. A video bitrate adaptation and prediction mechanism for http adaptive streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 13(2):1–25, 2017.

[162] Alberto Dainotti, Antonio Pescapé, Pierluigi Salvo Rossi, Francesco Palmieri, and Giorgio Ventre. Internet traffic modeling by means of hidden markov models. *Computer Networks*, 52(14):2645–2662, 2008.

[163] Antonio Montieri, Giampaolo Bovenzi, Giuseppe Aceto, Domenico Ciuonzo, Valerio Persico, and Antonio Pescapè. Packet-level prediction of mobile-app traffic using multitask deep learning. *Computer Networks*, 200:108529, 2021.

[164] Rushi Babaria, Sharat Chandra Madanapalli, Himal Kumar, and Vijay Sivaraman. Flowformers: Transformer-based models for real-time network flow classification. In *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*, pages 231–238. IEEE, 2021.

[165] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[166] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[167] Andy T Liu, Shang-Wen Li, and Hung-yi Lee. Tera: Self-supervised learning of transformer encoder representation for speech. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2351–2366, 2021.

[168] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[169] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[170] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[171] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[172] Rob J Hyndman. Moving averages., 2011.

[173] Ali H Sayed. *Fundamentals of adaptive filtering*. John Wiley & Sons, 2003.

[174] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[175] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[176] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[177] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, 2019.

[178] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[179] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), dec 2022.

[180] Obinna Izima, Ruairí de Fréin, and Ali Malik. A survey of machine learning techniques for video quality prediction from quality of delivery metrics. *Electronics*, 10(22):2851, 2021.

[181] Maghsoud Morshedi and Josef Noll. A survey on prediction of pqos using machine learning on wi-fi networks. In *2020 International Conference on Advanced Technologies for Communications (ATC)*, pages 5–11. IEEE, 2020.

[182] Tarik Begluk, Jasmina Baraković Husić, and Sabina Baraković. Machine learning-based qoe prediction for video streaming over lte network. In *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–5. IEEE, 2018.

[183] Laeticia Nneka Onyejegbu, Ugochi Adaku Okengwu, Linda Uchenna Oghenekaro, Martha Ozohu Musa, and Augustine Obolor Ugbari. Ai-based qos/qoe framework for multimedia systems. In *Proceedings of the Future Technologies Conference*, pages 248–259. Springer, 2022.

[184] Lavesh Babooram and Tulsi Pawan Fowdur. Performance analysis of collaborative real-time video quality of service prediction with machine learning algorithms. *International Journal of Data Science and Analytics*, pages 1–33, 2024.

[185] Hossein Ebrahimi Dinaki, Shervin Shirmohammadi, Emil Janulewicz, and David Côté. Forecasting video qoe with deep learning from multivariate time-series. *IEEE Open Journal of Signal Processing*, 2:512–521, 2021.

[186] Sidath Handurukande, Szymon Fedor, Stefan Wallin, and Martin Zach. Magneto approach to qos monitoring. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 209–216. IEEE, 2011.

[187] Obinna Izima, Ruairí de Fréin, and Mark Davis. Video quality prediction under time-varying loads. In *2018 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 129–132. IEEE, 2018.

[188] Sarah Wassermann, Michael Seufert, Pedro Casas, Li Gang, and Kuang Li. Let me decrypt your beauty: Real-time prediction of video resolution and bitrate for encrypted video streaming. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 199–200. IEEE, 2019.

[189] Dhananjay Kumar, S Aishwarya, A Srinivasan, and L Arun Raj. Adaptive video streaming over http using stochastic bitrate prediction in 4g wireless networks. In *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, pages 1–8. IEEE, 2016.

[190] Zongshuai Zhang, Jiaying Huang, Lin Tian, Chunjing Yuan, and Yuanyuan Wang. Delay prediction for real-time video based on improved lstm. In *2022 IEEE 8th International Conference on Computer and Communications (ICCC)*, pages 2053–2057. IEEE, 2022.

