Accurate Bandwidth Prediction for Real-Time Media Streaming with Offline Reinforcement Learning

Qingyue Tan[†§], Gerui Lv[†§], Xing Fang[¶§], Jiaxing Zhang[†§], Zejun Yang[†§], Yuan Jiang[†§], Qinghua Wu[†§]

[†]Institute of Computing Technology, Chinese Academy of Sciences
[¶]Institute of Automation, Chinese Academy of Sciences
[§]University of Chinese Academy of Sciences

Abstract

In real-time communication (RTC) systems, accurate bandwidth prediction is crucial for encoding and transmission strategies to optimize users' quality of experience (QoE) in various network environments. In this paper, we propose an offline reinforcement learning (RL) method to predict bandwidth for RTC video streaming. We use a representative algorithm, named Implicit Q-Learning (IQL), to train the model. To improve the performance, we carefully preprocess the given dataset and redesign the neural network structure and the reward function. Ablation studies are performed to verify our design choices. Furthermore, compared to a baseline method and six behavior policies, our method reduces the mean squared error (MSE) by 18%-22%, demonstrating high prediction accuracy. Our proposed method won the first prize in ACM MMSys 2024 Grand Challenge on Offline Reinforcement Learning for Bandwidth Estimation in Real Time Communications. The source code is available at https://github.com/n13eho/Schaferct.

CCS Concepts: **• Networks → Network protocols**; **• Computing methodologies → Reinforcement learning**.

Keywords: Real-time Communication, Bandwidth Prediction, Offline reinforcement learning

ACM Reference Format:
Qingyue Tan[†§], Gerui Lv[†§], Xing Fang[¶§], Jiaxing Zhang[†§], Zejun Yang[†§], Yuan Jiang[†§], Qinghua Wu[†§]. 2024. Accurate Bandwidth Prediction for Real-Time Media Streaming with Offline Reinforcement Learning. In *ACM Multimedia Systems Conference 2024 (MMSys '24), April 15–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3625468.3652183

Corresponding author: Qinghua Wu. Email: wuqinghua@ict.ac.cn.

1 Introduction

In recent years, real-time communication (RTC) has emerged as a crucial technological advancement and found widespread applications, including low-latency live streaming [28, 33–35], video conferencing [5, 23] and cloud gaming [18, 29]. RTC systems aim to optimize the Quality of Experience (QoE) for users by delivering high-quality video and audio and ensuring seamless communication processes (e.g., avoiding video stalling).

To achieve this goal, existing RTC systems, such as WebRTC [21], adaptively adjust the video quality (e.g., encoding bitrate) to dynamic network conditions based on bandwidth prediction. Mainstream bandwidth prediction methods can be classified into two categories: *(i)* heuristic and *(ii)* learning-based. One of the commonly used heuristic bandwidth prediction algorithms is Google Congestion Control (GCC) [3] in the WebRTC framework. GCC primarily predicts bandwidth by monitoring the Round-Trip Time (RTT) variations of the link. Although GCC demonstrates its high sensitivity for proactively avoiding congestion, the complex and variable nature of real-world RTC streaming can interfere with GCC's adaptability. To address this issue, previous studies have proposed learning-based methods to optimize QoE for real-time video streaming in a data-driven way. Examples include Pensieve-V [17], which utilizes online reinforcement learning (RL), and Concerto [35], an imitation learning (IL)-based approach.

Nevertheless, both online RL and IL methods rely on interacting with the environment, thereby requiring a long time to fully explore the solution space in the training phase. Worse still, training learning-based models directly in RTC production systems is costly and impractical. This is because the model's performance is unstable until the training converges, resulting in unpredictable QoE degradation for real users. Therefore, existing methods prefer to develop a simulator as the training environment [17, 35], which however is difficult to faithfully capture the dynamics of real-world video systems [1, 2, 16, 30].

