Bandwidth Estimation with Conservative Q-Learning

1st Caroline Chen
Tencent
Shenzhen, China
carolichen@tencent.com

***Abstract*—This research attempts to tackle the prevailing challenges in bandwidth estimation (BWE) for real-time communication systems, with a special emphasis on applying offline reinforcement learning to craft a more accurate neural network for bandwidth estimation than those built using traditional heuristics. The developed model, 'CQLBWE', represents a data-driven approach to BWE, operating offline. The model exploits heuristic-based techniques from the past to formulate a proficient BWE policy. Furthermore, the successful usage of CQLBWE underscores the practicability of deploying offline reinforcement learning algorithms in the field of bandwidth estimation.**

I. Introduction

In the realm of network communications, congestion control is a critical component that ensures efficient and reliable data transmission across networks. The advent of machine learning, particularly reinforcement learning (RL), has opened new avenues for enhancing congestion control mechanisms. This paper introduces a novel approach that employs Conservative Q-Learning, a form of reinforcement learning, to estimate bandwidth within the context of congestion control.

Traditional congestion control algorithms have relied on static rules or heuristics to adjust the sending rate based on network conditions. However, these methods sometimes struggle to adapt effectively to changing conditions. Machine learning, especially RL, addresses this by offering the capability to learn from and adapt to the environment, making it highly suitable for managing the uncertainties and variabilities of network conditions.

Conservative Q-Learning, a variant of Q-learning, excels in balancing exploration with exploitation, ensuring that the learning process is both efficient and robust. By integrating Conservative Q-Learning into congestion control, we aim to create a system capable of dynamically estimating available bandwidth more accurately and adapting its behavior based on real-time network conditions. This approach not only promises to enhance the reliability and efficiency of data transmission but also paves the way for more intelligent and adaptive network management.

This paper begins by exploring the background of congestion control and the role of bandwidth estimation in its mechanisms. It then delves into the principles of reinforcement learning, with a specific focus on Conservative Q-Learning. Following this, we present our methodology for applying Conservative Q-Learning to bandwidth estimation.

In summary, this work contributes to the field of network communications by introducing an offline RL method to congestion control, showcasing the strengths of Conservative Q-Learning for effective bandwidth estimation.

II. Preliminaries

In this section, we provide a foundational understanding of offline reinforcement learning and then delve into specific algorithms: Soft Actor-Critic (SAC) [3] and Conservative Q-Learning (CQL) [4].

A. Offline Reinforcement Learning

Offline reinforcement learning has emerged as a vital solution to the limitations inherent in traditional online reinforcement learning. While online reinforcement learning requires extensive interaction with the environment to learn effective policies—a process that can be resource-intensive and potentially risky—offline reinforcement learning leverages pre-collected data to enable learning without further environmental interaction. This approach is particularly beneficial in scenarios where data collection is costly or dangerous, and where safety and efficiency are critical. It facilitates learning from historical data, simulations, or expert demonstrations, thus reducing the risks and costs associated with exploration in online learning. However, offline RL also presents its own challenges, such as addressing the distribution shift between the learned policy and the policy used to generate the dataset, and effectively learning from a fixed dataset without the opportunity for new data exploration.

Offline reinforcement learning (offline RL) involves an agent learning from a fixed dataset of experiences, denoted as$\mathcal{D}$, without additional interaction with the environment. This method is particularly valuable in scenarios where real-time interaction is impractical, costly, or risky. The dataset $\mathcal{D}$ consists of tuples (s, a, r, s'), where s is the state, a is the action taken, r is the reward received, and s' the subsequent state. The objective is to learn a policy, π, that maximizes the expected return from any given state.

The Bellman equation for the state-action value function $Q(s, a)$ in offline RL is given by:

$$Q(s,a) = \mathbb{E}_{(s',r)\sim\mathcal{D}}\left[r + \gamma \max_{a'} Q(s', a')\right] \tag{1}$$

where γ is the discount factor that balances immediate and future rewards.

B. Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is an off-policy, model-free reinforcement learning algorithm in the context of deep learning and control theory. It is particularly noted for its sample efficiency and stability relative to other algorithms. SAC utilizes the maximum entropy reinforcement learning framework, which not only seeks to maximize expected return but also to maximize entropy. This characteristic encourages the policy to explore more widely, contributing to more robust learning [3].

The Maximum Entropy Objective

In the maximum entropy framework, the objective is to maximize both the expected return and the entropy of the policy. The entropy term ensures sufficient exploration. The objective function is defined as:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))] \quad (2)$$

where π is the policy, s_t and a_t are the state and action at time t, ρ_π is the state-action visitation distribution under policy π, $r(s_t, a_t)$ is the reward function, α is the temperature parameter that determines the relative importance of the entropy term against the reward, and $\mathcal{H}$ denotes the entropy of the policy.

Soft Q-Function and Value Function

In SAC, two types of value functions are defined: the soft Q-function and the soft value function. The soft Q-function, $Q(s, a)$, represents the expected return after taking an action a in state s, and then following policy π:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{P}}[V(s_{t+1})] \quad (3)$$

The soft value function, $V(s)$, is defined as:

$$V(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t, a_t) - \alpha\log\pi(a_t|s_t)] \quad (4)$$

where γ is the discount factor, and $\mathcal{P}$ represents the state transition probability.

Policy and Value Function Updates

The policy is updated by performing gradient ascent on the expected return plus entropy, while the soft Q-function and soft value function are updated to minimize their respective Bellman residuals. The update equations are:

$$\begin{aligned}\nabla_\theta J(\pi_\theta) \approx\ & \mathbb{E}_{s_t\sim\rho_\pi,\epsilon_t\sim\mathcal{N}}[\nabla_\theta\log\pi_\theta(f_\theta(\epsilon_t; s_t)) \\ & \times (Q(s_t, f_\theta(\epsilon_t; s_t)) - V(s_t))] \end{aligned} \quad (5)$$

$$\nabla_\phi J(Q_\phi) \approx \quad (6)$$

$$\nabla_\phi(Q_\phi(s_t, a_t) - (r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{P}}[V_{\bar{\phi}}(s_{t+1})]))^2 \quad (7)$$

where θ and ϕ are the parameters of the policy and value networks, respectively, $\mathcal{N}$ denotes a normal distribution, and f_θ represents the deterministic transformation of the Gaussian noise ϵ_t to the action space.

C. Conservative Q-Learning (CQL)

Conservative Q-Learning (CQL) is a novel off-policy reinforcement learning algorithm, particularly beneficial in offline settings where the overestimation of Q-values due to the distributional shift between the policy and the behavior policy is a significant challenge. CQL aims to learn a conservative estimate of the Q-function that lower-bounds the expected return, thus reducing overestimation and improving performance in offline RL tasks [4].

The CQL Objective

The primary objective of CQL is to minimize the overestimation error of the Q-function while ensuring efficient learning. The CQL objective can be formulated as:

$$J(\theta) = \alpha\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}[Q_\theta(s, a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\mathcal{D}}[Q_\theta(s, a)] \quad (8)$$

where θ are the parameters of the Q-network, $\mathcal{D}$ denotes the offline data distribution, π_θ is the policy, and α is a trade-off parameter that controls the conservatism of the Q-function estimation.

CQL Regularization

A key component of CQL is the introduction of a regularization term in the Q-function update rule. This regularization term penalizes Q-values for actions that deviate significantly from the behavior policy. The regularized Q-function update is:

$$\begin{aligned}\min_\theta \alpha\,&(\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}[Q_\theta(s, a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\mathcal{D}}[Q_\theta(s, a)]) \\ &+ \frac{1}{2}\mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\left(Q_\theta(s, a) - \hat{Q}(s, a, r, s')\right)^2\right]\end{aligned} \quad (9)$$

where $\hat{Q}(s, a, r, s')$ denotes the target Q-value, computed using the Bellman equation.

