Adaptive Streaming of 360-Degree Videos with Reinforcement Learning

Sohee Park[†], Minh Hoai[†], Arani Bhattacharya[§], Samir R. Das[†]
[†]Stony Brook University, Stony Brook, NY, USA, [§]IIIT-Delhi, New Delhi, India
{soheekim, minhhoai}@cs.stonybrook.edu, arani@iiitd.ac.in, samir@cs.stonybrook.edu

Abstract

For bandwidth-efficient streaming of 360-degree videos, the streaming technique must adapt both to the changing viewport of the user and variations of the available network bandwidth. The state-of-the-art streaming techniques for this problem attempt to solve an optimization using simplified rules that do not adapt very well to the uncertainties related to the viewport or network. We adopt a 3D-Convolutional Neural Networks (3DCNN) model to extract spatio-temporal features of videos and predict the viewport. Given the sequential decision-making nature of such streaming technique, we then apply a Reinforcement Learning (RL) based adaptive streaming approach. We address the challenges of using RL in this scenario, such as large action space and delayed reward evaluation. Comprehensive evaluations with real network traces show that the proposed method outperforms three tile-based streaming techniques for 360-degree videos. Compared to the tile-based streaming techniques, the average user-perceived bitrate of the proposed method is 1.3–1.7 times higher and the average quality of experience of the proposed method is also 1.6–3.4 times higher. Subjective user studies further confirm the superiority of the proposed approach.

1. Introduction

Recently, there has been a significant interest in streaming immersive multimedia content, such as 360-degree videos, over the Internet. Their popularity on streaming platforms like YouTube or Facebook is on the rise. A major challenge of streaming such videos is the amount of video data to be downloaded. Due to the panoramic nature of 360-degree videos, video data could be an order of magnitude larger than conventional videos to achieve similar video bit rates [4]. But the viewer only views a small portion (viewport) of the downloaded 360-degree scene. So, much of the network bandwidth is used up by content that is not actually viewed. This amounts to much poorer Quality of Experience (QoE) relative to conventional videos when there is a network bandwidth constraint. This issue is compounded when the available network bandwidth varies over time in an unpredictable fashion as is common over the Internet [27].

Adaptive video streaming techniques [32, 40, 19] have addressed the latter problem by first encoding the video into multiple chunks of a fixed duration at multiple rates on the server. Then it delivers the video chunks at a rate that best matches the conditions (for example, player buffer level, available bandwidth at that time). The rates here reflect different compression levels and thus different viewing qualities. These techniques basically solve an optimization problem that attempts to maximize the overall QoE (e.g., better video quality and less stalls) under the bandwidth constraint. Since the decision on the video quality must be made in advance of the actual download, the network bandwidth needs to be estimated. Attempting to download a higher quality than the network can support may lead to stalls. Such adaptive streaming is now widely adopted for streaming regular videos over the Internet [32].

Recent work has extended this for 360-degree videos by combining with viewport prediction [7, 10, 28, 38, 26, 42, 15], where an independent technique predicts the user's viewport in advance by observing the viewer behavior (head tracking) and prior static analysis of the video. This prediction is used to download a part of the 360-degree scene, putting more emphasis on the part the user is likely to view. This is achieved by dividing the scene into *tiles* and then choosing what tiles to download and in what quality for each video segment.

Overall, such viewport-adaptive tiled streaming for 360-degree content must run a complex optimization in real time in presence of multiple uncertainties. It strives to improve video QoE that depends on multiple parameters such as video quality (or encoding bit rate) and stall behavior. It must make decisions what to download and in what encoding quality, under uncertainties such as network bandwidth and user viewport. Decisions made at any instant impact the state of the downloading client and also future QoE. Choosing what and when to download is a sequential decision-making process, and it is very much amenable to Reinforcement Learning (RL) [34]. Exploiting RL for adaptive streaming of 360-degree videos is the focus of this

work. Our work is inspired by recent RL-based methods for conventional video streaming [23, 31] and multi-camera image acquisition [35].