As a solution, we in this paper propose a bandwidth prediction method for RTC streaming based on *offline (data-driven)* RL techniques. Offline RL utilizes a pre-collected and static offline dataset to search for policies that optimize QoE. In this way, the model can leverage diverse experiences collected by arbitrary other policies, without online interactions with the real environment [6, 14, 31].

Based on a real-world dataset of observed network dynamics with objective metrics reflecting user-perceived audio/video quality provided by Microsoft Teams, we take part in the ACM MMSys 2024 Grand Challenge [19] using Offline RL techniques to address the difficulties of bandwidth prediction in RTC systems. For effective training, we first preprocess the training dataset, including filling in the missing or abnormal values and carefully splitting the training and test sets (§2). Using a representative offline RL algorithm, namely Implicit Q-Learning (IQL) [12], we redesign the neural network (NN) structure and the reward function (§3). Extensive experiments are conducted to identify the design choice of our methods, including the training algorithm, the NN structure, and the reward setting (§5). Our local evaluation results indicate that our model outperforms the baseline model and all original behavior policies in terms of prediction metrics (§4).

The Grand Challenge includes a two-stage evaluation [11]. The first stage was conducted on an emulation platform where ground truth information is available, and the second stage was conducted on an intercontinental testbed, where audio/video calls are made between random pairs of nodes that are geographically distributed across the globe. Our proposed method ranked first in both stages and eventually won the first prize of the Grand Challenge.

2 Data Processing

2.1 Dataset Description

The training and evaluation datasets provided by the Grand Challenge are collected from audio/video peer-to-peer Microsoft Teams calls across the world. There are 18,859 sessions in the training dataset and 9,405 sessions in the evaluation dataset, and each session corresponds to one audio/video call, containing thousands of sequences of the following fields: *(i)* 150-dimensional observation, *(ii)* estimated bandwidth from 6 different behavior policies, *(iii)* objective audio quality, and *(iv)* objective video quality. The quality indicates the Mean Opinion Score (MOS) $\in [0, 5]$, with a score of 5 being the highest. The actual link capacity (also referred to as the ground truth) of every sequence is provided in the evaluation dataset. We use about 10% of the training dataset to train our model (§2.3), use all the evaluation datasets to evaluate our model (§4 and §5).

A 150-dim observation contains 15 metrics related to network conditions and packet information, including receiving rate, delay, packet jitter, packet loss ratio, audio/video packet proportion, etc. Each metric has 10 values over the 5 most recent 60ms short monitoring intervals (MIs) and the 5 most recent 600ms long monitoring intervals. Detailed data description can be found in [11].

2.2 Missing Value Filling

At the beginning and end of each session, there is a portion of transitions where the video quality values are missing (i.e., NaN). This is because there are no video data packets in the link during these phases, resulting in undefined video quality. Accordingly, three different methods have been attempted to handle the missing values as follows:

(i) Trace clipping. We remove the data at the beginning of these sessions, as well as the data at the end, where the video quality is NaN. This ensures that only transitions with valid audio and video quality throughout the session are retained in the dataset.

(ii) Zero filling. Zero filling simply assigns these NaN reward signals as 0.

(iii) Average filling. Average filling replaces NaN values with the average of the valid reward signals of the entire trajectory. With this approach, significant reward fluctuations are avoided throughout the entire trajectory, and it helps mitigate the model's tendency to severely overestimate the bandwidth at the beginning of a session. This approach also helps model making the right decisions at the beginning of a session.

While other methods could be considered, such as using the audio quality to fill in the missing video quality, these methods are not always feasible. In some cases, both video and audio qualities may be missing. In addition, the distributions of the two qualities are usually different. Therefore, simply copying values from one to the other is inappropriate.

We eventually chose the average filling method to replace missing values with an average value of all transitions, as discussed in §5.1.

2.3 Training Set Splitting

In the training set of offline RL, the diversity of decision policies significantly impacts the training effectiveness. The provided training set includes a total of six different behavior policies. Accordingly, we **randomly** select 300 sessions for each policy type, resulting in a total of 1800 sessions to compose the dataset used for training. Not all the available sessions are utilized due to the constraint of GPU memory, as discussed in §6.

