A Deep Reinforcement Learning Approach to Balance Viewport Prediction and Video Transmission in 360° Video Streaming

Guanghui Zhang[1] and **Jing Guo**[2]
[1]Shandong University
[2]The Hong Kong Polytechnic University
gh.zhang@sdu.edu.cn, guojing_jane@foxmail.com

Abstract

360° video streaming has seen tremendous growth in past years. However, our measurement reveals a dilemma that severely limits QoE. On the one hand, viewport prediction requires the shortest possible prediction distance for high predicting accuracy; On the other hand, video transmission requires more buffered data to compensate for bandwidth fluctuations otherwise substantial playback rebuffering would be incurred. Since no existing method can break this dilemma, the QoE optimization was naturally bottlenecked. This work tackles this challenge by developing QUTA – a novel learning-based streaming system. Specifically, our measurement shows that three kinds of internal streaming parameters have significant impacts on the prediction distance, namely, download pause, data rate threshold, and playback rate. On top of this, we design a new long-term-planning (LTP) learning method that tunes the parameters dynamically based on the network and streaming context. Evaluations with large-scale streaming trace data show that QUTA not only improves the prediction accuracy and QoE by up to 68.4% but also exhibits strong robustness.

1 Introduction

In recent years, the notion of Metaverse has become very hot where one attractive application is 360° panorama video streaming [Yaqoob et al. 2020]. Unlike the 2D videos without any interactivity, 360° videos allow viewers to change viewports freely during the 3D video playback to obtain an immersive experience. Due to the huge market demand, the major streaming vendors have all launched the 360° video services, e.g., YouTube, Facebook, and Netflix.

Given the 360° panorama scenes, the frame resolution is typically 4K or higher. However, transmitting such high-resolution videos over the Internet is far from trivial, especially under the mobile network with substantial bandwidth fluctuations. Moreover, since the viewer can only watch a portion of the panoramic scene (i.e., within the viewport, c.f. Fig. 1), delivering the entire frame in equally high quality would inevitably result in large wasted bandwidth such that Quality-of-Experience (QoE) is limited. Therefore, in practice, the goal for transmitting 360° videos is to maximize the video quality inside the viewport while reducing the quality outside to reach high transmission efficiency. To this end, on the server side, the video frame is cropped into different spatial tiles, each of which is encoded into multiple bitrate levels (Fig. 1). At runtime, the client periodically predicts viewers' future viewport and downloads the in-viewport tiles at the highest possible bitrate while the outside tiles are at low bitrate, and then pieces them together in the buffer for playback.

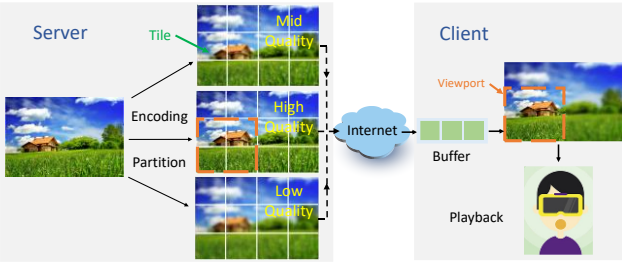

Figure 1: The architecture of 360° video streaming.

Dilemma. Theoretically, the above design can maximize the QoE. However, we measured some state-of-the-art viewport prediction algorithms in real streaming environments and found that the prediction accuracy is far from optimal. The reason is that these algorithms extract the historical viewports from the already played video segments and then make predictions for future pre-download segments (see Fig. 2). In the streaming context, the buffered video lengthens the *prediction distance* and thus weakens the viewport correlation. Consequently, the historical viewport cannot guide the predictions well and the tile bitrate cannot be assigned accurately, leading to severe limitations in video quality and QoE. Guided by this insight, one may turn to reducing the prediction distance to address the problem. This method did improve the accuracy. However, the shorter prediction distance indicates less buffered data, which leads to much rebuffering and large QoE degradations due to the fluctuating network.

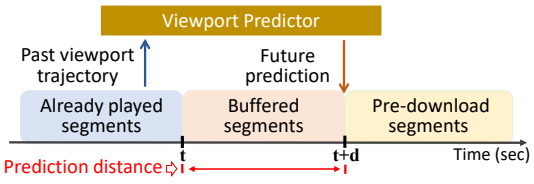

Figure 2: Illustration of viewport prediction.

In recent years, several 360° streaming algorithms were proposed but none of them can break the above dilemma, so the QoE optimization has naturally been bottlenecked (a comprehensive related work review is in §2). This paper tackles this challenge by developing QUTA, a novel learning-based streaming system that is capable of coordinating the viewport prediction and the video transmission well. Evaluations show that QUTA not only improves the viewport prediction and QoE by up to 68.4% but also exhibits strong robustness. In summary, this work makes *four contributions*:

- This is the first work so far to comprehensively investigate the aforementioned dilemma in 360° video streaming. Our measurement demonstrates that this problem is *real* and *pressing*, but *difficult* to solve in the current platform.
- We found that three kinds of internal streaming parameters have large impacts on the prediction distance: 1) download pause, i.e., making the video download periodically pause at the segment interval; 2) data rate threshold, i.e., matching the sum data rate of the video tiles to the throughput dynamics; 3) playback rate control. On top of this, we formulated an optimization problem to maximize QoE by tuning the parameter values. (§3.1)
- In the 360° streaming context, an intelligent model is needed to solve this optimization problem consecutively for all the video segments. Since the three parameters are all in continuous-valued domains, we applied Continuous Control Deep Reinforcement Learning (CC-DRL) [Lillicrap et al. 2015] for the parameter decisions. However, we found that the state-of-the-art CC-DRL method [Fujimoto et al. 2018] is very short-sighted, merely using single-step future planning. At each epoch, it only greedily computes the Q-value of the next (one) segment, which leads to suboptimal training performance in the streaming context with consecutive decision-making for all segments. To this end, we improved the training method and proposed a long-term-planning scheme to predict the Q-value of the next multi-step segments, which greatly improves the training efficacy. We call this new training method as "LTP" in this work. (§3.2)
- This work validates the performance of QUTA via large-scale data-driven emulations. We not only evaluated QUTA's streaming performance and robustness but also looked deeply into the underlying parameter decisions to analyze the root cause of its superiority. (§4)

2 Related Work

In this section, we will review the recent studies relevant to this paper. Interested readers can also refer to a recent survey paper [Yaqoob et al., 2020].

