Debunking Reinforcement Learning for Bandwidth Estimation in Real-Time Communications: When a Simple Regressor Suffices

Debunking Reinforcement Learning for Bandwidth Estimation in Real-Time Communications: When a Simple Regressor Suffices

Tailai Song, Michela Meo
Department of Electronics and Telecommunications Engineering (DET)
Politecnico di Torino, Turin, Italy
first.last@polito.it

***Abstract*—Bandwidth estimation (BWE) serves as a pivotal mechanism in real-time communications (RTC), supporting congestion control (CC) by governing traffic sending rate and thereby optimizing network performance. In recent years, reinforcement learning (RL) has surged to prominence, transcending conventional approaches as a superior bandwidth estimator. However, as the hype continues to escalate, the relentless pursuit of increasingly sophisticated algorithms eclipses the investigation into the core principles of CC, while the inherent flaws of RL appear to be overlooked. In this work, we rethink the necessity of RL in RTC, and contend that BWE itself, rather than reward-driven learning, is the deterministic factor, thus rendering a regressor sufficient. We first underscore the paramount importance of BWE accuracy and propose a simple feedforward neural network-based regressor with dual training stages: an offline stage imitating the perfect estimator and an online stage accommodating traffic dynamics. Utilizing an advanced RTC platform, we benchmark our solution against multiple state-of-the-art RL-based BWE algorithms. Conclusively, our regressor achieves superior performance, potentially charting a new course for BWE and CC in RTC.**

***Index Terms*—Real-time communications, Real-time Transport Protocol, bandwidth estimation, congestion control, reinforcement learning, regression, Quality of Experience**

I. Introduction

Real-time communications (RTC) form the cornerstone of numerous indispensable Internet services, enabling multimedia content delivery, real-time interactions, and more across both professional and recreational domains. Most RTC applications, including video-teleconferencing, online gaming, and live streaming [1]–[3], where nearly instantaneous responsiveness and minimum latency are imperative, employ Real-time Transport Protocol (RTP) [4], and web browsers and mobile devices rely on the globally recognized standard, WebRTC [5], an open-source framework built atop RTP. In recent years, RTC has witnessed flourishing development and gained unprecedented popularity, ascribed to multitudinous lifestyles as well as the widespread adoption of remote working from consumers, and augmented bandwidth accessibility, alongside substantial growth in network infrastructures from service providers. Accompanying this progression, user expectations and preferences also diversify, calling for effective optimizations in RTC systems to furnish an exceptional, seamless overall communication experience that extends beyond basic audiovisual quality.

To this end, there is a burgeoning impetus to develop innovative and robust technologies aimed at improving network performance and Quality of Experience (QoE). As an essential optimization component, congestion control (CC) inexorably ascends as a focal point of advancement, with bandwidth estimation (BWE) experiencing a prevalent transition from heuristic approaches to learning-based ones [6], [7]. Within this trend, reinforcement learning (RL) asserts dominance [8], [9], particularly in the context of RTC, where deep neural networks (DNNs) commonly function as bandwidth estimators, learning optimal strategies to maximize performance rewards [10]. Capitalizing on its conceptual alignment with the underlying logic of CC in RTC and its ability to leverage massive data-driven explorations among different actions, RL has achieved unparalleled progress, surpassing traditional approaches in almost every way [11]. Amidst the enthusiasm surrounding the remarkable success and the persistent fixation on this expanding tendency, the intrinsic issues of RL have been, however, largely ignored: inferior generalizability, reliance on imperfect expertise, and excessive training time [12], [13], not to mention that many studies only compare with simplistic baselines rather than sophisticated models [14]–[17]. These render the purportedly distinguished performance unreliable and significantly discredit the feasibility of implementing RL-based BWE algorithms in practice.

In this work, we revisit the necessity of RL-based BWE for CC in RTC, arguing and demonstrating that approaches propelled by rewards accounting for various aspects are not optimal, while the BWE per se, on the other hand, embodies the predominant determinant. To this end, a regressor dedicated solely to refining BWE and enhancing regression precision rather than optimizing for multidimensional rewards (i.e., potential noises) is deemed enough. To tackle the incompatibility between supervised learning and CC mechanisms, we propose a two-stage training strategy, where the bandwidth is assumed to be known during the first stage, and the regressor is deployed online for fine-tuning in the second one. We leverage the widely adopted RTC framework, *OpenNetLab* [10], and incorporate massive traffic traces from both real-world and emulated environments. Moreover, we compare our regressor against multiple state-of-the-art (SOTA) solutions, including both online and offline RL methods. Consequently, our proposal yields enhanced performance in terms of QoE score, and we also elaborate on the defects of the RL-based algorithms.

The remainder of this paper is organized as follows: Section II provides a concise overview of BWE and RL for

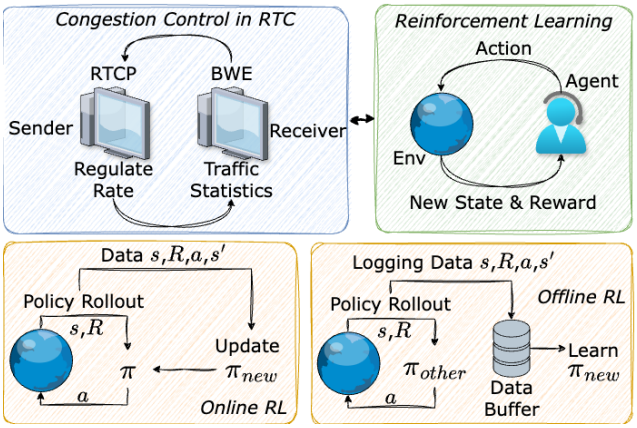

Fig. 1: RL & BWE in congestion control for RTC.