3 Design

3.1 Learning Algorithm

Preliminaries. In RL, the process by which an agent interacts with the environment is typically described as a Markov Decision Process (MDP) $< \mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma >$, where $\mathcal{S}$ represents the set of states and $\mathcal{A}$ represents the set of actions; $P(\cdot|s,a) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0,1]$ represents the dynamic transition model, $r(s,a)$ represents the reward function, $\rho_0 : \mathcal{S} \rightarrow [0,1]$ is the initial state distribution, and $\gamma \in [0,1]$ is the discount factor. The goal of RL is to learn a policy $\pi(a|s) : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expectation of the sum of discounted rewards, also known as the return:

$$\max_{\pi} \mathcal{J}_{\rho_0}(\pi) = \mathbb{E}_{s\sim\rho_0 a_t\sim\pi(s_t)}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s]. \quad (1)$$

In offline RL, the agent can only use a pre-collected dataset $\mathcal{D} = (s, a, r, s')$ for training and can not interact with the environment to obtain new experiences.

Choice of the algorithm. Three representative offline RL algorithms are off-the-shelf for our problem: *(i)* Twin delayed deep deterministic policy gradient algorithm plus behavior cloning (*TD3_BC*) [7], *(ii)* Conservative Q-Learning (*CQL*) [13], and *(iii)* Implicit Q-Learning (*IQL*) [12].

TD3_BC is a policy-constrained method. It simply adds a behavior cloning term to the policy update of an online RL algorithm TD3 [8]. Different from the policy constraint in TD3_BC, CQL places the penalty on the Q function (i.e., state-action value function), which aims to learn a conservative Q function such that the expected value of a policy under this Q function lower bounds its true value. IQL is a representative of SARSA-type learning [24], implicitly approximating policy improvement by treating the state value function as a random variable. Though both TD3_BC and CQL alleviate out-of-distribution (OOD) action sampling, their performance still be harmed due to the potential distributional shift [22]. In contrast, IQL utilizes expectile regression to achieve the in-sample training, avoiding errors caused by the distributional shift. Correspondingly, IQL is promising to learn the Q function more accurately and obtain a better final policy. Therefore, we choose the IQL algorithm to train the model in this work. The evaluation results in §5.4 also demonstrate the superior performance of IQL over the other two algorithms.

IQL algorithm. IQL aims to approximate the value function $V(s) : \mathcal{S} \rightarrow \mathbb{R}$ and the state-action value function $Q(s,a) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ in the *policy evaluation* stage, and obtain the final policy $\pi(s)$ in the *policy extraction* stage.

In the policy evaluation stage, IQL uses the expectile regression update method to approximate the optimal value function $V(s)$ with the asymmetric loss L_2^τ. This can be achieved by minimizing the following loss on $V_\psi(s)$:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^\tau\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right], \quad (2)$$

where $L_2^\tau(u) = |\tau - \mathbf{1}(u < 0)|u^2$ is an asymmetric l_2 loss, and $Q_{\hat{\theta}}(s,a)$ is the target state-action network. The state-action value function $Q_\theta(s,a)$ is updated by minimizing the temporal difference (TD) loss :

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(r(s,a) + \gamma V_\psi(s') - Q_\theta(s,a)\right)^2\right]. \quad (3)$$

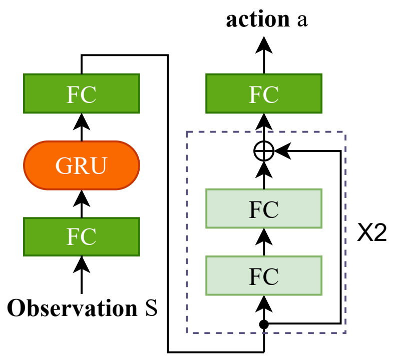


Figure 1. Structure of actor network

In the policy extraction stage, IQL minimizes the loss for optimizing the final policy $\pi_\phi(s)$ is:

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\exp\left(\beta\left(Q_{\hat{\theta}}(s,a) - V_\psi(s)\right)\right)\log\pi_\phi(a \mid s)\right], \quad (4)$$

where $\beta \in [0,\infty)$ is used to adjust the balance between maximizing Q-values and behavioral cloning (BC) [27]. A smaller β makes the overall policy more biased toward BC, while a larger β tends to maximize the Q function.