However, applying RL to 360-degree tiled video streaming is not straightforward. First, the state and action spaces are large. For example, if we use typical values such as 24 tiles and 6 video encoding quality levels, the search space becomes $(6+1)^{24}$, while there are only 6 choices for a regular video. To address this problem we take the approach of deciding on and downloading one tile at a time instead of trying to make a decision for all tiles at once. This has the added benefit of a more frequent sampling of available network bandwidth, leading to a better bandwidth estimate and shorter adaptation cycle. But there is another challenge: the reward signal for the RL agent is not available immediately after the agent takes an action. This is because the reward is defined based on the QoE, which cannot be evaluated until all the necessary tiles for playing a video segment have been downloaded.

Our contribution in this work is developing an RL formulation that addresses the above challenges. We learn a streaming policy that decides sequentially the tile and tile quality to download and the policy can be trained with delayed rewards. Our evaluations show that the use of RL in this fashion provides a much more agile streaming technique that adapts well to the changing network and user viewport. This provides a significant step-up from the state-of-the-art in 360-degree video streaming literature [10, 16, 28, 38, 26], where simplified/fixed rules are used without any learning component. In the work closest to ours, DRL360 [42], RL is indeed used but this study does not address the large action space challenge. Instead, it settles for downloading all tiles outside the predicted viewport in the lowest video quality, while downloading predicted viewport in just one encoding quality matching the available network capacity. When the viewport prediction is inaccurate, this leads to a high quality variance with the viewport.

Overall, we explicitly consider the need for streaming different video tiles with different bitrates and for redownloading tiles at a higher encoding quality if the network condition allows. We propose a formulation with a manageable action space and a short adaptation cycle that works with delayed reward signals. In a comprehensive evaluation with other techniques, we show that our method quantitatively outperforms the state-of-the-art methods. We also perform user studies to evaluate our methods qualitatively and tie quantitative measures to users' perception.

2. Adaptive Streaming and Related Work

Adaptive Streaming. For adaptive streaming, a 360-degree video is divided across both time and space. Across time, the video is split into multiple chunks of fixed duration, called *segments*, similar to conventional video streaming. Across space, each segment is split into multiple *tiles*. This is done after a projection from the 3D sphere to a 2D plane, e.g., equirectangular projection [25, 17]. Thus, the <segment, tile> tuple is the unit of encoding, storage, and network communication. Each <segment, tile> tuple is encoded in multiple quality levels (i.e., encoding bitrates) at the video server.

In general, tiles are smaller than the viewport (the field of view of the user), so multiple tiles are needed to cover the viewport. The video content needs to be downloaded in advance of playback. Thus, at the time of download, the viewport of the segment being downloaded is unknown and must be predicted. The prediction is modeled as a probability distribution over all possible tiles [10, 26]. Given this prediction as an input, our task is to *select the tiles along with their bit rates for the next segment to be fetched subject to the network constraint.* The network capacity here is the same as available bandwidth. This can vary over time and but can be measured (sampled) by noting the download speed. Overestimating the network capacity may lead to stalls as the segment may not yet to be completely downloaded at the time of playback. Underestimating the capacity, in turn, will have a smooth playback, but at a lower video quality than possible. Inaccurate viewport prediction also leads to similar issues—tiles may be missing at playback or have poor video quality. Overall, this is fundamentally an optimization problem: maximizing the user's quality of experience aggregated over time, subject to network capacity constraints. This optimization is to be addressed using an appropriate adaptive bit rate (ABR) algorithm.

Related Work. ML techniques have been used to improve video streaming but most prior methods are based on supervised or unsupervised learning [40, 33, 6, 38, 15, 26]. In this work, we propose to use RL instead, which is more appropriate for adaptive stream given the need for optimal sequential decisions [34]. A number of recent studies use RL that uses different features and learn the best strategy to fetch video tiles. Q-based RL to learn the fetching strategy has been proposed in [9]. However, this works well only if the network follows a Markovian property. To resolve this, Pensieve [23] and D-Dash [13] propose using RL to learn an effective streaming strategy. However, these approaches do not scale well for tile-based 360 video streaming. The work closest to our work is DRL360 [42]. However, unlike our work, DRL360 does not support different bitrates for predicted viewport tiles. Rest of the tiles are assigned with the lowest encoding quality level.