CC in RTC. Section III highlights the deterministic role of BWE, followed by Section IV, which delineates our proposed regressor. Section V describes the experiment details, and Section VI presents experimental results regarding model performance and the limitations of RL. Finally, Section VII offers an in-depth discussion, and Section VIII concludes the work.

II. Problem Context

In this section, we first briefly introduce the role of BWE and RL in CC for RTC, with related literature. Subsequently, we provide the necessary preliminary information.

A. Congestion control in RTC

For RTP-based services, CC is realized through BWE-regulated rate adaptation at the application layer, coupled with a feedback mechanism operating in a loop fashion, as portrayed in Figure 1. During an RTC session, the receiver recurrently estimates the available bandwidth of the bottleneck link based on statistics extracted from received traffic and feeds back this information to the sender via Real-time control protocol (RTCP) messages [18]. Upon receiving the estimated bandwidth, the sender adjusts the traffic sending rate accordingly, thereby adapting to bandwidth variations and mitigating network congestion. The BWE and rate adjustment are technique-decoupled, with the sender simply consuming the estimation to instruct the encoder to generate multimedia traffic that conforms to the available bandwidth. Conventionally, heuristic techniques such as NADA [19] and Google Congestion Control (GCC) [20] are broadly employed to optimize RTC, where delay- and loss-based controllers are implemented to address congestion. Nonetheless, these methods have been comprehensively proved to be deficient [21]–[23], therefore driving the advancement of RL-based methodologies.

Crucially, while other protocols, primarily TCP, share the overarching objective of regulating traffic rates to prevent overwhelming network links, the underlying mechanics behind their CC fundamentally differs from those of RTP, e.g., the operation vantage point/OSI layer, the required traffic statistics, the feedback granularity, the latency/loss tolerance, among others [20], [24], which underpins why researchers/engineers always develop new CC technologies tailored for RTC applications rather than simply adapting TCP-based CC schemes. Consequently, it is unreasonable/infeasible to (at least directly) adopt the assumptions, apply the theories, or leverage competent CC solutions (e.g., Sage [11] and Orca [9]) from TCP.

B. Reinforcement learning for BWE

Generally, RL follows a stochastic control process in a discrete-time way, e.g., Markov decision process (MDP), with an agent (a deep bandwidth estimator) interacting with the environment (an RTC session). At a given time instant t, the agent resides in a state S_t (received traffic statistics) and executes an action a_t (estimating the current bandwidth), which influences the environment (causing the sender to regulate the bitrate) to grant a reward R_t (reflecting network performance) and transition to the next 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, and it does not (nor is it designed to) necessarily as well as explicitly achieve a precise estimation of the available bandwidth. It is intuitive to apply RL to CC owing to the inherent alignment between CC's iterative decision-making loop and RL's sequential learning paradigm, as depicted in Figure 1. In particular, 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, while 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 [11].

The surge of RL-driven BWE in RTC can be traced back to the *ACM MMSys2021 Grand Challenge* [25] with the provision of *OpenNetLab* alongside its built-in *AlphaRTC* and *Gym* environments [10], which provide an open platform and testing ground to emulate real video-call sessions and enable as well as assess plug-in customized bandwidth estimators. In response, plenty of pertinent RL-based BWE solutions have emerged [26]–[30]. They engage either a hybrid scheme with the heuristic method intervening when RL models underperform, or purely deep learning (DL)-based agents trained using various RL paradigms (e.g., the actor-critic (AC) method) or neural network (NN) architectures (e.g., convolutional neural networks (CNNs)). Despite their notable advantages over heuristic methods, they are swiftly exceeded by other more advanced and newly-released RL approaches. The trend further proliferates fueled by the 2^{nd} *Bandwidth Estimation Challenge* in *ACM MMSys2024* [31], which released an abundant corpus of traces and impelled the development of offline RL models. As a result, the challenge winners [32]–[34] that employ different RL techniques (e.g., implicit Q-learning (IOL) and Twin Delayed Deep Deterministic policy gradient algorithm with behavior clone (TD3+BC)) and outperform relatively earlier offline models like Merlin [35], have demonstrated promising performance.

Notwithstanding their advancement, the fundamental disadvantages of RL are intentionally or otherwise disregarded: *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 other experts (existing BWE policies), which may be inherently defective, and *3)* the training time of RL models is overly prolonged, diminishing the efficiency when retraining is frequently needed, given the two above-mentioned drawbacks. Although certain works [36], [37] have attempted to partially address these limitations, they remain entrenched within the box of RL. The aspiration for an optimal BWE algorithm is unduly tethered to deep reinforcement learning (DRL), i.e., a powerful yet opaque black box, while the elementary root-cause analysis is largely neglected, which essentially restricts RL-based BWE and underpins why most of contemporary RTC applications continue to be heuristic-driven.

C. Preliminary

The *Gym* framework Throughout our work, we leverage *OpenNetLab Gym*, a plug-and-play tool that uses *ns-3* and WebRTC to emulate one-way RTC. It provides an interface for implementing different bandwidth estimators and conducts RTC sessions to transmit traffic at a discrete pace. The environment initializes with a sample video (multimedia content to be sent) and a predefined network trace (bandwidth profile). At each time step, we (receiver) estimate the bandwidth based on the received traffic statistics, and afterwards, the sender varies 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 access the logged data and evaluate the network performance of the employed bandwidth estimator.