Bitrate Adaptation and Allocation. Under the DASH standard [Stockhammer et al., 2011], bitrate adaptation has been a hot research topic in the field of conventional 2D video streaming. Compared to the 2D streaming merely making bitrate decisions for video segments, the 360° streaming algorithm is more fine-grained, where it first predicts the future viewport motions and then separately determines the bitrate of each video tile based on the prediction results.

[Qian et al., 2018] conducted subjective experiments on real viewers and collected their viewport motion trajectories. Based on the experimental analysis, they developed an online system Flare which includes a viewport prediction algorithm based on real-time viewports and a bitrate allocation strategy built on optimization problem solving; [Guan et al., 2019] discovered several new features that affect the perceptual quality of the 360° videos (e.g., viewport motion speed), and on this basis, they developed a new visual perception model to guide bitrate allocation; [Zhang et al., 2021] found that the size of the video tile has a significant impact on the streaming performance. For example, reducing the tile size decreases the accuracy of viewport prediction, leading to video quality degradation, while upsizing increases the data size to fetch and thus could incur more rebuffering. Driven by this finding, they proposed an adaptive scheme that dynamically adjusts the tile size according to the accuracy of the viewport prediction and then assigns bitrates on the changing tiles.

Another branch of work is built on deep reinforcement learning (DRL). [Zhang et al., 2019] proposed the first DRL-based algorithm that uses LSTM (Long Short-Term Memory) networks to predict future viewport changes, and then allocate bitrate via an Actor-Critic network; [Wu et al., 2021] found that the current DRL-based approaches are typically trained with a solo QoE function, but in practice, different viewers could have different QoE preferences. If the adopted QoE function does not match the viewer's actual preference, then the service quality would be significantly degraded. To this end, they proposed PAAS (Preference Aware Adaptive 360° Video Streaming), which offers a multi-objective DRL setup that uses a preference-aware Deep Q-learning algorithm to implement a training agent under the dynamic QoE preferences and enforce the agents to execute preference-dependent bitrate allocation. [Lu et al., 2022] developed a meta-learning-based framework that uses efficient meta-network designs to capture the commonality among the viewers with different viewing patterns and QoE preferences. [Shi et al., 2022] proposed Sophon, which leverages saliency-aware prefetch and super-resolution model to address the challenge of insufficient computing resources and dynamic user preferences.

Viewport Prediction. Viewport prediction plays a decisive role in the performance of the 360° video streaming system. Therefore, in recent years, many research works have been conducted to improve the prediction accuracy.

The majority of the existing prediction algorithms are driven by viewers' past viewports. For example, [Nasrabadi et al., 2020] proposed a clustering-based method that segregates the collected viewport trajectories into different clusters and then performs feature extraction. At runtime, it will

match each viewer into different clusters and then make predictions according to the cluster features; [Xu et al., 2019] developed a DHP method (DRL-based Head-movement Prediction), which has an offline and an online phase. In the offline DHP, multiple DRL workflows are run simultaneously to learn the features of different viewport trajectories. Then the system outputs viewport hotspot maps which will guide viewport prediction in the online DHP.

In addition to the past viewports, researchers found that the video content also plays a key role in affecting the future viewport motion, so, if the hot video content can be well detected, then the viewport prediction will be greatly facilitated. Based on this motivation, [Nguyen et al., 2018] developed PanoSalNet, a system that trains a convolutional neural network (CNN) to detect hot contents in 360° videos and then quantifies them via the saliency map. In addition, this work also trains an LSTM network to perform viewport prediction based on the saliency map as well as the past viewports; Thereafter, [Xu et al., 2018] and [Rondon et al., 2021] further improved the LSTM network in PanoSalNet and achieved significant accuracy improvements. In another study, [Chopra et al., 2021] found that the viewport usually changes with the major objective movement in the video content (e.g., the athletes in a sports event). Based on this finding, they developed PARIMA which attempts to identify the major objectives in the content and then predicts future viewports by tracking their movement trajectories; [Wang et al., 2022] developed SalientVR that only relies on the saliency maps for the prediction and completely abandons the use of the historical viewport trajectory.

Problem: As presented in §1, all the existing algorithms of 360° video streaming suffer from a dilemma that significantly limits the QoE performance. Specifically, on the one hand, the viewport prediction requires the shortest possible prediction distance for high accuracy, but the buffered video data lengthens the prediction distance and weakens the viewport correlation; On the other hand, the video delivery requires more buffered data to compensate for the bandwidth fluctuations otherwise substantial rebuffering events would be incurred. In §3, we will show how the proposed QUTA solves this problem.

3 Methodology

In this section, we will introduce QUTA. In §3.1, three parameters are proposed to control the prediction distance. In §3.2, a new learning method is proposed for determining the parameter values in the video streaming context.