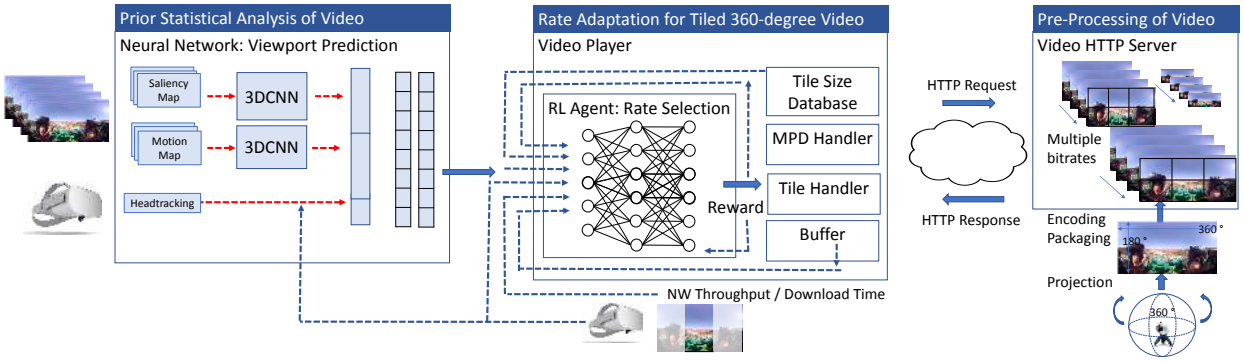


Figure 1: Overview of our system design. *RL Agent* runs at the *Video Player*. It uses the output of *Viewport Prediction* module and other status (Buffer, Network) to choose tile rates. Then the player downloads tiles from the *Video Server*.

Other studies has also used RL to improve perceptual QoE of video streaming. HotDASH uses RL to improve the QoE by prefetching temporally high priority segment for conventional video streaming [31]. Another study called Qarc uses deep neural networks to improve the perceptual quality of experience of streaming videos [18]. These studies are orthogonal to our technique, and can be combined with our technique to further improve the quality of experience of end users.

Using existing adaptive video streaming ecosystem, one tile is downloaded at a time for 360-degree tiled video streaming. However, the existing approaches [42, 28, 26, 10, 14, 37] select the video quality levels of all the tiles of the same segment all at once, based on bandwidth estimation made at the time of this rate selection. When the tiles are actually downloaded, this estimation can change yet the techniques do not allow for any further adaptation.

More recent works [36, 12, 21, 30] uses RL for video streaming but limited in adaptiveness. For example, [30] uses RL to direct viewers to points of interest which are predefined by the content producer instead of streaming based on true user's viewpoint. [21] also uses RL to select a rate for a predicted FoV but it groups tiles into a fixed number of regions (I FoVs) and RL agents select rates for those groups. Our results already show that *ATRIA* outperforms a technique that groups tiles into 4 levels. [12] has sequential decision making for tiles, similar to *ATRIA*, but uses lower performance RL technique [36].

3. Proposed Approach

We use deep RL to learn a streaming policy that can adapt to the predicted behavior of a viewer and the dynamics of the network conditions. We name the resulting streaming policy *ATRIA* (**A**daptive s**T**reaming using **R**e**I**nforcement le**A**rning). Figure 1 illustrates the pipeline of our proposed system. *ATRIA* assumes that a video for streaming has been divided into smaller spatiotemporal subvolumes (Section 2) and hosted at the video server. The core of *ATRIA* is an artificial agent (RL agent in Figure 1) that determines the downloading order and the downloading bitrate for each subvolume. Using viewport prediction, network conditions and other status, the agent follows sequential decision process to maximize the sum of rewards, where the rewards are defined on the quality of viewing experience. In the following, we describe the main components of our RL formulation, including the state representation, the reward function, and action space.