QoE score The performance is assessed using a metric adopted by the *Grand Challenge* [25] (and many other works), taking into account three networking aspects: receiving rate, delay and packet loss. Specifically, the QoE score for receiving rate is computed by: $QoE_{rr} = 100 \times U$, where U represents the median bandwidth utilization, the delay QoE score is given by: $QoE_{delay} = 100 \times \frac{d_{max} - d_{95th}}{d_{max} - d_{min}}$, in which d_{max}, d_{min}, d_{95th} are specific values in the distribution of all delays observed during the session, and the loss QoE is defined as: $QoE_{loss} = 100 \times (1 - L)$, where L denotes the mean of all loss ratios. Finally, the overall QoE score is the average, i.e., $QoE = (QoE_{rr} + QoE_{delay} + QoE_{loss})/3$.

III. BWE: The Deterministic Factor

Numerous studies underscore the critical role of network bandwidth in RTC [18], [24], [38], [39], especially for RTP traffic, which encapsulates multimedia content and normally exhibits high throughput [40], [41]. We assert that accurate BWE stands for the most decisive factor in facilitating effective CC in RTC. To substantiate this claim, we first train an online RL model with five randomly selected traces of different bandwidth characteristics (low, medium, and high) and then assess its QoE score on these traces[1]. In this way, we deduce the upper performance limit of RL, as the model discovers the optimum strategy specifically for environments matching the test set. In parallel, we employ a "perfect" bandwidth estimator named Prophet, which possesses the actual bandwidth a priori — it informs the ground-truth of the available bandwidth as its estimation to the sender at each time step. Moreover, we envision that an ideal BWE may lead to relatively aggressive traffic generation, probably overwhelming the network due to inherent bandwidth fluctuations. Hence, we introduce a margin ratio to send a relaxed BWE instead of the intact value (e.g., $BW_{actual} \times 0.9$)[2], thereby proffering a certain degree of tolerance when confronting bandwidth variations.

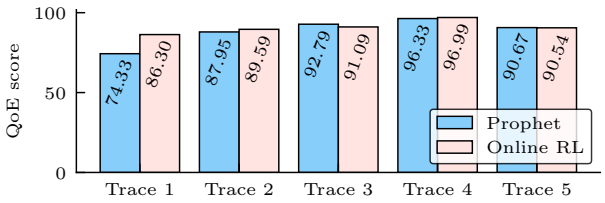

Fig. 2: QoE scores for Prophet and online RL model.

The bar plots in Figure 2 showcase the resultant QoE scores for both estimators on each trace. Except for trace 1, where the online RL model yields enhanced performance[3], the score differences are trivial for other traces. This finding suggests that comprehending bandwidth availability alone is enough to achieve remarkable performance comparable to RL models' ceiling, which is, however, a theoretical superiority yet unattainable in reality, as estimators are invariably deployed in new scenarios distinct from their training environments. Indeed, a perfect estimator like Prophet does not exist, but it implies that prioritizing precise bandwidth estimation is able to obtain decent outcomes, given the aforementioned observations. Intuitively, if the bottleneck bandwidth is accurately respected, the sender can manage the bitrate to deliver an appropriate amount of traffic, ensuring neither underutilization nor overuse of the link, and thus elongated delays as well as potential packet losses are prevented, resulting in improved network/application performance.

IV. Our Proposal

As previously illustrated, CC in RTC can be optimized solely through accurate BWE without the need of considering multifaceted rewards, re-framing the problem as a supervised learning task (regression) — predicting the current available bandwidth based on traffic statistics. However, the standard supervised learning paradigm, wherein predicted targets and features form fixed pairs, does not align with the dynamic nature of CC. At any given step in an RTC session, the current BWE is derived from the most-recent traffic statistics, which are subjected to the previous BWE, and meanwhile

[1]Notes: 1) When it comes to environment-specific performance, online models naturally outstrip 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.

[2]A ratio between 0.90 and 0.92 is found to yield optimal performance in our experiments, as it helps prevent network capacity saturation, particularly during bandwidth drops. Due to the page limit, we omit a detailed discussion and experiments that would rationalize this margin ratio.

[3]This arises from an uncommon low and stable bandwidth, where even minor bursts at constant sending rate with only 10% slack create persistent queues, causing high latency despite good throughput and minimal loss.

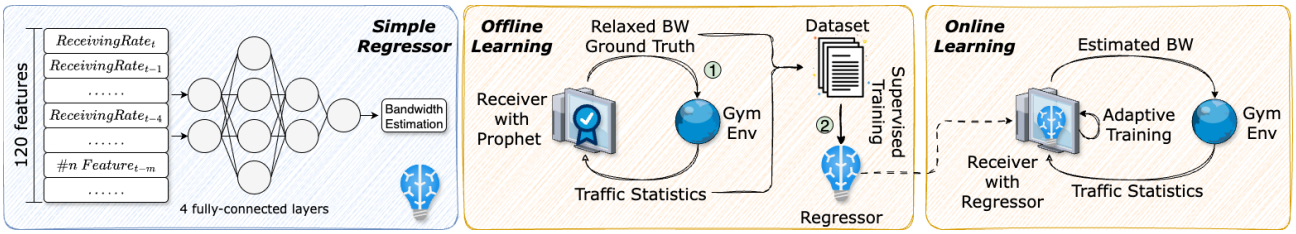

Fig. 3: The simple regressor with dual training stages.