3.2 State, Action and Reward

State. We use the original 150-dimensional observations (§2.1) as our input state, keeping its information within the past 5 short and 5 long MIs, so the input state naturally has the temporal correlation information across the time domain. Before being passed to the neural networks, all the input features are normalized to facilitate the model training. Specifically, all feature values are limited to [-10, 10]. For example, the receiving rate is divided by 10^6.

Action. The model directly outputs the predicted bandwidth as the action in each decision period. Note that the output is also normalized, divided by 10^6.

Reward function. For each state-action pair, the dataset provides the audio quality and video quality during each MI. We set the reward function as the weighted sum of these two qualities:

$$r(s,a) = (2-\alpha)q_a + \alpha * q_v, \quad (5)$$

where q_a is audio quality, q_v is video quality, and $\alpha \in [0,2]$ controls the weight of these two qualities, considering that they may not equally contribute to the overall quality. A detailed discussion on identifying α is provided in §5.2.

3.3 Actor Network Architecture

The structure of the actor neural network in our proposed method is illustrated in Figure 1. The input of each state is a 150-dimensional normalized observation containing both short-term and long-term temporal information. The input is then fed into a fully connected (FC) layer with the size of 150x256. Next, a Gated Recurrent Unit (GRU) [4] is introduced to capture the internal temporal information in each state, inspired by Sage [31]. The output then passes through another FC of size 256x256 before entering a Residual Block [9], which mitigates the risk of gradient vanishing and promotes the stability of the model's performance. The tensor output goes into the final FC of size 256x1. After applying the *tanh* activation function, the output action represents the predicted bandwidth in Mbps and is further multiplied by 10^6 to convert to bps. Since *tanh* may output a negative bandwidth value in [-1, 0), we use a clamp function to ensure that the final prediction is not less than 10bps. For more details, please refer to our open-source code.

3.4 Implementation

Our proposed method is implemented based on CORL [26], an open-source Offline Reinforcement Learning library. Hyperparameters in the IQL algorithm are set as default in CORL, i.e., $\tau = 0.7$ (in $L_2^\tau(u)$, see Eq. 2) and $\beta = 3$ (Eq. 4).

4 Evaluation

4.1 Setup

Performance metrics. The best metric to evaluate the performance of our proposed method is the audio and video quality. However, since the quality assessment model is not available, it is not feasible for us to obtain the actual quality after each action. Therefore, we turn to evaluate the **prediction accuracy** of our method. In general, accurate bandwidth prediction is expected to ensure higher audio and video quality [16, 25, 30].

Three metrics are selected to assess the prediction accuracy of all methods, including *(i)* prediction error rate [10, 25], *(ii)* overestimation rate [15], and *(iii)* mean-square error (MSE). Generally, lower values for these three metrics correspond to higher prediction accuracy.

The *error rate* is defined as follows:

$$error\_rate = \frac{1}{n}\sum_{i=1}^{n}\min(1, \frac{|\hat{B}-B|}{B}), \tag{6}$$

where n is the total number of times the model is called in a session, B is the true bottleneck link bandwidth, and $\hat{B}$ is the model output, the predicted value of the link bandwidth.

The *overestimation rate* is a metric that assesses the degree to which the model overestimates the true values of the link, reflecting the model's conservatism or aggressiveness [15, 32]. Overestimating the link bandwidth may directly lead to video rebuffering, while underestimation is relatively safer, causing only a decrease in video quality. In comparison to potential video rebuffering in RTC systems, a more conservative estimate is considered better. The overestimation rate is defined as follows:

$$overest\_rate = \frac{1}{n}\sum_{i=1}^{n}\max(0, \frac{\hat{B}-B}{B}). \tag{7}$$

Finally, *MSE* is commonly used as the loss function in deep learning methods and is defined as:

$$mse = \frac{1}{n}\sum_{i=1}^{n}(\hat{B}-B)^2. \tag{8}$$

Comparison schemes. We use one baseline model [20] provided by the competition committee and six anonymous behavior policies from the original trajectories to compare with our model on the evaluation dataset.