3.1. State Representation

The state is designed to represent various characteristics of video segments, network conditions, and expected viewer's behavior. The state s_t at time t contains:

- $[\tau_{t-1}, \tau_{t-2}, \ldots, \tau_{t-n}]$: downlink throughputs measured for the previous tile downloads, where n is the number of past downloads considered.
- $[\delta_{t-1}, \delta_{t-2}, \ldots, \delta_{t-n}]$: download times taken for the previous n downloads.
- $[s_1, s_2, \ldots, s_K]$: sizes of tiles in K different qualities (encoding bitrates).
- $[p_1, p_2, \ldots, p_N]$: the probabilities that the tiles will be viewed, where N is number of tiles.
- $[q_1, q_2, \ldots, q_N]$: q_i is the quality level selected for i^{th} tile of segment being downloaded. $q_i = 0$ if the tile has not been downloaded yet.
- $[b_1, b_2, \ldots, b_N]$: quality levels of the tiles of the previously downloaded segment. $b_i = 0$ for tiles that were not downloaded.
- c_t: number of segments remaining at the time of downloading this segment.
- α_t: video player buffer size in seconds of playback. i.e. the duration of the video segments that have been downloaded but waiting in the buffer to be played.

These input features for the state representation are also depicted in Figure 2.

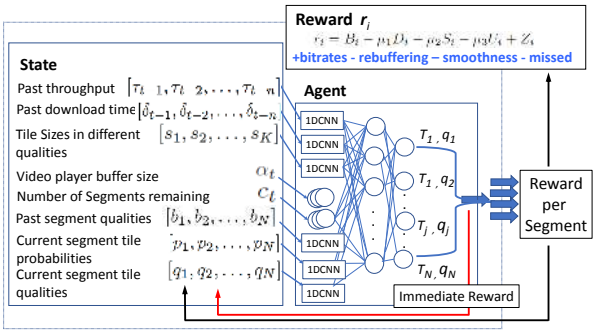


Figure 2: High-level network architecture of *ATRIA*.

3.2. Action Space

We consider two action spaces, leading to adaptive streaming methods: *ATRIA* and *ATRIA*-2.

***ATRIA*: Adaptive tile ordering.** The RL network selects the tiles to download and their download quality for each action. Tiles are allowed to be downloaded more than once, e.g., first with low encoding quality and subsequently with higher encoding quality when network capacity gets better. Some tiles can be skipped (never downloaded). We let the *agent* learn what is the best action in each step that maximizes the accumulated rewards. The action a_t at time t is a pair of positive integers (a_t^1, a_t^2). The first quantity a_t^1 is the ID of the tile that should be downloaded, and a_t^2 is the chosen encoding quality level for downloading. In this work, the policy function π_θ is a deep neural network with learnable parameter set θ. The input to the network is the state representation vectors as in Section 3.1. The last layer of the network is a softmax layer, and the output of the network is a probability vector of size $N(K+1)$, where N is the number of tiles and K is the number of video quality levels. The tile ID a_t^1 and the download quality a_t^2 are determined together, not sequentially or independently. We illustrate this in Figure 2.

***ATRIA*-2: Fixed tile ordering.** This policy is similar to *ATRIA*, but downloads the tiles sequentially in a fixed order, i.e., tiles $1, 2, \ldots, N$ in that order. At each step, the tile to download is already determined, and the policy only needs to determine the bitrate quality to download. It is also possible for the policy to decide not to download a tile. Therefore, at each action a_t at time t, there are $K+1$ action space, where K is number of different video bitrates offered. This significantly reduces the action spaces compared to *ATRIA* action space. *ATRIA*-2 also learns a probabilistic policy: at time t, given the state s_t, action a_t is selected with a probability value given by the function $\pi_\theta(a_t|s_t)$. In this work, the policy function π_θ is a deep neural network with learnable parameter set θ. The input to the network is the state representation vectors as in Section 3.1. Figure 2 also illustrates *ATRIA*-2 except for the last layer of action space. Table 1 compares the design approaches of *ATRIA*, *ATRIA*-2, and a regular method (selects the qualities for N tiles).