TABLE I: Regressor details

Parameter	Value
Initial learning rate	1×10^{-3}
Number of neurons per layer	120, 240, 120, 1
Activation function	*Leaky ReLU* [43]
Training optimizer	*Adam* [44]
Loss function	Mean Absolute Error (MAE)
Batch size (offline)	32
Batch size (online)	8

influences subsequent traffic and future BWE. Since BWE is never perfect, this recursive dependency (a propagation loop) results in varying traffic statistics, but the ground truth bandwidth remains fixed. In other words, there is no constant data pool, where each target (bandwidth) uniquely corresponds to its exclusive features (traffic statistics). To overcome this conflict, we propose a dual-stage training process, as delineated in Figure 3.

Simple regressor We adopt a simple pure multilayer perceptron (MLP)-based NN with four fully connected layers as the base model. 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 long (600 ms) and short (60 ms) monitoring intervals, similar to those proposed by *MMSys2024 Challenge* [31][4]. The output is basically a scalar bandwidth estimate (*bps*). During training, we intend to refrain from implementing any sophisticated tricks (coherent to "a simple regressor") except for a cosine annealing schedule [42] 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 I.

Offline learning (first training stage) 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, relaxed "estimation" (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. Consequently, we reproduce the behavior of the ideal bandwidth estimator, and since the environment is determinate, the fixed BWE and its resulting statistics consistently match, enabling the regressor to be trained following the conventional supervised learning procedure. In this way, 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 (analogous to imitation learning).

Online learning (second training stage) We then 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 a BWE and compare with the ground truth to calculate the 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 alone is insufficient. Therefore, we aim to adapt the regressor on the fly, mastering the nuanced relationship between optimal BWE and traffic dynamics. Additionally, to avoid overly leaning 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. Each environment is initiated with a random trace drawn from the training trace pool, and upon consuming a trace, the regressor randomly selects a new trace to initialize the next environment, with a higher probability assigned to those not previously chosen.

It is crucial to reiterate that our solution is solely dedicated to precise BWE, bypassing the complexity and constraints associated with RL. Diametrically opposed to regression, RL algorithms 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 example, the offline RL models in [31] are specifically designed to optimize audio and video quality.

V. Experiment Setup

In this section, we delineate the traces and the associated dataset utilized throughout our work and then present the RL models for the sake of comparison.

A. Traffic trace & Dataset

The *AlphaRTC-Gym* platform requires traffic traces to emulate real-world environments. Each trace comprises a time series of bandwidth profile (along with other auxiliary information including round-trip-time and loss rate), which determines the available link capacity that governs the transmitted traffic. In our work, we employ traces from multiple

[4]Notes: 1) Reference: https://www.microsoft.com/en-us/research/academic-program/bandwidth-estimation-challenge/data/. 2) We opt not to use the entire feature set, as some features, such as those related to packet probability, are considered trivial for our regressor.

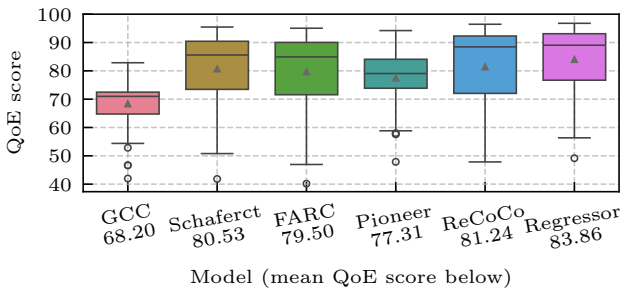

Fig. 4: Performance evaluation: QoE scores of all models.

sources and meticulously split them into training and test sets to circumvent potential commonalities of environments between datasets, as a model is always trained on historical traces 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* [31], collected from massive video-calls using *Microsoft Teams* between testbed nodes or during emulated sessions. This dataset is the only one that offers the behaviors of various BWE policies alongside audio/video qualities as the required rewards, i.e., experts from which offline RL models aim to learn. Besides this, the training set is selected also because the corresponding test traces referenced in the Challenge were not publicly released. Notably, the training set (is supposed to) covers a diverse network conditions with a wide range of bandwidth profiles 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 leverage a diverse collection of traces (163 in total) recorded from real-world scenarios across different network connections (wireless/wired), times, and locations [25], [45]–[47]. We choose the test set to utterly distinguish from the training set, aiming to evaluate the model transferability/generalizability over diverse real-world networks. Whenever necessary, during either training or testing, a given trace acts as the input of *Gym*, defining a particular environment and representing the bottleneck bandwidth availability between the sender and the receiver in an RTC session.

B. Comparative RL model

In order to comprehensively validate the effectiveness of our regressor, we compare with multiple SOTA RL-based bandwidth estimators[5], that outshine the majority of existing works — one online RL model that is tailored for CC and demonstrates heightened performance: ReCoCo [55], and three offline RL models that win the 2^{nd} *ACM MMSys2024 BWE Challenge*: Schaferct [56], FARC [33], Pioneer [34]. Additionally, we also consider GCC, the most well-known heuristic approach widely adopted by modern applications (e.g., Google Meet), as a simple baseline. Notably, we do not consider alternative regressors, as the MLP already represents the simplest regressor capable of continuous online training. 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 serves 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, as indicated in Section II-C.

[5]Notes: 1) Certain existing models like BoB [28] commonly used for comparison in many prior works, are relatively outdated and thus omitted hereafter; 2) Some RL-based frameworks, such as Loki [48] and OnRL [49], target different objectives; 3) Few purportedly competitive BWE solutions [50], [51] exist but are close-sourced and thereby non-reproducible; 4) Other ML/DL models that are designed for bandwidth estimation [52]–[54] 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.