Policy Improvement in CQL

In CQL, the policy is updated to maximize the conservative Q-function, leading to more robust decision-making, especially in environments with limited or noisy data. The policy improvement step involves performing gradient ascent on the expected Q-value:

$$\nabla_\phi J(\pi_\phi) = \mathbb{E}_{s\sim\mathcal{D}}[\nabla_\phi\mathbb{E}_{a\sim\pi_\phi}[Q_\theta(s, a)]] \quad (10)$$

where ϕ represents the parameters of the policy network.

III. Related Work

A. Traditional Bandwidth Estimation Techniques

Traditional approaches to bandwidth estimation, such as probing, packet pair techniques, and queueing theory methods, have certain limitations. These methods often require detailed knowledge of network parameters and can introduce significant overhead.

B. Reinforcement Learning in Bandwidth Estimation

Reinforcement Learning (RL) has emerged as a powerful tool for network optimization, offering notable improvements in bandwidth estimation and congestion control. A prominent example is the Pensieve system, which employs the Asynchronous Advantage Actor-Critic (A3C) algorithm for adaptive video streaming. Pensieve demonstrates the efficacy of RL through its online training within a network simulator, enabling dynamic bitrate selection from predefined options [1]. Similarly, in the domain of real-time communications, Proximal Policy Optimization (PPO) has been applied to discrete bandwidth estimation tasks. R3Net exemplifies this approach by integrating actual real-time communication (RTC) processes with network simulation for enhanced online RL training. Such advancements underscore RL's adaptability and optimization capacity in complex, dynamic network settings. Additionally, the OpenNetLab initiative has established a comprehensive online RTC platform that allows researchers to rigorously train and evaluate their RL models, facilitating the transition from theoretical research to tangible enhancements in network management [7]. Despite these advances, the inherent complexity of network environments poses significant challenges for accurate simulation. In response, our research seeks to investigate the untapped potential of offline RL to overcome these limitations.

C. Offline Reinforcement Learning in Other Domains

Offline reinforcement learning has been effectively applied in various domains, including robotics and healthcare [9]. The challenges in these domains, such as the need for safety in trial-and-error and the scarcity of data, are analogous to those in bandwidth estimation, making offline RL a promising approach.

IV. Methodology

We utilize Microsoft's testbed dataset, which centers on Microsoft Teams audio/video calls. The dataset includes trajectories and two objective signals for assessing user-perceived audio and video quality. Derived from peer-to-peer Microsoft Teams calls among geographically dispersed nodes, the dataset features diverse Internet Service Providers (ISPs), wired and wireless connections, and employs varied bandwidth estimators such as Kalman-filtering-based methods, WebRTC, and multiple machine learning policies; these bandwidth estimators are seen as the behavior policy in our experiment [2].

At each time step n, the observation vector captures network statistics that describe the bottleneck link between the sender and receiver. As a result, at each time step of each network trajectory, we have a state vector s_t which includes information about receiving rate, number of received bytes, queuing delay, delay, minimum seen delay, delay ratio, delay average minimum difference, packet inter-arrival time, packet loss ratio, average number of lost packets, video packets probability, and audio packets probability at the current time step. The dataset also includes audio quality and video quality on a scale of [0,5] at each time step which can be interpreted as the reward r_t. In this case, the action, a_t, at each time step is the bandwidth generated by the algorithm.

The integration of Conservative Q-Learning (CQL) and Soft Actor-Critic (SAC) algorithms allows us to address two primary dimensions in reinforcement learning—offline batch learning and continuous control.