[191] Inayat Ali, Sonia Sabir, Seungwoo Hong, and Taesik Cheung. Congestion or no congestion: Packet loss identification and prediction using machine learning. *arXiv preprint arXiv:2408.03007*, 2024.

[192] Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew Gormley. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36, 2024.

[193] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[194] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*, 2022.

[195] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International conference on big data (Big Data)*, pages 3285–3292. IEEE, 2019.

[196] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural networks*, 116:237–245, 2019.

[197] Ben Krause, Liang Lu, Iain Murray, and Steve Renals. Multiplicative lstm for sequence modelling. *arXiv preprint arXiv:1609.07959*, 2016.

[198] Daniel Soutner and Luděk Müller. Application of lstm neural networks in language modelling. In *Text, Speech, and Dialogue: 16th International Conference, TSD 2013, Pilsen, Czech Republic, September 1-5, 2013. Proceedings 16*, pages 105–112. Springer, 2013.

[199] Youru Li, Zhenfeng Zhu, Deqiang Kong, Hua Han, and Yao Zhao. Ea-lstm: Evolutionary attention-based lstm for time series prediction. *Knowledge-Based Systems*, 181:104785, 2019.

[200] Hossein Abbasimehr and Reza Paki. Improving time series forecasting using lstm and attention models. *Journal of Ambient Intelligence and Humanized Computing*, 13(1):673–691, 2022.

[201] Ali Hassani, Steven Walton, Jiachen Li, Shen Li, and Humphrey Shi. Neighborhood attention transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6185–6194, 2023.

[202] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.

[203] Jin Xu, Zishan Li, Bowen Du, Miaomiao Zhang, and Jing Liu. Reluplex made more practical: Leaky relu. In *2020 IEEE Symposium on Computers and communications (ISCC)*, pages 1–7. IEEE, 2020.

[204] Yan Li, Xinjiang Lu, Haoyi Xiong, Jian Tang, Jiantao Su, Bo Jin, and Dejing Dou. Towards long-term time-series forecasting: Feature, pattern, and distribution. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1611–1624. IEEE, 2023.

[205] Mingtian Tan, Mike Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. Are language models actually useful for time series forecasting? *Advances in Neural Information Processing Systems*, 37:60162–60191, 2024.

[206] Michel Fliess, Cédric Join, and Cyril Voyant. Prediction bands for solar energy: New short-term time series forecasting techniques. *Solar Energy*, 166:519–528, 2018.

[207] Xiaolei Liu, Zi Lin, and Ziming Feng. Short-term offshore wind speed forecast by seasonal arima-a comparison against gru and lstm. *Energy*, 227:120492, 2021.

[208] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[209] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430, 2021.

[210] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Yong Liu, Mingsheng Long, and Jianmin Wang. Deep time series models: A comprehensive survey and benchmark. *arXiv preprint arXiv:2407.13278*, 2024. Code available at https://github.com/thuml/Time-Series-Library/tree/main.

[211] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35:9881–9893, 2022.

[212] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[213] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*, 2022.

[214] Si-An Chen, Chun-Liang Li, Sercan O Arik, Nathanael Christian Yoder, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecast-ing. *Transactions on Machine Learning Research*, 2023.

[215] Songyang Zhang, Weimin Lei, Wei Zhang, and Yunchong Guan. Congestion control for rtp media: A comparison on simulated environment. In *International Conference on Simulation Tools and Techniques*, pages 43–52. Springer, 2019.

[216] Soheil Abbasloo, Chen-Yu Yen, and H Jonathan Chao. Classic meets modern: A pragmatic learning-based congestion control for the internet. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 632–647, 2020.

[217] Yiqing Ma, Han Tian, Xudong Liao, Junxue Zhang, Weiyan Wang, Kai Chen, and Xin Jin. Multi-objective congestion control. In *Proceedings of the Seventeenth European Conference on Computer Systems*, pages 218–235, 2022.