VI. Experimental result

In this section, we present the outcomes, analyzing model performance, conducting an ablation test, and discussing the limitations of RL-based approaches.

A. Model performance

The box plots in Figure 4 outline the performance. As anticipated, the traditional heuristic GCC approach struggles to compete against machine learning (ML)-based models, delivering an average QoE score of mere 68.2. Among offline RL models, Schaferct and FARC manifest comparable performance, both yielding mean scores around 80, while Pioneer, despite producing the lowest QoE score, demonstrates relatively steady behavior, signified by its narrower distribution with only few instances scoring below 60. As for the online RL model ReCoCo, we observe an enhanced result, potentially benefiting from its ability to interact dynamically with the *Gym* platform. Remarkably, our regressor achieves the best performance with the highest mean QoE score of nearly 84 and without compromising certain traces to generate multiple scores lower than 60. Beyond the fact that the performance disparity can be validated by statistical tests, even if our regressor does not yield superior performance but merely remains on par with RL-based approaches, we still advocate for it owing to its unique advantages, such as temporal efficiency. Additionally, our preliminary results substantiate a consistently enhanced perceptual (audiovisual) quality, which we reserve for future work.

To provide a more intuitive comparison, we showcase the performance evolution, i.e., the time series of actual bandwidth, BWE, and receiving rate, of each methodology for an example trace in Figure 5. GCC exhibits a typical sawtooth pattern, gradually increasing its estimation until congestion occurs. RL-based solutions reveal various behaviors that are either aggressive, conservative, or volatile. While their general progressions ostensibly track the network oscillation, they are 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. Remarkably, our regressor, as the runner-up, yields a ratio of 13.8% thanks to

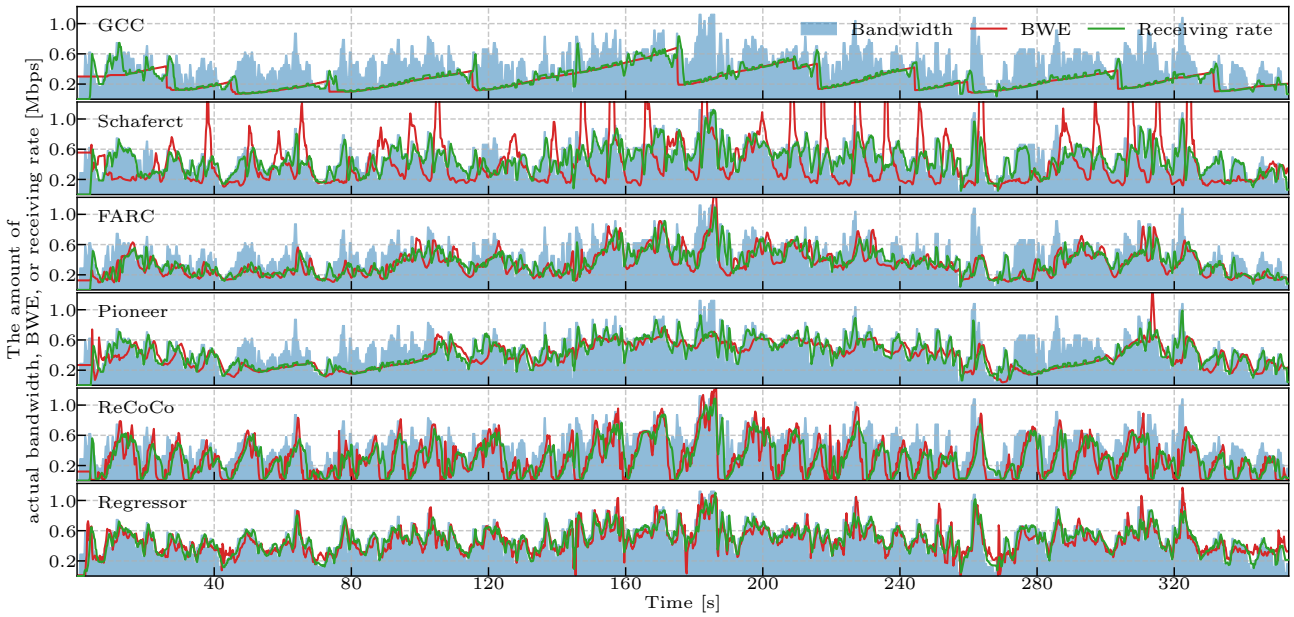

Fig. 5: Behaviors of different bandwidth estimators for an example traffic trace.

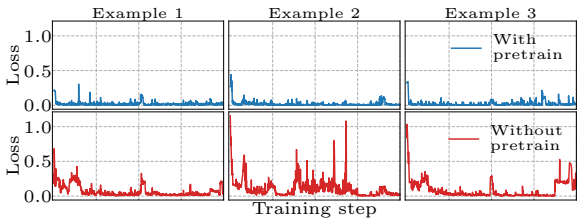

Fig. 6: Loss evolution with/without pretraining.

its deliberate compliance with the constraint-loosened bandwidth targets. Additionally, the resulting ratios are generally above 20% for all RL models.

B. Ablation study

Given the simplicity of our framework, which relies solely on a basic regressor, we identify two possible ablation experiments — removing either online or offline learning.

On one hand, it is conceivable that pure offline learning without online tuning is inadequate, since the model is trained in a hard-coded way with pre-collected data (supervised learning), but estimation errors are inevitable and will propagate in reality, leaving traces that fundamentally deviate from the ideal bandwidth estimator (Prophet that trains the regressor offline), as elucidated in Section IV.