- Offline Batch Learning: In machine learning, offline learning involves training a model with a specific batch of data and then deploying it for prediction without further interaction or learning from the environment. CQL is well-suited for this purpose. It is a variant of Q-learning known for its stability and reliability while learning from static batch data. It addresses a common pitfall in reinforcement learning, which arises when the policy extrapolates beyond its training experiences. CQL does this by augmenting the Q-value loss function to restrict the action-value function from overestimating the value of less experienced actions.
- Continuous Control: The second dimension deals with continuous action spaces—an environment where possible actions are not discrete but form a continuous range. SAC excels in dealing with such situations. It is an off-policy algorithm that not only maximizes expected return but also encourages exploration. The idea is to derive a policy that seeks to minimize its own uncertainty about the actions it should take. SAC accomplishes this via the principle of maximum entropy, leading to improved exploration abilities and more robust learning.

In summary, combining CQL and SAC is an effective approach to handle offline batch reinforcement learning in a continuous action space. CQL helps prevent overoptimistic value estimates when learning from historical data. SAC, on the other hand, ensures efficient and robust exploration in the high-dimensional, continuous action space. Together, they serve as a powerful tandem for addressing a broad range of complex reinforcement learning tasks.

V. Experiment Setup

VI. Dataset Description and Preprocessing

In our study, we utilize a comprehensive dataset provided by the Microsoft Research team, which comprises trajectories from 18,859 real-world Microsoft Teams audio/video calls and 9,405 emulated calls. These trajectories are gathered from testbed nodes strategically distributed across various global locations, employing diverse Internet Service Providers and connection modalities [2].

Each trajectory records an audio/video call leg through a 150-dimensional observation vector, derived from real-time packet information. This vector captures intricate network statistics over both short and long monitoring intervals, detailing aspects such as receiving rates, packet loss, and queuing delays. Additionally, it includes bandwidth estimates influenced by a range of behavior policies—from traditional Kalman-filtering-based estimators to advanced machine learning algorithms—and objective quality metrics for audio and video, rated on a scale from 0 to 5 [2].

To optimize the validity of our analytical models, we employ the real-world dataset for training purposes, while the emulated dataset, which includes ground truth data on bandwidth capacity, is reserved for testing. This approach ensures that our models are not only trained on a broad spectrum of real-world data but are also rigorously tested against controlled emulated scenarios to assess their robustness and generalizability.

A. Behavior Cloning

During the training, for the first few epochs, update the policy using the following equation to do behavior cloning.

$$L_\pi = \mathbb{E}[\alpha * \log(\pi(a|s)) - P(s,a)] \tag{11}$$

where $P(a|s)$ is the behavior policy's probability of taking action a given state s. The equation essentially try to map the distribution of the policy network to $u(a,s)$, which represent the distribution of state and action in the behavior dataset.

B. Policy Update

We update the policy with the following policy loss function.

$$L_\pi = \mathbb{E}\left[\alpha \cdot \log(\pi(a|s)) - Q(s,a)\right] \tag{12}$$

The policy network is updated to balance exploration and exploitation. It encourages both policy with higher entropy and policy that leads to higher Q value.

We apply the "automatic gradient-based temperature tuning method" [5] to update the temperature α

$$L_\alpha = \mathbb{E}\left[-\alpha \cdot (\log(\pi(a|s)) + H)\right] \tag{13}$$

C. Dual Q-Functions

In line with the standard SAC algorithm, our approach incorporates two Q-functions, denoted as Q_1 and Q_2. We select the minimum of Q_1 and Q_2 to compute the Q-value contributing to the calculation of policy loss, thereby enhancing the stability of the training process.

D. Application of Conservative Q-Learning

Conservative Q-Learning (CQL) plays a significant role in updating the Q function. The core principle of CQL is to impose heavier penalties on overestimation errors compared to underestimation errors [4]. As a result, we update Q_1 and Q_2 by minimizing objective function:

$$\begin{aligned}\min_Q \alpha' \left(\mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}[Q_\theta(s,a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\mathcal{D}}[Q_\theta(s,a)]\right) \\ + \frac{1}{2}\mathbb{E}_{s,a,r,s'\sim\mathcal{D}}\left[\left(Q_\theta(s,a) - \hat{Q}(s,a,r,s')\right)^2\right]\end{aligned} \tag{14}$$

where the first term penalize the overestimation errors and the second term is a bellman update. We set α', the weight of the penalization term, to 1 in our experiment.