3.1 Internal Streaming Parameter

How to control the predication distance? As discussed in §1 ~ §2, the prediction distance has a strong positive correlation with the buffering. In video streaming, the buffered data originates from the video download so one can regulate the download speed to control the prediction distance. Furthermore, the video download and the playback are two synchronized processes in streaming, so one can also regulate the playback speed. These insights inspire us to propose the following:

Video download pause. The mechanism is to periodically pause the video download via setting a sleeping time σ_k before requesting each segment k. Meanwhile, the playback is still going on so the buffer level would drop. Thus, controlling the sleep time can control the buffer. Specifically, ignoring the network latency, let t_k and f_k be the starting and completion time for transferring segment k. Let b_k be the buffer occupancy at time f_k. Via tuning σ_k, we can schedule the starting time to transmit the next segment at t_{k+1}:

$$t_{k+1} = \begin{cases} f_k, \text{ if } b_k < \sigma_k \\ f_k + \sigma_k, \text{ otherwise} \end{cases} \quad (1)$$

Discussion: One may wonder whether the frequent download pause will leave a large amount of bandwidth idle and then the bandwidth utilization will be greatly reduced. The answer is *no*, because the bandwidth is typically in a multi-user competition state. Once the base station detects any idle usage, it will immediately allocate the resources to other users to balance the whole service quality [Mao et al. 2017].

Data rate allocation. Theoretically, if the total video data rate (i.e., the sum bitrate of all the tiles) is exactly equal to the throughput during streaming, then the buffer level can be kept in a dynamic equilibrium because the download time equals the playback time. If one further turns up (down) the data rate, then the download time will accordingly increase (decrease) and the buffer level will decrease (increase). Therefore, this inspires us to manage the data rate.

In 360° video streaming, each segment k is cropped into multiple spatial tiles t=0,1,…,T-1, and each tile is encoded into multiple bitrate versions $r_{k,t}$. Before the download, the player makes bitrate decisions for each tile, denoted by function $G(.)$, which assigns the highest possible bitrate to the predicted in-viewport tiles (flag $\xi_{k,t}$=1) and a low bitrate to outside tiles (flag $\xi_{k,t}$=0). Upon this workflow, we defined a new parameter μ_k (called data rate threshold):

$$\{r_{k,t} \mid \forall t\} \leftarrow G(\mu_k, \xi_{k,t}), \text{ s.t. } \sum\nolimits_{\forall t} r_{k,t} \le \mu_k \quad (2)$$

where the sum bitrate of all the tiles in segment k is limited within μ_k. As a result, by tuning μ_k, the system is able to manage the video download time.

Playback rate control. We denote the buffer level before requesting segment k as b_k. After downloading k, the buffer level (denoted by b_{k+1}) will become

$$b_{k+1} = \max\left[\left(b_k - \rho_k \times \tau_k\right), 0\right] + l_k \quad (3)$$

where ρ_k is the playback rate of segment k, τ_k is the time spent on downloading k, l_k is the segment physical duration. Intuitively, turning up ρ_k will result in more buffered data to be consumed during downloading segment k, and conversely, turning down ρ_k will increase the buffer.

Subjective test for playback rate change: Tuning the playback rate for the 360° videos has been widely adopted by many commercial platforms such as YouTube, Netflix, etc. Interested readers can visit [YouTube Instance 2023] to experience an actual instance. In practice, the streaming vendor would like to make the rate changes imperceptible to the

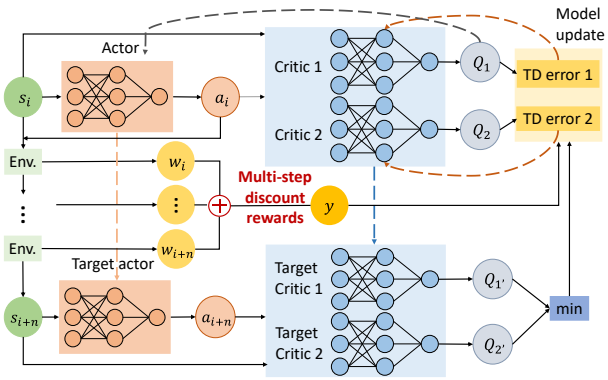

Figure 3: An illustration of the training method: LTP training is based on the decisions for historical segment i, i+1 … i+n. Actor inputs state s_i and outputs action a_i. Target actor inputs state s_{i+n} and outputs action a_{i+n}. Critic 1(2) input s_i and a_i, and output Q-value $Q_{1(2)}$. Target Critic 1(2) input s_{i+n} and a_{i+n}, and output target Q-value $Q_{1'(2')}$. Rewards w_i, w_{i+1} … w_{i+n} are evaluated in the environment, and y is the multiple-step discount reward. Critic 1(2) is updated based on TD errors. Actor is updated based on Q_1. Target networks are updated based on the corresponding learned networks. The training details are described in §3.2.

viewers (the normal rate is ×1.0) to avoid QoE degradation. To this end, we recruited 100+ real viewers to test the effects of the rate change. We selected five 360° videos from YouTube with different genres and then iteratively assigned them to each viewer. From the normal rate ×1.0, we gradually turned up/down the rate each minute (within ×0.5 ~ ×1.5). The viewers were asked for feedback whenever detecting a rate change. Finally, we found that only a rate change larger than ±20% can be detected. Other research work offered similar observations [Nguyen et al. 2022]. Therefore, we defined a maximum change limit $\kappa_{\max}$ on the playback rate deviation:

$$\beta_k = \begin{cases} \min(\rho_k, 1+\kappa_{\max}), & \rho_k > 1 \\ \rho_k, & \rho_k = 1 \\ \max(\rho_k, 1-\kappa_{\max}), & \rho_k < 1 \end{cases} \quad (4)$$

where $\kappa_{\max}$ is set to 20% in this work, i.e., β_k is in×0.8 ~ ×1.2.

3.2 Long-Term-Planning Model Training

The fundamental purpose of this work is to optimize the QoE. To this end, we formulate an optimization problem where the objective function is to maximize QoE w_k via consecutively deciding σ_k, μ_k, ρ_k (see §3.1) for all the video segments k=1,2,…,K in one streaming session:

$$\max_{\{\sigma_1,\sigma_2\ldots\sigma_K\},\{\mu_1,\mu_2\ldots\mu_K\},\{\rho_1,\rho_2\ldots\rho_K\}} \sum_{k=1}^{K} w_k \quad (5)$$

where QoE w_k can be quantified by any QoE function like (13), and the optimization constraints follow (1) ~ (4).