	Regular	*ATRIA*	*ATRIA*-2
Space Size	$(K+1)^N$	$N(K+1)$	$K+1$
Action	$(a_t^1, .., a_t^N)$ $a_t^i = 0..K$	(a_t^1, a_t^2) $a_t^1 = 1..N, a_t^2 = 0..K$	a_t $a_t = 0..K$

Table 1: Summary of *ATRIA* & *ATRIA*-2 design approaches.

3.3. Reward Function

The reward function is defined based on the quality of viewer experience, which is high when the video bitrate is high, the rebuffering time is small, and the playback is smooth (both spatially and temporally).

Viewer Perceived Video Bitrate. We define viewer perceived video bitrate as the sum of qualities of tiles that overlaps with the viewport, i.e., tiles that the user actually views. Tiles that are downloaded but not viewed do not count toward the user perceived video bitrate. We assume all viewed tiles contributes to the quality of experience equally. Mathematically, for the i^{th} video segment, the perceived bitrate B_i during playback is given by:

$$B_i = \sum_{j=1}^{N} \beta_{ij} o_{ij}, \tag{1}$$

where β_{ij} is the bitrate of tile j of segment i, and $o_{ij} = 1$ if tile j overlaps with the viewport, and 0 otherwise.

Rebuffering. If the playback buffer has depleted but the next video segment has not been downloaded completely, playback stalls and the viewer experiences rebuffering. We measure the rebuffering duration for each tile j being downloaded. We define the rebuffering duration during the download of a segment as the sum of rebuffering duration for each tile for this segment:

$$D_i = \sum_{j=1}^{N} d_{ij}, \tag{2}$$

where d_{ij} is the rebuffering duration of downloading tile j of segment i.

Quality Smoothness Across Segments. Similar to conventional video streaming, quality fluctuation between the segments also degrades the QoE. We model the quality smoothness between segments as the difference of user perceived video bitrates of two consecutive segments. Let B_i represents the user perceived video bitrate of the i^{th} segment. The smoothness measure is defined as:

$$S_i = |B_i - B_{i-1}|, \text{ for } i \geq 2. \tag{3}$$

Quality Smoothness Within Viewport. Unlike conventional video streaming, user might view the scene with different levels of qualities in tiled 360-degree video streaming. This is inevitable, unless the system downloads all tiles

in the same quality level. We measure this quality smoothness within a viewport as a variance among the tiles that overlaps with viewport:

$$U_i = \text{StdDev}\left\{\beta_{ij} | o_{ij} = 1\right\} \tag{4}$$

Penalty for repeating/missing tiles. *ATRIA* allows a tile to be re-downloaded with a different encoding quality if the network condition allows. Rationally, a tile should not be re-downloaded with a lower quality than what has been downloaded, so we penalize this undesirable action by giving an immediate reward of -1.

If the policy does not select tiles that overlap with the viewport (i.e., missing titles) for segment i, we also assign a negative reward.

$$Z_i = -\sum_{j=1}^{N} o_{ij} R_{ij}(k)\, \delta(q_{ij} = 0), \tag{5}$$

where $R_{ij}(k)$ is the bitrate of tile j of segment i at quality level k. q_{ij} is the quality level downloaded for the tile j of segment i; q_{ij} is 0 if the title is not downloaded. $\delta(\cdot)$ is the delta function; $\delta(x) = 1$ if x is true and 0 otherwise. k is set to be the average quality level of the viewed tiles: $k = round((\sum_{j=1}^{N} o_{ij} q_{ij})/(\sum_{j=1}^{N} o_{ij}))$.

User Perceived Quality of Experience (QoE). Following [40, 23] and also [28, 26, 42], the total user perceived QoE is defined as the sum of user perceived bitrates, rebuffering, and smoothness for all M video segments:

$$QoE = \sum_{i=1}^{M} (B_i - \mu_1 D_i - \mu_2 S_i - \mu_3 U_i). \tag{6}$$

where the μ_1, μ_2, and μ_3 are constants modeling contributions of rebuffering and quality smoothness on the QoE.