Additionally, we have incorporated a penalty mechanism specifically designed to minimize overestimation that significantly exceed the true bandwidth. This strategy aims to counteract potential drawbacks associated with the model's overestimation. Such a penalty becomes necessary as over-predicting bandwidth beyond the actual value may cause severe network congestion, leading to video freeze incidents. This eventuality could adversely affect user experience, underlining the need for mitigating overestimation risks.

VII. Results

We evaluated our data using the average Euclidean distance between the predicted and the true bandwidth values, comparing it with a traditional bandwidth estimator and BC. The normalized Euclidean distances for the RL model and the behavior policy are presented in Table I.

Estimator	Traditional Bandwidth	CQL-BWE Model
Normalized Euclidean Distance	16.5	19.04

TABLE I
COMPARATIVE PERFORMANCE OF DIFFERENT BANDWIDTH ESTIMATORS.

This table reflects the average normalized scores of different estimators evaluated on an offline dataset.

Figures below provide a visual comparison between our CQL-BWE model results, the ground truth, and traditional bandwidth estimators.

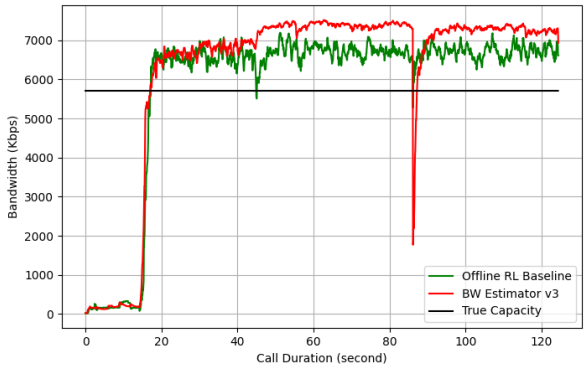

Fig. 1. Comparative results of the CQL-BWE model against traditional estimators and ground truth.

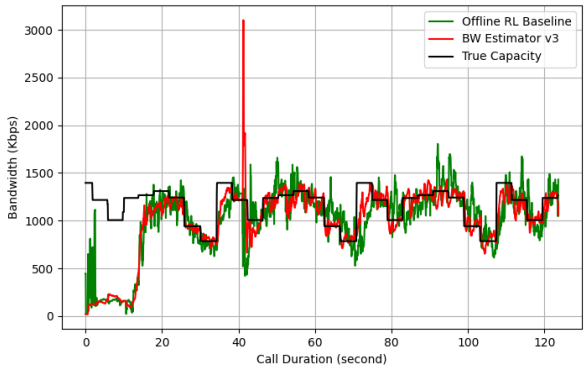

Fig. 2. Comparative results of the CQL-BWE model against traditional estimators and ground truth.

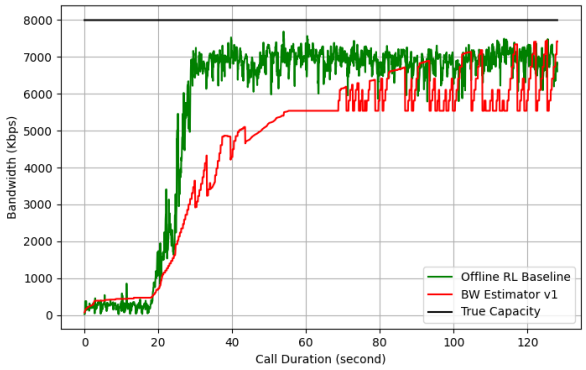

Fig. 3. Comparative results of the CQL-BWE model against traditional estimators and ground truth.