[218] Xiaoqing Zhu, Rong Pan, M Ramalho, and S Mena. Network-assisted dynamic adaptation (nada): a unified congestion control scheme for real-time media. *RFC 8698*, 2020.

[219] Luca De Cicco, Gaetano Carlucci, and Saverio Mascolo. Experimental investigation of the google congestion control for real-time flows. In *Proceedings of the 2013 ACM SIGCOMM workshop on Future human-centric multimedia networking*, pages 21–26, 2013.

[220] Ticao Zhang and Shiwen Mao. Machine learning for end-to-end congestion control. *IEEE Communications Magazine*, 58(6):52–57, 2020.

[221] Wenting Wei, Huaxi Gu, and Baochun Li. Congestion control: A renaissance with machine learning. *IEEE network*, 35(4):262–269, 2021.

[222] Huiling Jiang, Qing Li, Yong Jiang, GengBiao Shen, Richard Sinnott, Chen Tian, and Mingwei Xu. When machine learning meets congestion control: A survey and comparison. *Computer Networks*, 192:108033, 2021.

[223] Jeongyoon Eo, Zhixiong Niu, Wenxue Cheng, Francis Y Yan, Rui Gao, Jorina Kardhashi, Scott Inglis, Michael Revow, Byung-Gon Chun, Peng Cheng, and Yongqiang Xiong. Opennetlab: Open platform for rl-based congestion control for real-time communications. In *Proceedings of the 6th Asia-Pacific Workshop on Networking*, pages 70–75, 2022.

[224] ACM MMSys 2021 RTC Challenge. Grand challenge on bandwidth estimation for real-time communications, 2021. Accessed: Feb. 4, 2025.

[225] Y Tianrun, W Hongyu, H Runyu, Y Shushu, L Dingwei, and Z Jiaqi. Gemini: An ensemble framework for bandwidth estimation in web real-time communications. *ACM MMSys*, 2021.

[226] Bo Wang, Yuan Zhang, Size Qian, Zipeng Pan, and Yuhong Xie. A hybrid receiver-side congestion control scheme for web real-time communication. In *Proceedings of the 12th ACM Multimedia Systems Conference*, pages 332–338, 2021.

[227] Abdelhak Bentaleb, Mehmet N Akcay, May Lim, Ali C Begen, and Roger Zimmermann. Bob: Bandwidth prediction for real-time communications using heuristic and reinforcement learning. *IEEE Transactions on Multimedia*, 25:6930–6945, 2022.

[228] Haoyong Li, Bingcong Lu, Jun Xu, Li Song, Wenjun Zhang, Lin Li, and Yaoyao Yin. Reinforcement learning based cross-layer congestion control for real-time communication. In *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 01–06. IEEE, 2022.

[229] Feng Peng, Bingcong Lu, Li Song, Rong Xie, Yanmei Liu, and Ying Chen. Pacc: Perception aware congestion control for real-time communication. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 978–983. IEEE, 2023.

[230] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[231] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[232] Sami Khairy, Gabriel Mittag, Vishak Gopal, Francis Y Yan, Zhixiong Niu, Ezra Ameri, Scott Inglis, Mehrsa Golestaneh, and Ross Cutler. Acm mmsys 2024 bandwidth estimation in real time communications challenge. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 339–345, 2024.

[233] Qingyue Tan, Gerui Lv, Xing Fang, Jiaxing Zhang, Zejun Yang, Yuan Jiang, and Qinghua Wu. Accurate bandwidth prediction for real-time media streaming with offline reinforcement learning. In *Proceedings of the 15th ACM*

Multimedia Systems Conference, MMSys '24, page 381–387, New York, NY, USA, 2024. Association for Computing Machinery.

[234] Bingcong Lu, Keyu Wang, Jun Xu, Rong Xie, Li Song, and Wenjun Zhang. Pioneer: Offline reinforcement learning based bandwidth estimation for real-time communication. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 306–312, 2024.