Reward. We define the reward r_i for a video segment i based on the perceived QoE for this segment and the penalty for having repeating or missing tiles.

$$r_i = B_i - \mu_1 D_i - \mu_2 S_i - \mu_3 U_i + Z_i, \tag{7}$$

The reward is defined for each video segment, collectively for all tiles after they have been downloaded and viewed. This reward is not defined for an individual download step, and this corresponds to having delayed rewards.

3.4. Network Architecture and Policy Learning

Network Architecture. Figure 2 shows the high level architecture of the actor network. The critic network has the same architecture, except for the linear neuron output. All non-output layers are the same architecture at the actor and the critic network. We use 1D CNNs to encode the history of past values, including throughput measurements and download times. We also use 1D CNNs to encode tile features such as probabilities, file sizes, and selected qualities. The filter size is 4 and number of filters is 128. For the number of remaining segments c_t and the buffer size α_t, we use a 128-dim fully connected layer with RELU activation. The outputs of these layers of actor network are connected to a hidden layer with 128 units, followed by the softmax layer.

Policy Learning. To learn a streaming policy, we use A3C [24], a state-of-the-art RL method. A3C is a policy gradient method [11] that asynchronously trains two agents (both are neural networks): an *actor* and a *critic*. Each agent has its own copy of the environment. The actor agent is the policy function for video streaming: in *ATRIA*, it takes the state as input and decides the tile to download and the download quality (a_t^1, a_t^2) while in *ATRIA*-2, it decides the download quality a_t. The critic agent is trained to estimate the value of the state, called *value function*. It evaluates how good the policy is by estimating the expected total reward for a given state s and and the policy π_θ. We use the *Temporal Difference* method [11] to train the critic network.

We use TensorFlow [2] and TFLearn [3] to implement neural network for both training and testing. The discount factor γ is set to 0.99. We configure the entropy parameter weight to encourage the agent to explore and to exploit its knowledge about the state space and environment. Both *ATRIA* and *ATRIA*-2 converges. We observe that *ATRIA*-2 converges faster.

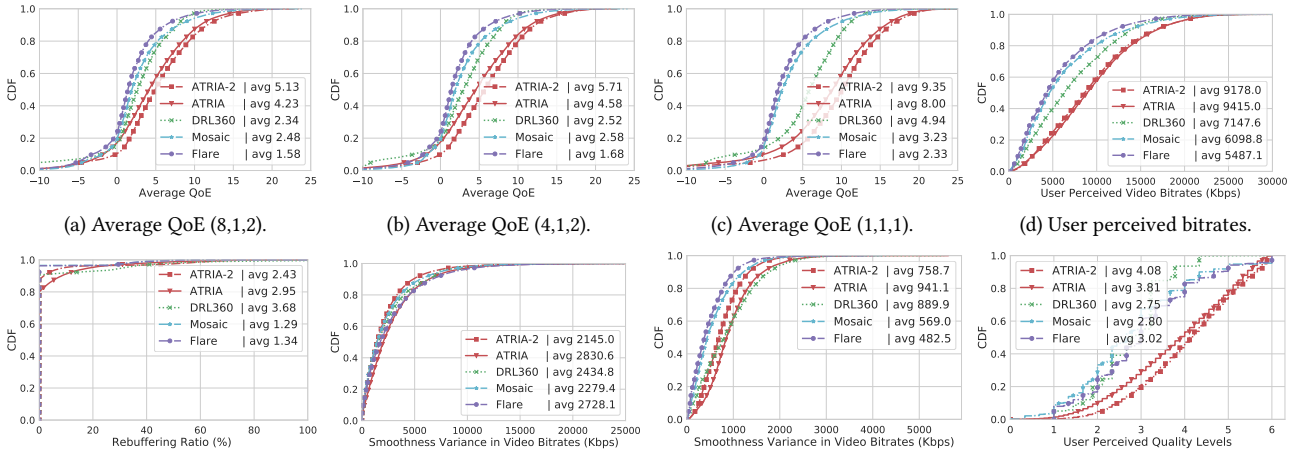
4. Experiments

We compare our methods quantitatively and qualitatively with three state-of-the-art 360-degree tile-based video streaming algorithms: Mosaic [26], Flare [28], and DRL360 [42]. These methods represent the entire gamut of relevant approaches for bit rate adaptation for 360-degree video streaming, and they have been shown to achieve higher QoE than conventional non-tile video streaming methods. The methods are evaluated on testing environments that emulate real network conditions.