The figures illustrate the performance of CQL-BWE with green lines, demonstrating its capability to closely match the true bandwidth capacities across various test scenarios. Despite the RL algorithm indicating a larger Euclidean distance compared to the behavior policy, these experiments underline the potential of offline RL algorithms to surpass traditional methods, particularly in estimating values closer to the actual bandwidth capacity under certain conditions.

VIII. Future Work

Our research has highlighted the efficacy of offline reinforcement learning (RL) systems. Acknowledging the dynamic nature of real-world environments, we foresee substantial enhancements in model performance through the incorporation of larger, more diverse datasets in future training phases. To optimize adaptability and efficacy, we propose a hybrid approach that leverages the robust framework of our offline RL system augmented by online RL techniques. This strategy aims to utilize the rapid, risk-mitigated learning capabilities of offline RL as a foundation, subsequently refined by the responsive, environment-specific adaptations provided by online RL. Our future initiatives will focus on developing seamless integration methods for these two modalities, aiming to surpass the performance benchmarks of traditional offline systems while also improving efficiency and reducing the resource expenditures typically associated with online RL.

IX. Conclusion

This study underscores the considerable potential of offline RL to enhance system performance through the efficient utilization of existing extensive data repositories, thereby minimizing exploration costs and avoiding the risks associated with trial-and-error methods in unpredictable environments. Nevertheless, the perpetual evolution of real-world conditions necessitates more agile and responsive models. Our proposed solution—a synthesis of offline training with strategic online fine-tuning—marries the efficiency of offline RL with the dynamic adaptability of online RL, offering a promising avenue to achieve optimal balance. Future research will focus on refining this hybrid model, utilizing comprehensive offline datasets for initial training and selectively applying online RL to ensure the model remains robust and responsive to environmental changes.

References

[1] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM, 2017, pp. 197–210.

[2] Microsoft Corp., "Microsoft Network Trace Dataset 2024," 2024, [Online]. Available: https://github.com/microsoft/RL4BandwidthEstimationChallenge.

[3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in Proceedings of the 35th International Conference on Machine Learning, vol. 80, 2018, pp. 1861–1870.

[4] A. Kumar, J. Fu, G. Tucker, and S. Levine, "Conservative Q-Learning for Offline Reinforcement Learning," arXiv preprint arXiv:2006.04779, 2020.

[5] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft Actor-Critic Algorithms and Applications," CoRR, vol. abs/1812.05905, 2018, [Online]. Available: http://arxiv.org/abs/1812.05905.

[6] S. Holmer, H. Lundin, G. Carlucci, L. De Cicco, and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication," Internet-Draft, draft-ietf-rmcat-gcc-02, Internet Engineering Task Force, July 8, 2016, [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rmcat-gcc/02/.

[7] Jeongyoon Eo, Zhixiong Niu, Wenxue Cheng, Francis Y. Yan, Rui Gao, Jorina Kardhashi, Scott Inglis, Michael Revow, Byung-Gon Chun, Peng Cheng, and Yongqiang Xiong. 2023. OpenNetLab: Open Platform for RL-based Congestion Control for Real-Time Communications. In Proceedings of the 6th Asia-Pacific Workshop on Networking (APNet '22). Association for Computing Machinery, New York, NY, USA, 70–75. https://doi.org/10.1145/3542637.3542648

[8] J. Fang, M. Ellis, B. Li, S. Liu, Y. Hosseinkashi, M. Revow, A. Sadovnikov, Z. Liu, P. Cheng, S. Ashok, D. Zhao, R. Cutler, Y. Lu, and J. Gehrke, "Reinforcement learning for bandwidth estimation and congestion control in real-time communications," in CoRR, vol. abs/1912.02222, 2019, [Online]. Available: http://arxiv.org/abs/1912.02222

[9] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," in CoRR, vol. abs/2005.01643, 2020.