Why CC-DRL? In practice, however, solving the above problem is challenging. First, the computation overhead is not negligible because the decision for σ_k, μ_k and ρ_k is consecutively made on each segment k so the video player needs to solve a complex nonlinear optimization problem at each time step (epoch) online. Second, there is no direct solver for this nonlinear problem so one may need to use an external solver, which leads to deployment issues. In comparison, DRL methods offer an offline-online combinational solving method where most of the complexities can be consolidated in the offline training, leaving a simple model for online decision-making. Moreover, σ_k, μ_k, ρ_k are in continuous (real-valued) action domains, so in QUTA, we applied Continuous Control Deep Reinforcement Learning (CC-DRL) [Lillicrap et al. 2015] to train the parameter decision model.

The challenge to adopt CC-DRL. However, we found that the state-of-the-art CC-DRL, e.g., TD3 [Fujimoto et al. 2018], is very short-sighted, merely using single-step future planning to calculate the Q value of only the next (one) step segment, which makes the training difficult to converge under the consecutive decisions in the streaming context. To this end, we improved this existing method by proposing a long-term planning strategy to predict the Q value of the next multi-step segments. We called this new method as *LTP* which greatly improves the training convergence speed and stability. Its core procedure is illustrated in Fig. 3. The details are in the following.

Input States. On requesting segment k, the neural network captures the environmental features by taking in a list of input states s_k. In this work, we defined two states: *State* 0 is called throughput-vector which consists of g samples, i.e., $\mathbf{C}_k = <c_{k-1}, c_{k-1}, \ldots, c_{k-g}>$, where each sample quantifies the mean throughput in downloading one of the past g segments, i.e., k-g ~ k-1; *State* 1 is called buffer-vector which consists of h samples, i.e., $\mathbf{B}_k = <b_{k-1}, b_{k-2}, \ldots, b_{k-h}>$, where each sample represents the instant buffer level on requesting one of the past h segments, i.e., k-h ~ k-1.

Neural Network. The decision model consists of six neural networks (Fig. 3), namely, Actor π_ϕ with neuron weight ϕ, two Critic $Q_{\theta 1}$ and $Q_{\theta 2}$ with weight $\theta 1$ and $\theta 2$, Target Actor $\pi_{\phi'}$, and two Target Critic $Q_{\theta 1'}$ and $Q_{\theta 2'}$. Note that only Actor π_ϕ performs the parameter decisions and interacts with the streaming environment, and the remaining five are all training assistants. We will detail the training later.

Output Action. Actor π_ϕ has three output ports, i.e., $\{\sigma_k, \mu_k, \rho_k\}$, collectively denoted by a_k. They are all continuous values with unlimited ranges, i.e., $a_k \in \mathrm{R}$. To limit a_k into an available range $[a_{\min}, a_{\max}]$, we defined a linear mapping:

$$\hat{a}_k = \begin{cases} a_{\min}, & a_k < a_{\min} \\ a_{\max}, & a_k > a_{\max} \\ (a_k)_{0.01}, & \text{else} \end{cases} \quad (6)$$

where the output is the final action of segment k, and $(a_k)_{0.01}$ denotes keeping two decimal places to facilitate the practical system implementation.

Environment. In this work, to speed up the training, we leverage virtual streaming [Mao et al. 2017] to emulate the streaming environment where the neural network is executed in a simulated environment replayed by network traces and video traces (c.f. §4.1).

Training. At first, whenever receiving state s_k, Actor π_ϕ will make action a_k for segment k: $a_k \leftarrow \pi_\Phi(s_k)+\delta$ where δ is a random exploration noise, and a_k will be mapped to the available range according to policy (6). The system then applies a_k in streaming segment k via the environment, and reward w_k is calculated according to the resulting streaming performance to express the goodness of the last decision. Since QUTA targets optimizing QoE, the reward w_k can be quantified by any QoE function like (13). After downloading segment k, state s_k is translated to s_{k+1} due to the model's interaction with the environment. The transition tuple (s_k, a_k, w_k, s_{k+1}) will be gathered and stored in a resource pool ψ.

Long-Term Planning: At each epoch, the system samples a mini-batch of M transition queues $\{\Pi_m \mid m=1\ldots M\}$ from ψ where each queue Π_m contains a sequence of n transition tuples sorted by the order of the streaming segments:

$$\Pi_m \leftrightarrow \left\langle \left(s_{m,i}, a_{m,i}, w_{m,i}, s_{m,i+1}\right), \ldots, \left(s_{m,i+n}, a_{m,i+n}, w_{m,i+n}, s_{m,i+n+1}\right) \right\rangle \tag{7}$$

where the segment index is from i to $i+n$ and the initial one is randomly chosen. The two Critics $Q_{\theta 1}$ and $Q_{\theta 2}$ will be updated separately based on the M transition queues by tuning their neural weights $\theta_{\varepsilon=1,2}$ to achieve a minimum TD (temporal difference) error [Sutton et al. 1988]:

$$\theta_{\varepsilon=1,2} = arg\ \min_{\theta_\varepsilon} M^{-1} \sum_{\forall m} \left(P_m - Q_{\theta_\varepsilon}\left(s_{m,i}, a_{m,i}\right)\right)^2 \tag{8}$$

where P_m is an expectation for the Q value which is calculated by target reward y_m and target Q value (output by two Target Critics $Q_{\theta 1'}$ and $Q_{\theta 2'}$ separately):

$$P_m = y_m + \gamma \times \min\left\{Q_{\theta 1'}\left(s_{m,i+n}, a_{m,i+n}\right), Q_{\theta 2'}\left(s_{m,i+n}, a_{m,i+n}\right)\right\} \tag{9}$$

where $a_{m,i+n}$ is the target action for the last segment $i+n$ output by Target Actor $\pi_{\phi'}$, γ is the discount factor, and the target reward y_m is a discount combination of the rewards in all the transitions from (7):

$$y_m = \gamma^0 w_{m,i} + \gamma^1 w_{m,i+1} + \ \ldots\ + \gamma^{n-1} w_{m,i+n} \tag{10}$$

It is worth noting that the Q value estimation is the most remarkable improvement of the LTP training beyond the existing methods, which is a long-term planning strategy to predict the target Q of the next *multiple-step* segments instead of the short-sighted single step [Fujimoto et al. 2018]. This greatly improves the training convergence speed and stability.