On the other hand, to rigorously verify that online learning alone may not fulfill the task, we analyze the pattern of training losses during the online stage instead of naively assessing QoE results. In particular, we perform 3 training trials, each featuring a different initialization of parameters and training set, and involving a regressor with or without the offline pretraining phase. We observe the progression of losses over 4000 training steps and show the results in Figure 6. Accordingly, the regressor with pretraining not only exhibits comparatively stable behavior but also commences with a minor loss, reflecting a reliable training process and the efficacy of the pretrained model. Conversely, the regressor devoid of offline pretraining produces substantially fluctuated losses, with the last example even failing to reach convergence. This is rooted in the fact that the model with totally randomly initialized parameters is initially trained on and thus only fits the first fixed batch of 8 traces, and consequently, it has to re-adapt to new environments when the training trace shifts (as denoted by the intermediate spikes among losses). As a result, it is a huge waste of time even if the model can eventually converges, and on top of that, the risk of training instability that cannot guarantee a successful model is not acceptable. In this context, the pretrained offline model, which has already acquired a high degree of correlations between traffic statistics and the desired BWE, serves as an optimal start for online learning, stabilizing and expediting the whole training process.

C. Drawback of RL

Herein, we elaborate on the three intrinsic flaws of RL-based solutions mentioned previously.

Model generalizability Online RL models are inherently environmentally dependent, as they are trained to earn high rewards in the specific environments (emulation spaces and training traces), which commonly 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, evolving network infrastructures, and 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. In accordance to the results in Figure 7 (left), ReCoCo undergoes a significant ascent with a performance gain of roughly 4 in terms of 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 phenomena 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, i.e., accurately estimating bandwidth, ensuring unbiased treatment of varying network conditions, which augments generalizability and maintains consistent performance.

Limitation of experts Offline RL approaches rely on pre-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 the three offline models using an advanced expert, Prophet — the perfect bandwidth estimator. As illustrated in Section III, Prophet proves to be a promising candidate capable of approaching the theoretical performance cap of CC in RTC. For the purpose of model training, when generating the dataset for the regressor during offline learning, we also compute a reward [55] for each sample based on the optimal BWE. Consequently, we derive the Prophet-based training traces with data structures akin to the original ones, except that the ideal estimations with their corresponding traffic statistics and the network-related rewards substitute present experts' behaviors and video/audio qualities, respectively. Figure 7 (right) exhibits 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, in turn, 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. In other words, superior instructors are more likely to cultivate high-performing learners.

Time consumption In the contemporary realm of 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 (spanning days in some cases), as they commonly necessitate a large data pool 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

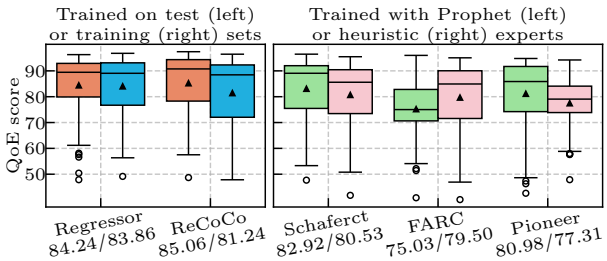

Fig. 7: Performance comparisons in different scenarios (the right one in each box plot duo is the original performance indicated in Figure 4).

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.

VII. Discussion

To further elucidate our findings, we provide in-depth analyses through the following self-questioning and answering.

Why are RL-based approaches lagging? Besides the previously-discussed shortcomings, we posit that the reward encompassing multifarious aspects (e.g., network metrics or multimedia content qualities) introduces misleading noises while steering the model. Reward design constitutes a crucial element of RL, typically integrating diverse components to ensure the model is optimized holistically rather than being biased toward specific directions that may lead to unintended behaviors in real-world deployments. However, the involving components are not necessarily correlated or comparable, e.g., delays and packet losses may arise from excessive traffic, which, paradoxically, could induce a high receiving rate, rendering the reward seemingly acceptable. RL models are propelled to derive reward-wise optimal policies, yet the reciprocally constraining factors embedded lead to suboptimal policies that fail to align with the intended objective. For example, the enhanced performance of offline models trained with Prophet (Figure 7) remains inferior to our regressor, despite utilizing the same expert. This disparity stems from the merit of an explicit target (BWE) in our regressor, as opposed to the demerit of the entangled reward of RL models, which could occasionally obscure the optimization objective and hinder effective policy learning.

Why does a simple regressor suffice? Regression problems are naturally more tractable than RL-based optimization tasks. In other words, a regressor is easier to perform accurate BWE than an RL model to discover the optimal policy. The primary objective of our regressor is solely to accurately estimate the current bandwidth, mirroring the functionality of Prophet. By leveraging supervised learning on extensive

trace data, the regressor effectively captures the intricate relationships between bandwidth and traffic statistics across diverse network conditions, thereby ensuring reliable performance and generalizing over unseen environments. Unlike RL, which is inherently complicated and prone to instability, regression offers a straightforward yet well-established alternative that bypasses the unpredictability, tuning difficulties, and computational overhead associated with RL.

What is the underlying logic of the regression-based BWE in CC? The paradigm operates on a fundamentally different principle from traditional approaches. Instead of reacting to congestion after its onset, our regressor proactively estimates the available bandwidth to guide transmission decisions in real time. By accurately predicting current network conditions, the regressor enables the sender to transmit an appropriate volume of traffic — augmenting network utilization while preventing excessive load, which ensures that traffic remains within the network capacity, effectively mitigating congestion (i.e., congestion avoidance) rather than merely responding to it. In contrast to conventional CC mechanisms, which rely on feedback loops and reactive adjustments, regression-based BWE optimizes flow control through direct, data-driven estimation, enhancing both efficiency and stability in dynamic network environments.