[235] Aashish Gottipati, Sami Khairy, Gabriel Mittag, Vishak Gopal, and Ross Cutler. Real-time bandwidth estimation from offline expert demonstrations. *arXiv preprint arXiv:2309.13481*, 2023.

[236] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[237] Stephen Dankwa and Wenfeng Zheng. Twin-delayed ddpg: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In *Proceedings of the 3rd international conference on vision, image and signal processing*, pages 1–5, 2019.

[238] Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468, 2021.

[239] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[240] Wei Zhang, Xuefeng Tao, and Jianan Wang. Naorl: Network feature aware offline reinforcement learning for real time bandwidth estimation. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 326–331, 2024.

[241] Jingshun Du, Chaokun Zhang, Shen He, and Wenyu Qu. Learning-based congestion control assisted by recurrent neural networks for real-time communication. In *2023 IEEE Symposium on Computers and Communications (ISCC)*, pages 323–328. IEEE, 2023.

[242] Joyce Fang, Martin Ellis, Bin Li, Siyao Liu, Yasaman Hosseinkashi, Michael Revow, Albert Sadovnikov, Ziyuan Liu, Peng Cheng, Sachin Ashok, et al. Reinforcement learning for bandwidth estimation and congestion control in real-time communications. *arXiv preprint arXiv:1912.02222*, 2019.

[243] Aashish Gottipati, Sami Khairy, Yasaman Hosseinkashi, Gabriel Mittag, Vishak Gopal, Francis Y Yan, and Ross Cutler. Balancing generalization and specialization: Offline metalearning for bandwidth estimation. *arXiv preprint arXiv:2409.19867*, 2024.

[244] Aashish Gottipati, Sami Khairy, Gabriel Mittag, Vishak Gopal, and Ross Cutler. Offline to online learning for real-time bandwidth estimation, 2024.

[245] Sushi Anna George and Vinay Joseph. Optimizing bandwidth sharing for real-time traffic in wireless networks. In *ICC 2023-IEEE International Conference on Communications*, pages 3199–3204. IEEE, 2023.

[246] R Jesup. Rfc 8836: Congestion control requirements for interactive real-time media, 2021.

[247] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[248] Lifan Mei, Runchen Hu, Houwei Cao, Yong Liu, Zifa Han, Feng Li, and Jin Li. Realtime mobile bandwidth prediction using lstm neural network. In *Passive and Active Measurement: 20th International Conference, PAM 2019, Puerto Varas, Chile, March 27–29, 2019, Proceedings 20*, pages 34–47. Springer, 2019.

[249] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute path bandwidth traces from 3g networks: Analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 114–118, 2013.

[250] Jeroen Van Der Hooft, Stefano Petrangeli, Tim Wauters, Rafael Huysegems, Patrice Rondao Alface, Tom Bostoen, and Filip De Turck. Http/2-based adaptive streaming of hevc video over 4g/lte networks. *IEEE Communications Letters*, 20(11):2177–2180, 2016.

[251] Huanhuan Zhang, Anfu Zhou, Jiamin Lu, Ruoxuan Ma, Yuhan Hu, Cong Li, Xinyu Zhang, Huadong Ma, and Xiaojiang Chen. Onrl: Improving mobile video telephony via online reinforcement learning. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[252] Jingshun Du, Chaokun Zhang, Tao Tang, and Wenyu Qu. Learning-based transport control adapted to non-stationarity for real-time communication. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[253] Santhosh Sunderrajan and Liyan Liu. Optimizing rtc bandwidth estimation with machine learning, 2024. Accessed: 2025-02-11.

[254] Qingyue Tan, Gerui Lv, Xing Fang, Jiaxing Zhang, Zejun Yang, Yuan Jiang, and Qinghua Wu. Accurate bandwidth prediction for real-time media streaming with offline reinforcement learning. In *Proceedings of the 15th ACM Multimedia Systems Conference*, pages 381–387, 2024.