Actor π_ϕ is updated at a lower frequency than Critics, which is at a fixed step number of d (e.g., =2). Each update follows the deterministic policy gradient:

$$\nabla_\phi J(\phi) = M^{-1} \sum_{\forall m} \nabla_{a_{m,i}} Q_{\theta 1}\left(s_{m,i}, a_{m,i}\right) \nabla_\phi \pi_\phi\left(s_{m,i}\right) \tag{11}$$

The rule behind this update is to find the optimal policy π_ϕ by tuning the neural weight ϕ to maximize the expected return $J(\phi)$, and Eq. (11) takes only one step toward this direction. The neuron weights of all the target networks (i.e., Target Critic $\theta'_{\varepsilon=1,2}$, and Target Actor ϕ') are then updated via EWMA policy (exponential weighted moving average). This makes the target neuron weights slowly track the newly learned networks (i.e., Critic $\theta_{\varepsilon=1,2}$, and Actor ϕ) so that the TD error remains small to keep the training steady.

Features	#1	#2	#3
Mean throughput (Mbps)	3.97	6.43	10.4
Throughput variation	0.74	0.58	0.42
Network condition	Poor	Medium	Good

Table 1: Characters of Network Traces

4 Performance Evaluation

In this section, we conducted data-driven emulations to compare QUTA to the state-of-the-art algorithms to see the performance improvements.

4.1 Experimental Setup

We modified an existing emulator [Emulator, 2021] developed by [Rondon et al. 2021]. The setup is the following:

Video Trace. We used the video trace offered by the emulator which includes 4,275 viewport trajectories captured from 75 videos (each video was watched by 57 viewers). The viewport position is quantified as 3-dimensional spherical coordinates. We split each video into a sequence of 1s-long segments where each frame is cropped into 10×10 tiles. The encoding bitrate follows the setting of [YouTube 2023].

Network Trace. The network condition is emulated by TCP throughput traces captured from real networks. We employed three trace sources #1 ~ #3, including 60,000+ streaming sessions [TCP Trace 2023]. Based on the mean throughput and variations (c.f. Table 1), we marked them as "poor", "medium", and "good" networks respectively.

Viewport Prediction Baseline. Three cutting-edge viewport prediction algorithms are evaluated: 1) Flare [Qian et al. 2018] is only based on historical viewport; 2) SalientVR [Wang et al. 2022] only takes in saliency maps extracted from video content; 3) TRACK [Rondon et al. 2021] incorporates both historical viewport and video content. Upon the viewport prediction, we adopted a common pyramid-based bitrate allocation scheme [Chopra et al. 2021].

Evaluation Metric. 1) Prediction accuracy:

$$\sigma = \sum_{\forall i} \theta_i / K \tag{12}$$

where θ_i is a 0/1 flag that indicates whether the prediction for video tile i is correct (=1) or not (=0), and K is the total tile number in each frame;

2) QoE performance [Wu et al. 2021]:

$$w_k = w_{k,0} - 0.5 \times w_{k,1} - 0.5 \times w_{k,2} - 5 \times w_{k,3} \tag{13}$$

where k is the segment index, $w_{k,0}$~ $w_{k,3}$ are four QoE metrics:

$w_{k,0}$ – average tile bitrate (quality) within the viewport:

$$w_{k,0} = \sum_{\forall i} \left(r_{k,i} \times \xi_{k,i}\right) / \sum_{\forall i} \xi_{k,i} \tag{14}$$

where $r_{k,i}$ is the bitrate of tile i, and $\xi_{k,i}$ is a 0/1 flag indicating whether tile i is actually viewed (=1) or not (=0);

$w_{k,1}$ – spatial quality variation (over the in-viewport tiles):

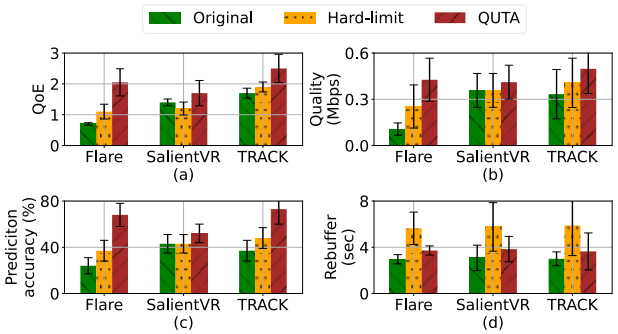

Figure 4: Comparison of QoE (↑), viewing video quality (↑), prediction accuracy (↑), and rebuffering (↓).

$$w_{k,1} = \sum_{\forall i} \left(\left| r_{k,i} - w_{k,0} \right| \times \xi_{k,i} \right) \Big/ \sum_{\forall i} \xi_{k,i} \tag{15}$$

$w_{k,2}$ – temporal quality variations (to the last segment):

$$w_{k,2} = \left| w_{k,0} - w_{k-1,0} \right| \tag{16}$$

$w_{k,3}$ – rebuffering duration:

$$w_{k,3} = \max\left[\left(\rho_k \times \tau_k - b_k \right), 0 \right] \tag{17}$$

where τ_k is the download time of segment k, ρ_k is the playback rate, and b_k is the buffer level on requesting segment k.