What if BWE were the sole factor in the RL reward? Since accurate BWE represents the decisive factor for CC in RTC, it is logical to design the RL reward exclusively centered on BWE, leveraging the prowess of RL. However, in spite of being plausible, such an 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.

Are there any new potentials? Several promising directions can be explored: *1)* The performance of BWE can be further improved with a more sophisticated regressor (e.g., potent DL models specifically designed for time series forecasting) and comprehensive feature engineering; *2)* While the operation of CC for RTP traffic differs from other protocols, the fundamental principle of BWE and rate adaptation remains universal across networking systems, which offers the prospect of extending our framework to broader applications; *3)* Beyond data-driven methodologies, we should also focus on the root causes of network congestion, which allows us to incorporate prior domain knowledge to guide the design of efficient BWE algorithms, striking a balance between empirical learning and theoretical foundations.

VIII. Conclusion

In this paper, we refute the trend of RL-based BWE in CC for RTC, arguing that RL is not the mandatory path while BWE denotes the deterministic factor. We first highlight the critical role of accurate BWE, which unlocks the potential to attain the theoretical performance ceiling of CC algorithms. To this end, we formulate an irregular supervised learning problem and propose a simple NN-based regressor with two training stages to accommodate the feedback loops in CC. In our work, we leverage the widely adopted *AlphaRTC-Gym* emulation platform with ample traffic traces to establish various network environments. Moreover, we compare our solution against a heuristic baseline and multiple SOTA RL algorithms. In consequence, our regressor demonstrates superior QoE performance, and we conduct a series of experiments to illustrate the disadvantages of RL. Lastly, we elaborate on the work to further discuss in detail the underlying logic and potentials, paving the way for a more reliable and effective BWE solution in CC. Future work could consist of integrating our regressor into real-world deployments.

Acknowledgment

This work was sponsored by Cisco Systems Inc., and the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART", Focused Project R4R). This work was supported by the SmartData@PoliTO center on Big Data and Data Science, and the computational resources provided by hpc@polito (http://www.hpc.polito.it).

References

[1] A. Nistico, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio, "A comparative study of rtc applications," in *2020 IEEE International Symposium on Multimedia (ISM)*, pp. 1–8, IEEE, 2020.

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

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

[4] R. Frederick, S. L. Casner, V. Jacobson, and H. Schulzrinne, "RTP: A Transport Protocol for Real-Time Applications." RFC 1889, Jan. 1996.

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

[6] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 52–57, 2020.

[7] W. Wei, H. Gu, and B. Li, "Congestion control: A renaissance with machine learning," *IEEE network*, vol. 35, no. 4, pp. 262–269, 2021.

[8] H. Jiang, Q. Li, Y. Jiang, G. Shen, R. Sinnott, C. Tian, and M. Xu, "When machine learning meets congestion control: A survey and comparison," *Computer Networks*, vol. 192, p. 108033, 2021.

[9] S. Abbasloo, C.-Y. Yen, and H. J. 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*, pp. 632–647, 2020.

[10] J. Eo, Z. Niu, W. Cheng, F. Y. Yan, R. Gao, J. Kardhashi, S. Inglis, M. Revow, B.-G. Chun, P. Cheng, and Y. Xiong, "Opennetlab: Open platform for rl-based congestion control for real-time communications," in *Proceedings of the 6th Asia-Pacific Workshop on Networking*, pp. 70–75, 2022.

[11] C.-Y. Yen, S. Abbasloo, and H. J. Chao, "Computers can learn from the heuristic designs and master internet congestion control," in *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 255–274, 2023.

[12] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[13] R. F. Prudencio, M. R. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy, review, and open problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[14] W. Zhang, X. Tao, and J. Wang, "Naorl: Network feature aware offline reinforcement learning for real time bandwidth estimation," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 326–331, 2024.

[15] J. Du, C. Zhang, S. He, and W. Qu, "Learning-based congestion control assisted by recurrent neural networks for real-time communication," in *2023 IEEE Symposium on Computers and Communications (ISCC)*, pp. 323–328, IEEE, 2023.

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

[17] N. Smirnov and S. Tomforde, "Real-time rate control of webrtc video streams in 5g networks: Improving quality of experience with deep reinforcement learning," *Journal of Systems Architecture*, vol. 148, p. 103066, 2024.

[18] Z. Sarker, C. Perkins, V. Singh, and M. Ramalho, "Rtp control protocol (rtcp) feedback for congestion control," *Internet RFC*, no. 8888, 2021.

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

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

[21] D. Vucic and L. Skorin-Kapov, "The impact of packet loss and google congestion control on qoe for webrtc-based mobile multiparty audiovisual telemeetings," in *MultiMedia Modeling: 25th International Conference, MMM 2019, Thessaloniki, Greece, January 8–11, 2019, Proceedings, Part I 25*, pp. 459–470, Springer, 2019.

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

[23] L. De Cicco, G. Carlucci, and S. 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*, pp. 21–26, 2013.

[24] S. Zhang, W. Lei, W. Zhang, and Y. Guan, "Congestion control for rtp media: A comparison on simulated environment," in *International Conference on Simulation Tools and Techniques*, pp. 43–52, Springer, 2019.

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

[26] 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.