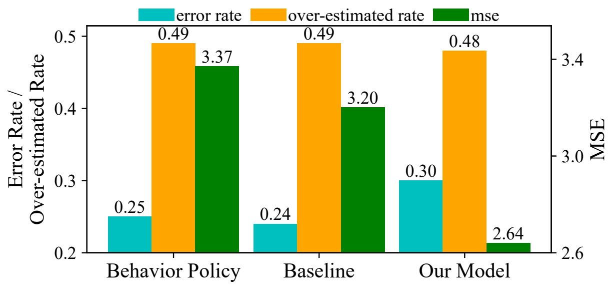


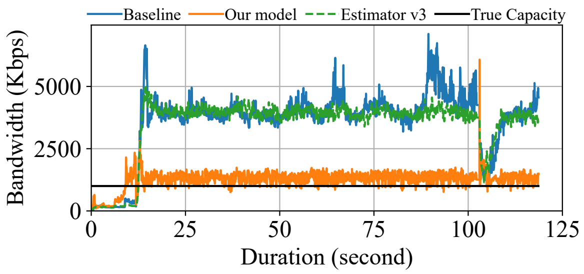
Figure 2. Performance of Our Model

4.2 Performance of Our Model

The performance of our final model on the evaluation dataset is illustrated in Figure 2. In terms of the error rate, our model did not surpass the performance of the behavior policy and baseline. The over-estimated rates of the three models are also similar, with our model slightly lower. However, when considering the MSE, our model exhibits a much smaller mean square error compared to the others, which is 18% and 22% lower than the baseline and behavior policy, respectively.

The two cases in Figure 3 present the superior prediction performance of our model over the baseline and behavior policies (Estimator v1 and Estimator v3). In the first case (Figure 3a), our model demonstrates the ability to make accurate predictions when the behavior policy significantly overestimates the link bandwidth. The baseline model follows the behavior policy to maintain the large overestimation. In contrast, our model closely aligns with the true capacity. In the second case (Figure 3b), the baseline tends to overestimate after the start-up phase. Under these circumstances, our model leans towards alignment with the behavior policy, resulting in more conservative and accurate predictions.

These results indicate that offline RL allows the model to learn from other policies and further outperform the original policies, as reported in [31]. Note that we do not claim that our model is the best one in all scenarios, but it performs well in most cases and achieves the best overall performance. Further discussion is provided in §6.


(a) Case 1: Session 02703

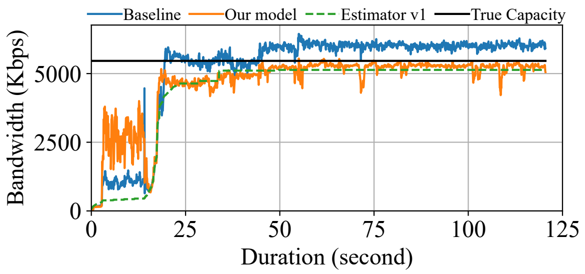

(b) Case 2: Session 05692

Figure 3. Cases of prediction trajectories where our model is more accurate

5 Ablation Study

5.1 Missing Value Filling Methods

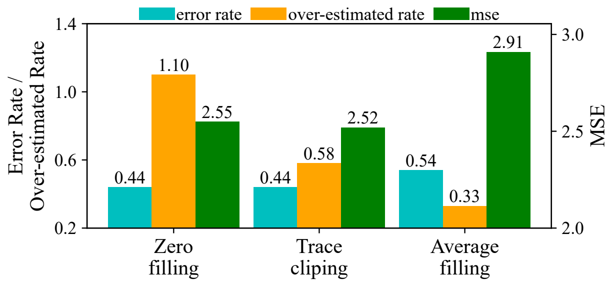


Figure 4. Performance of three different methods

In §2.2, three methods are considered to handle missing values, where both zero filling and average filling are value-replacement methods. Under the same training algorithm and actor-network structure, the comparison results of these three methods on the evaluation set are illustrated in Figure 4. Note that after average filling, the proportion of audio quality and video quality in the reward function is equal.