QUTA Settings. The neural network model of QUTA applies sliding windows to receive input states (§3.2). We defined the throughput-vector with g=8 window size which is passed to a CNN with 128 filters. Similarly, the buffer-vector has a h=8 window size which is passed to a CNN with 128 filters. Then the outputs of the CNNs will be aggregated and passed to two hidden layers, each with 128 neurons. We used the optimizer Adam with a learning rate of 10^{-4} for all the networks. The tuning range of the three internal streaming parameters, i.e., sleeping time σ, data rate threshold μ, and playback rate ρ, are 0.1s~3s, 0.2Mbps~12Mbps, and 0.8~1.2, respectively. More discussions about the neural settings and the ablation experiments are shown in §4.4. In QUTA, 60% of the trace data are for training and 40% are for testing.

4.2 Optimizing Viewport Prediction

In this section, we applied QUTA to optimize the viewport prediction algorithms, Flare, SalientVR, and TRACK. We compared QUTA with two variant schemes: 1) The *original* version that eliminates the buffer control; 2) *Hard-limit* that places a fixed-duration buffer capacity (set it to 3s according to [Qian et al. 2018]) to limit the prediction distance. In Fig. 4, we compare QoE (4-a), viewing video quality (4-b), prediction accuracy (4-c), and rebuffering (4-d).

Flare and TRACK have similar performance patterns. Specifically, the hard-limit scheme can effectively improve their QoE over the original version (4-a). This is because the prediction distance is limited within 3s which improves the prediction accuracy (4-c) and viewing quality (4-b). However, this fixed capacity cannot be dynamically adjusted for different network conditions and thus leads to far more rebuffering events (4-d). In comparison, QUTA achieves the best QoE over the three schemes. This benefits from its ability to make long-term parameter decisions based on the dynamic streaming context (c.f. §3.2).

Networks	Original	Hard-limit	QUTA	Improve (%)
Poor	0.13±0.03	0.15±0.03	0.20±0.04	53.8 / 33.3
Medium	1.52±0.31	2.04±0.35	2.56±0.40	68.4 / 25.8
Good	2.41±0.43	3.09±0.51	4.05±0.77	68.0 / 31.1

Table 2: QoE and Improvement over Different Networks

The performance of SalientVR is different from Flare and TRACK. Its QoE under the original scheme is even better than that under the hard-limit (4-a). This is reasonable because SalientVR relies solely on the video content to make the predictions so its accuracy cannot be improved even if the prediction distance is limited (4-c). At the same time, setting a hard limit to the buffer incurs more rebuffering (4-d), leading to QoE degradations. In comparison, QUTA can still improve its QoE, which is due to the ability to dynamically adjust the data rate threshold based on the network conditions (c.f. §3.1).

We further evaluated the dynamics of QoE across three throughput trace sources which are denoted by as "good", "medium", and "poor" networks based on the average throughput and variations (c.f. Table 1). The performance is summarized in Table 2. It is clear that QUTA achieves consistently better QoE than the other two over all the three networks. The QoE improvement is up to 25.8%~68.4%. This demonstrates that QUTA is highly robust. As long as enough network throughput data is provided in the training to allow sufficient learning, QUTA is able to operate stably in the environment with large feature differences.

4.3 Comparison to Streaming Algorithms

We further compared QUTA to three state-of-the-art 360° video streaming algorithms, namely,

- DLR360, based on Policy Gradient [Zhang et al. 2019];
- PAAS, based on Deep Q Network [Wu et al. 2021];
- Meta, based on Meta-Learning [Lu et al. 2022].

In Fig. 5-left, we compare the normalized QoE (N-QoE) of the four algorithms. As can be seen, QUTA significantly outperforms the other three with an average QoE improvement from 20.5% to 29.9%. By contrast, Meta performs the worst where the QoE is only 77% of QUTA. To quantify the main contribution from the QoE metrics, we further evaluated the viewing video quality and rebuffering in Fig. 5-right. We observed that QUTA achieves the highest average video quality (=PAAS) and the least amount of rebuffering.

To further analyze the QoE in different network conditions, we evaluated the algorithms on the three trace sources, i.e., poor, medium, and good, and drew the CDF distribution for per-session QoE in Fig. 6. We observed that DRL360 performs worst in the poor network, indicating that it is very sensitive to the throughput fluctuations. Meta achieves the lowest QoE in the medium and good networks because its model is unable to fully utilize the high-level bandwidth. In comparison, QUTA performs best in all the three sources.

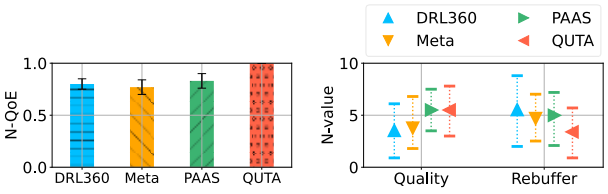

Figure 5: Comparison of QoE (↑), viewing video quality (↑), and rebuffering (↓) over streaming algorithms.

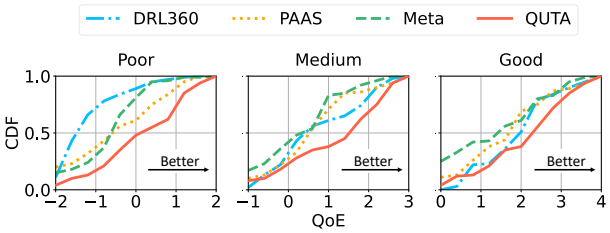

Figure 6: The CDF distribution of QoE over different network conditions (poor, medium, and good).

The superior performance of QUTA can be summarized in threefold: 1) All the design factors of QUTA focus on coordinating the viewport prediction and video delivery so that it manages to break the QoE optimization dilemma in 360° video streaming; 2) QUTA is trained by LTP where the trained model is able to perform long-term planning decisions to pursue the global optimum; 3) Unlike other algorithms, QUTA is a generalized framework that can incorporate any existing best-performing logic (e.g., viewport prediction, bitrate adaptation, etc.), which liberates the maximum performance potential.