[27] B. Wang, Y. Zhang, S. Qian, Z. Pan, and Y. Xie, "A hybrid receiver-side congestion control scheme for web real-time communication," in *Proceedings of the 12th ACM Multimedia Systems Conference*, pp. 332–338, 2021.

[28] A. Bentaleb, M. N. Akcay, M. Lim, A. C. Begen, and R. Zimmermann, "Bob: Bandwidth prediction for real-time communications using heuristic and reinforcement learning," *IEEE Transactions on Multimedia*, vol. 25, pp. 6930–6945, 2022.

[29] H. Li, B. Lu, J. Xu, L. Song, W. Zhang, L. Li, and Y. Yin, "Reinforcement learning based cross-layer congestion control for real-time communication," in *2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 01–06, IEEE, 2022.

[30] F. Peng, B. Lu, L. Song, R. Xie, Y. Liu, and Y. Chen, "Pacc: Perception aware congestion control for real-time communication," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 978–983, IEEE, 2023.

[31] S. Khairy, G. Mittag, V. Gopal, F. Y. Yan, Z. Niu, E. Ameri, S. Inglis, M. Golestaneh, and R. Cutler, "Acm mmsys 2024 bandwidth estimation in real time communications challenge," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 339–345, 2024.

[32] Q. Tan, G. Lv, X. Fang, J. Zhang, Z. Yang, Y. Jiang, and Q. Wu, "Accurate bandwidth prediction for real-time media streaming with offline reinforcement learning," in *Proceedings of the 15th ACM Multimedia Systems Conference*, MMSys '24, (New York, NY, USA), p. 381–387, Association for Computing Machinery, 2024.

[33] E. Çetinkaya, A. Pehlivanoglu, I. U. Ayten, B. Yumakogullari, M. E. Ozgun, Y. K. Erinc, E. Deniz, and A. 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*, pp. 388–393, 2024.

[34] B. Lu, K. Wang, J. Xu, R. Xie, L. Song, and W. Zhang, "Pioneer: Offline reinforcement learning based bandwidth estimation for real-time communication," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 306–312, 2024.

[35] A. Gottipati, S. Khairy, G. Mittag, V. Gopal, and R. Cutler, "Real-time bandwidth estimation from offline expert demonstrations," *arXiv preprint arXiv:2309.13481*, 2023.

[36] A. Gottipati, S. Khairy, Y. Hosseinkashi, G. Mittag, V. Gopal, F. Y. Yan, and R. Cutler, "Balancing generalization and specialization: Offline metalearning for bandwidth estimation," *arXiv preprint arXiv:2409.19867*, 2024.

[37] A. Gottipati, S. Khairy, G. Mittag, V. Gopal, and R. Cutler, "Offline to online learning for real-time bandwidth estimation," 2024.

[38] S. A. George and V. Joseph, "Optimizing bandwidth sharing for real-time traffic in wireless networks," in *ICC 2023-IEEE International Conference on Communications*, pp. 3199–3204, IEEE, 2023.

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

[40] T. Song, P. Garza, M. Meo, and M. M. Munafò, "Dex: Deep learning-based throughput prediction for real-time communications with emphasis on traffic extremes," *Computer Networks*, vol. 249, p. 110507, 2024.

[41] M. Carrascosa and B. Bellalta, "Cloud-gaming: Analysis of google stadia traffic," *Computer Communications*, vol. 188, pp. 99–116, 2022.

[42] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[43] J. Xu, Z. Li, B. Du, M. Zhang, and J. Liu, "Reluplex made more practical: Leaky relu," in *2020 IEEE Symposium on Computers and communications (ISCC)*, pp. 1–7, IEEE, 2020.

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

[45] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li, and J. 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*, pp. 34–47, Springer, 2019.

[46] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3g networks: Analysis and applications," in *Proceedings of the 4th ACM Multimedia Systems Conference*, pp. 114–118, 2013.

[47] J. Van Der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "Http/2-based adaptive streaming of hevc video over 4g/lte networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[48] H. Zhang, A. Zhou, Y. Hu, C. Li, G. Wang, X. Zhang, H. Ma, L. Wu, A. Chen, and C. 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*, pp. 775–788, 2021.

[49] H. Zhang, A. Zhou, J. Lu, R. Ma, Y. Hu, C. Li, X. Zhang, H. Ma, and X. Chen, "Onrl: Improving mobile video telephony via online reinforcement learning," in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pp. 1–14, 2020.

[50] J. Du, C. Zhang, T. Tang, and W. 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)*, pp. 1–10, IEEE, 2024.

[51] S. Sunderrajan and L. Liu, "Optimizing rtc bandwidth estimation with machine learning," 2024. Accessed: 2025-02-11.

[52] C. Yue, R. Jin, K. Suh, Y. Qin, B. Wang, and W. Wei, "Linkforecast: Cellular link bandwidth prediction in lte networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1582–1594, 2017.

[53] L. Mei, R. Hu, H. Cao, Y. Liu, Z. Han, F. Li, and J. Li, "Realtime mobile bandwidth prediction using lstm neural network and bayesian fusion," *Computer Networks*, vol. 182, p. 107515, 2020.

[54] T. Song, P. Garza, M. Meo, and M. M. Munafò, "Throughput prediction in real-time communications: Spotlight on traffic extremes," in *2024 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–7, IEEE, 2024.

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

[56] Q. Tan, G. Lv, X. Fang, J. Zhang, Z. Yang, Y. Jiang, and Q. Wu, "Accurate bandwidth prediction for real-time media streaming with offline reinforcement learning," in *Proceedings of the 15th ACM Multimedia Systems Conference*, pp. 381–387, 2024.