From the results, average filling only has an advantage in overestimation rate, but we still choose this method. This is because, compared to the other two methods, it retains more complete session information. On the one hand, zero filling introduces a sudden change in the reward trajectory for the entire session, where the undefined reward is simply taken as the worst case (0). As a result, the model may mistakenly consider "good" behavior to be "bad". On the other hand, trace clipping drops all the missing data, making the model unable to learn from actions at the beginning of sessions.

5.2 Weight α in Reward Function

Eq. 5 incorporates a parameter α to adjust the weight of audio and video quality. We evaluate the prediction performance of our model under different settings of *alpha*, ranging from 1.0 to 2.0. As depicted in Figure 5, $\alpha = 1$ causes the highest error rate of 0.54. On the other hand, $\alpha = 2$ produces the highest MSE (2.92). Therefore, we finally chose $\alpha = 1.5$ for training because it demonstrated the lowest error rate and MSE.

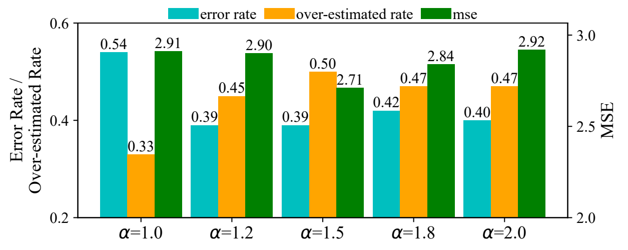


Figure 5. Results of Different α Values

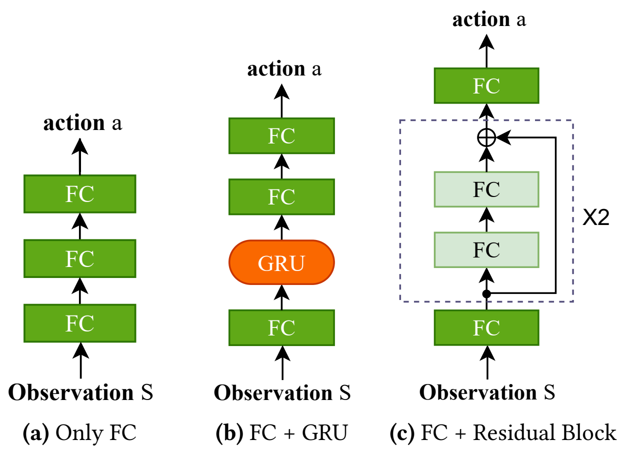


Figure 6. Three different NN architectures

5.3 Actor Network Structure

The NN architecture of the actor is initially designed with just three FC layers. Here, we conduct the following experiments to verify the effectiveness of incorporating GRU and Residual

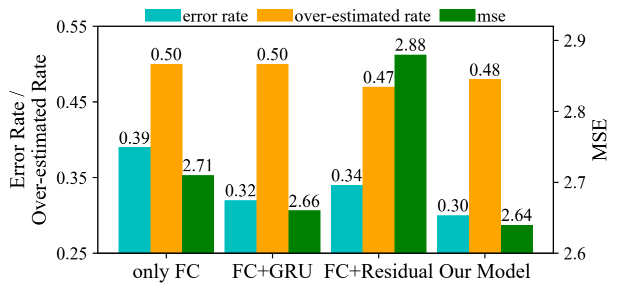


Figure 7. Performance of different NN architectures

Blocks. There are four different design choices, including: *(i)* only three FC layers (Figure 6a), *(ii)* three FC layers with GRU (Figure 6b), *(iii)* three FC layers with two Residual Blocks (Figure 6c), and *(iv)* the final adopted architecture, namely three FC layers with GRU and Residual Blocks, as shown in Figure 1.