4.4 LTP Training and Parameter Decisions

The offline training taken by LTP plays a decisive role in QUTA's superior performance. To evaluate its efficiency, we compared the training QoE to two existing CC-DRL methods: TD3 [Fujimoto et al 2018] and DDPG [Lillicrap et al. 2015] in Fig. 7-left and -middle, and to two discrete-action DRL methods: A3C [Mnih et al. 2016] and DQN [Mnih et al. 2015] in Fig. 7-right. First, compared to TD3 and DDPG, LTP achieves a more stable training QoE, which is due to its future long-term Q-value estimation to pursue a global optimum at each time step (epoch). By contrast, both TD3 and DDPG use short-sighted single-step planning, leading to the QoE having noticeably sharp fluctuations and hard to converge especially under the poor network condition (see Fig. 7-left).

Second, since A3C and DQN are both in discrete action domains, we tuned the number of action ports to see the effect. As shown in Fig. 7-right, while they can keep a relatively high QoE with a small number of action ports, their performance drops dramatically as the port number increases. This is due to the fine-grained discretization for the action domain that makes the learning agent difficult to explore the action space efficiently. On the whole, their performance falls short of LTP in all the cases, even in the best-performing case with a small port number, because the coarse-grained action discretization discards a lot of important decision information.

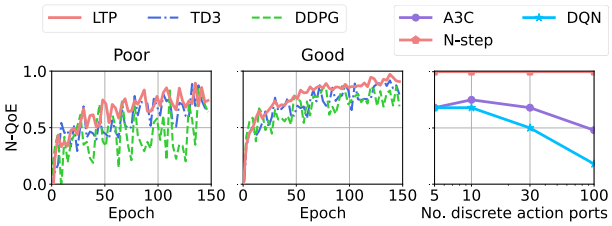

Figure 7: The training QoE over different DRL methods.

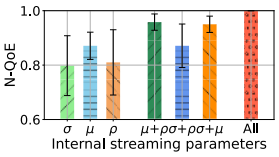

Figure 8: Ablation study (σ–sleeping time, μ–data rate threshold, ρ–playback rate).

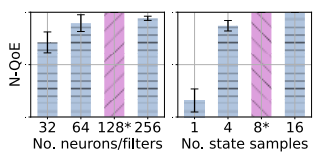

Figure 9: Sensitivity analysis on neural settings (* denotes the default setting).

Next, we conducted ablation studies to investigate the individual contributions from tuning the three parameters, namely, σ – sleeping time, μ – data rate threshold, ρ – playback rate. Specifically, we investigate the significance of: (a) tuning one only while freezing the other two; (b) simultaneously tuning two and freezing the rest one, e.g., tuning both μ and ρ is denoted by "μ+ρ"; (c) tuning all the three, i.e., the full version of QUTA, denoted by "All". The available tuning range of the three parameters is shown in §4.1. Fig. 8 compares the normalized QoE which offers two observations. First, enabling more parameters gives better performance. This demonstrates that each of them offers a unique contribution; Second, in all the cases where the μ is tuned, the QoE performance is typically better than the counterparts, which suggests that the data rate allocation has higher efficacy than the other two parameters.

Starting with the default setting, we further varied the neural settings to understand their impact on the system. The first is the number of neurons/filters in FCN/ CNN. As shown in Fig. 9-left, the QoE keeps increasing from 32 to 128 (the default) and then starts to drop. Next, we tuned the sample number of the input states (see §3.2). Fig. 9-right shows that the neural network with the default state number, i.e., 8, yields the best result while a smaller or larger value would drop the QoE performance.

5 Conclusion

This study reveals a severe dilemma in the existing 360° video streaming system, which bottlenecks the QoE optimization significantly. Motivated by this challenge, we proposed QUTA, a novel learning-based streaming system that is capable of coordinating the viewport prediction and the video transmission well. As a result, QUTA not only achieves substantial QoE improvement but also has strong robustness over the various network environments. Therefore, it fills an important gap in the 360° video streaming and opens a new paradigm for future high-quality services.

Acknowledgments

The authors wish to thank the program chairs and the anonymous reviewers for their insightful comments in improving this paper. This work is supported by the National Natural Science Foundation of China (No. 62302268), the Natural Science Foundation of Shandong Province (No. 2023HWYQ-045, No. ZR2023QF060), the Natural Science Foundation of Qingdao (No. 23-2-1-127-zyyd-jch), and the Taishan Scholar Project of Shandong Province (No. tsqn202312051).

References

[Yaqoob et al., 2020] A. Yaqoob, T. Bi and G. Muntean, "A Survey on Adaptive 360° Video Streaming: Solutions, Challenges and Opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22(4), Fourth Quarter 2020, pp. 2801-2838.

[Stockhammer et al., 2011] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP: Standards and Design Principles," *Proc. ACM Multimedia Systems Conference (MMSys'11)*, San Jose, USA, Feb 2011, pp.133-144.

[Qian et al., 2018] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical Viewport-adaptive 360-degree Video Streaming for Mobile Devices," *Proc. Annual International Conference on Mobile Computing and Networking (MobiCom'18)*, New Delhi, India, Oct. 2018, pp. 99-114.

[Guan et al., 2019] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang. "Pano: Optimizing 360° Video Streaming with A Better Understanding of Quality Perception," *Proc. ACM Special Interest Group on Data Communication (SIGCOMM'19)*, Beijing, China, Aug 2019, pp. 394-407.

[Zhang et al., 2021] L. Zhang, Y. Suo, X. Wu, F. Wang, Y. Chen, L. Cui, J. Liu, and Z. Ming, "TBRA: Tiling and Bitrate Adaptation for Mobile 360-Degree Video Streaming," *Proc. ACM International Conference on Multimedia (MM'21)*, Chengdu, China, Oct. 2021, pp. 4007–4015.