Figure 7 presents the evaluation results of four corresponding models. It can be seen that our proposed architecture achieves the lowest error rate (0.3) and MSE (2.64). Removing GRU or Residual Blocks leads to worse prediction performance. Therefore, the actor architecture of our model is necessary for accurate prediction.

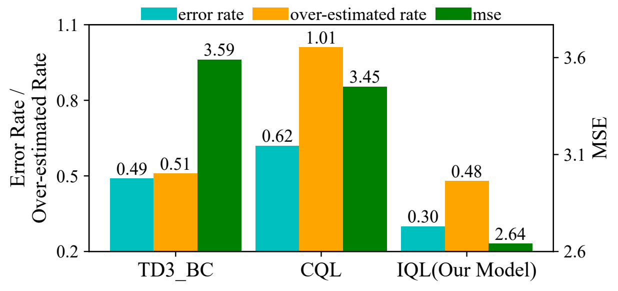


Figure 8. Performance under different offline RL algorithm

5.4 Offline RL Algorithms

Finally, we study the choice of the training algorithm by evaluating the performance of models based on three different algorithms (§3.1): TD3_BC, CQL, and IQL. These evaluations are performed under consistent conditions (i.e., the same training set and input features). As illustrated in Figure 8, the model trained using IQL outperformed the others in terms of all metrics, indicating that IQL is more suitable for training bandwidth prediction models.

6 Limitations

As noted in the Grand Challenge result [11], our model performs well in high-bandwidth, burst-loss, and fluctuating-burst-loss scenarios, but can still be improved in low-bandwidth and fluctuating-bandwidth scenarios. Here, we list possible directions to improve our model further.

Dataset. Out of a total of 18,860 sessions, only 1,800 sessions are used for training due to the hardware constraints (e.g., GPU memory size) in our training environment. Besides, we simply apply random sampling in constructing the training set, but it might be better to consider the network and reward distributions. While the diversity in policy types is ensured, our training set may not represent all states of the video streaming system. Therefore, the performance of our method could be further improved by utilizing more data more carefully. We leave this point as the future work.

Feature engineering. The provided dataset consists of a total of 15 input features. Each feature has 10 numerical values, varying in the length of the monitor interval. We directly use all features in their original form as the input to our model. However, feature engineering techniques can be applied, which require further research.

7 Conclusion

Real-time communication (RTC) video streaming relies on accurate bandwidth prediction to optimize QoE for users. State-of-the-art learning-based methods leverage online reinforcement learning (RL) or imitation learning (IL) algorithms, relying on interacting with the environment in the training process. This nature poses challenges to training and deploying learning-based models in real-world RTC systems.

To address this issue, this paper proposes an offline-RL-based bandwidth prediction method. Specifically, based on a representative algorithm IQL, we redesign the neural network structure and the reward function. Extensive evaluations demonstrate the design choices and predictive performance of our model. Specifically, our model reduces 18%-22% MSE compared to both the baseline and six behavior policies.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback. This work was supported in part by the National Key R&D Program of China (2022YFB2901800) and Natural Science Foundation of China (U20A20180, 62072437).

References

[1] Abdullah Alomar, Pouya Hamadanian, Arash Nasr-Esfahany, Anish Agarwal, Mohammad Alizadeh, and Devavrat Shah. 2023. {CausalSim}: A Causal Framework for Unbiased {Trace-Driven} Simulation. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 1115–1147.

[2] Chandan Bothra, Jianfei Gao, Sanjay Rao, and Bruno Ribeiro. 2023. Veritas: Answering causal queries from video streaming traces. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 738–753.

[3] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2017. Congestion control for web real-time communication. *IEEE/ACM Transactions on Networking* 25, 5 (2017), 2629–2642.

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks

on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[5] Sandesh Dhawaskar Sathyanarayana, Kyunghan Lee, Dirk Grunwald, and Sangtae Ha. 2023. Converge: Qoe-driven multipath video conferencing over webrtc. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 637–653.

[6] Xing Fang, Qichao Zhang, Yinfeng Gao, and Dongbin Zhao. 2022. Offline reinforcement learning for autonomous driving with real world driving data. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 3417–3422.

[7] Scott Fujimoto and Shixiang Gu. 2021. A Minimalist Approach to Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (Eds.). https://openreview.net/forum?id=Q32U7dzWXpc

[8] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*. 1582–1591.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*. Springer, 630–645.

[10] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 97–108.

[11] Sami Khairy, Gabriel Mittag, Scott Inglis, Vishak Gopal, Mehrsa Golestaneh, Ross Cutler, Francis Yan, and Zhixiong Niu. 2024. ACM MMSys 2024 Bandwidth Estimation in Real Time Communications Challenge. arXiv:2403.06324 [cs.NI]

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

[13] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. arXiv:2006.04779 [cs.LG]

[14] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643* (2020).

[15] Gerui Lv, Qinghua Wu, Yanmei Liu, Zhenyu Li, Qingyue Tan, Furong Yang, Wentao Chen, Yunfei Ma, Hongyu Guo, Ying Chen, and Gaogang Xie. 2024. Chorus: Coordinating Mobile Multipath Scheduling and Adaptive Video Streaming. In *The 30th Annual International Conference on Mobile Computing and Networking*. 1–17.

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

[17] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*. 197–210.

[18] Zili Meng, Xiao Kong, Jing Chen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2024. Hairpin: Rethinking Packet Loss Recovery in Edge-based Interactive Video Streaming. In *USENIX NSDI*.

[19] Microsoft. 2024. https://www.microsoft.com/en-us/research/academic-program/bandwidth-estimation-challenge/.

[20] Microsoft. 2024. https://github.com/microsoft/RL4BandwidthEstimationChallenge/tree/main/onnx_models.

[21] WebRTC project. 2024. https://webrtc.org/. Accessed: 2024-02-03.

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

[23] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 953–971.

[24] Gavin A Rummery and Mahesan Niranjan. 1994. *On-line Q-learning using connectionist systems*. Vol. 37. University of Cambridge, Department of Engineering Cambridge, UK.

[25] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*. 272–285.

[26] Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. 2023. CORL: Research-oriented deep offline reinforcement learning library. *Advances in Neural Information Processing Systems* 36 (2023).

[27] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954* (2018).

[28] Haiping Wang, Zhenhua Yu, Ruixiao Zhang, Siping Tao, Hebin Yu, and Shu Shi. 2023. TwinStar: A Practical Multi-path Transmission Framework for Ultra-Low Latency Video Delivery. In *Proceedings of the 31st ACM International Conference on Multimedia*. 9234–9242.

[29] Jiangkai Wu, Yu Guan, Qi Mao, Yong Cui, Zongming Guo, and Xinggong Zhang. 2023. ZGaming: Zero-latency 3D cloud gaming by image prediction. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 710–723.

[30] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 495–511.

[31] Chen-Yu Yen, Soheil Abbasloo, and H Jonathan Chao. 2023. Computers Can Learn from the Heuristic Designs and Master Internet Congestion Control. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 255–274.

[32] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*. 325–338.

[33] Huanhuan Zhang, Anfu Zhou, Yuhan Hu, Chaoyue Li, Guangping Wang, Xinyu Zhang, Huadong Ma, Leilei Wu, Aiyun Chen, and Changhui Wu. 2021. 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*. 775–788.

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

[35] Anfu Zhou, Huanhuan Zhang, Guangyuan Su, Leilei Wu, Ruoxuan Ma, Zhen Meng, Xinyu Zhang, Xiufeng Xie, Huadong Ma, and Xiaojiang Chen. 2019. Learning to coordinate video codec with transport protocol for mobile video telephony. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.