[Zhang et al., 2019] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li. "DRL360: 360-degree Video Streaming with Deep Reinforcement Learning," *Proc. IEEE Conference on Computer Communications (INFOCOM 2019)*, Paris, France, April 2019, pp. 1252-1260.

[Wu et al., 2021] C. Wu, Z. Wang, and L. Sun, "PAAS: A Preference-aware Deep Reinforcement Learning Approach for 360° Video Streaming," *Proc. ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'21)*, Oct 2021, Istanbul, Turkey, pp. 34-41.

[Nasrabadi et al., 2020] A. Nasrabadi, A. Samiei, and R. Prakash, "Viewport Prediction for 360° Videos: A Clustering Approach," *Proc. ACM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'20)*, Istanbul, Turkey, June 2020, pp. 34-39.

[Xu et al., 2019] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang, "Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41(11), Nov. 2019, pp. 2693-2708.

[Nguyen et al., 2018] A. Nguyen, Z. Yan, and K. Nahrstedt. "Your attention is Unique: Detecting 360-degree Video Saliency in Head-mounted Display for Head Movement Prediction," *Proc. ACM International Conference on Multimedia (MM'18)*, Seoul, South Korea, Oct. 2018, pp. 1190-1198.

[Xu et al., 2018] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao, "Gaze Prediction in Dynamic 360 Immersive Videos," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2018)*, Salt Lake City, UT, USA, June 2018, pp. 5333-5342.

[Rondon et al., 2021] M. F. R. Rondon, L. Sassatelli, R. Aparicio-Pardo and F. Precioso, "TRACK: A New Method from a Re-examination of Deep Architectures for Head Motion Prediction in 360 Videos," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44(9), April 2021, pp. 5681-5699.

[Chopra et al., 2021] L. Chopra, S. Chakraborty, A. Mondal, and S. Chakraborty, "PARIMA: Viewport Adaptive 360-Degree Video Streaming," *Proc. The Web Conference (WWW'21)*, Virtual, April 2021, pp. 2379-2391.

[Wang et al., 2022] S. Wang, S. Yang, H. Li, X. Zhang, C. Zhou, C. Xu, F. Qian, N Wang, and, Z. Xu, "SalientVR: Saliency-Driven Mobile 360-Degree Video Streaming with Gaze Information," *Proc. Annual International Conference on Mobile Computing and Networking (MobiCom'22)*, Sydney, NSW, Australia, Oct. 2022, pp. 1-14.

[YouTube Instance, 2023] An Actual 360° Video for Experiencing the Playback Rate Change on YouTube: https://www.youtube.com/watch?v=hNAbQYU0wpg

[Mnih et al., 2016] V. Mnih, A. P. Badia, M. Mirza, A. Graves, "Asynchronous Methods for Deep Reinforcement Learning," *International Conference on Machine Learning (ICML 2016)*, New York City, NY, USA, Jun 2016, pp. 1928-1937.

[Mao et al., 2017] H. Mao, R. Netravali, M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," *Proc. ACM Special Interest Group on Data Communication (SIGCOMM'17)*, Los Angeles, CA, USA, Aug 2017, pp. 197-210.

[Mnih et al., 2015] V. Mnih, K. Kavukcuoglu, D. Silver, et al. "Human-level Control through Deep Reinforcement Learning," *Nature*, Feb 2015, vol. 518(7540), pp. 529-533.

[Lillicrap et al., 2015] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control with Deep Reinforcement Learning," *arXiv Preprint*, Sep 2015.

[Sutton et al., 1988] RS. Sutton, “Learning to Predict by the Methods of Temporal Differences, *Machine Learning*, 1988, pp. 9-44.

[Lu et al., 2022] Y. Lu, Y. Zhu, Z. Wang, “Personalized 360-Degree Video Streaming: A Meta-Learning Approach,” *Proc. ACM International Conference on Multimedia (MM'22)*, Lisboa, Portugal, Oct. 2022, pp. 3143-3151.

[Shi et al., 2022] J. Shi, L. Pu, X. Yuan, Q. Gong, J. Xu, “Sophon: Super-Resolution Enhanced 360° Video Streaming with Visual Saliency-aware Prefetch,” *Proc. ACM International Conference on Multimedia (MM'22)*, Lisboa, Portugal, Oct. 2022, pp. 3124-3133.

[Emulator, 2021] Trace Data Driven Emulator: http://gitlab.com/miguelfromeror/head-motion-prediction

[YouTube, 2023] Encoding Recommendations: https://support.google.com/youtube/answer/2853702

[TCP Trace 2023] TCP Throughput Trace Data: https://github.com/Streaming-code/TraceData/releases/tag/TraceData

[Nguyen et al., 2022] D. Nguyen, L. Ngan, L. Thuong and T. Huong, "LL-VAS: Adaptation Method for Low-Latency 360-degree Video Streaming over Mobile Networks," *Proc. IEEE Symposium on Computers and Communications (ISCC)*, Rhodes, Greece, Oct. 2022, pp. 1-6.

[Zhang et al., 2019] G. Zhang, and J. Lee, "LAPAS: Latency-Aware Playback-Adaptive Streaming," *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakesh, Morocco, Apr. 2019, pp.1-6.

[Zhang et al., 2021] B. Zhang, T. Teixeira, and Y. Reznik, “Performance of Low-Latency HTTP-based Streaming Players,” *Proc. ACM Multimedia Systems Conference (MMSys '21)*, Istanbul, Turkey, Sep 2021, pp. 356–362.

[Dummynet, 2023] An Improved Version of Dummynet: https://github.com/mclab-cuhk/netmap-ipfw

[Fujimoto et al., 2018] S. Fujimoto, H. Hoof, D. Meger, “Addressing Function Approximation Error in Actor-Critic Methods,” *Proc. International Conference on Machine Learning*, Stockholm, Sweden, July 2018, pp. 1587-1596.