4.1. Experiment settings

Emulated streaming environments. We use ten videos for training and evaluating various streaming algorithms. Each video is one-minute long and encoded with six different bit rates: 0.512, 2, 5, 10, 15, and 20 Mbps. Each video is split temporally in two-second segments. Each video is accompanied with head tracking data from 50 viewers [22]. The head tracking data is used as the ground truth to evaluate the viewing experience of a viewer, not as the input to any streaming algorithm.

For each pair of video and the head tracking data, we



(a) Average QoE (8,1,2). (b) Average QoE (4,1,2). (c) Average QoE (1,1,1). (d) User perceived bitrates.

(e) Rebuffering. (f) Smoothness across segments. (g) Smoothness within viewport. (h) User perceived quality levels.

Figure 3: Comparison of *ATRIA* and *ATRIA*-2 with other baseline techniques using CDFs of various metrics: (a)-(c) average QoE (μ_1, μ_2, μ_3), (d) user perceived bitrate, (e) rebuffering ratio, (f) temporal smoothness, (g) viewport smoothness, (h) user perceived video quality levels. A point (x, y) on a performance curve means that the metric has a value $\leq x$ for a fraction y of the times.

can emulate different streaming conditions and calculate the quality of experience. The emulated environments are based on about 300 real network traces of public datasets: broadband dataset [1] and a mobile dataset [29]. We linearly increase the bandwidth to reflect prevalent Internet connection speed [5]. We use a large corpus of real network traces for training and a different set of network traces for testing. In all experiments reported in this paper, the training and testing network environments are separate. This amount to 135,000 video steaming sessions (equivalent to 2,250 hrs, 3 months).

Viewport Prediction. Predicting where a viewer will attend to is an active area of research, e.g., [26, 41, 39]. In this work, we use the viewport prediction network developed by [26]. This is a 3DCNN network that uses the I3D network for human action recognition [8]. The inputs to the network are: a saliency map, a motion map and the user's head tracking history dataset [10]. The outputs are the probabilities for the tiles to be viewed by the user. The prediction accuracy is 91–92% for the prediction lead time of one second, and 88% for the lead time of two seconds [26].

The overhead for running the viewport prediction network and the RL policy are less than 0.6ms and 0.1ms respectively. The overhead for encoding video tiles is proportional to the number of tiles, and we use the 4×6 tiling as in previous work [14, 26, 28]. Because the viewport prediction network and the RL policy are trained off-line, the overhead of the inference on the client is small, and the proposed solution is feasible to run even on a smartphone.

4.2. Quantitative Comparison Results

We randomly split the ten videos into two disjoint training and testing subsets, each with five videos. Additionally, the network traces used for emulating the network conditions for training data are not used for testing.

Quality of Experience. In computer networking and multimedia community, it is challenging to quantify the quality of experience. There is no rigorous formulation to compute a numerical score for the experience of an user based on measurable values such as perceived bitrates and rebuffering time. Following [40, 23], we experiment with multiple values of the QoE parameters μ_1, μ_2, and μ_3 defined in Eq. (6). Specifically, the ranges of these parameters are set to 1–8, 1–4, and 1–4 respectively. The reward function is defined accordingly, based on the QoE parameters.

Figure 3a, 3b, and 3c plot the QoE distributions of different streaming methods for three different QoE parameter settings. As can be seen, both *ATRIA* and *ATRIA*-2 have significantly higher QoE than other methods for all parameter settings in consideration. In terms of average QoE, our methods are 1.6 to 3.4 times better than Mosaic, DRL360, and Flare. We perform experiments with different QoE parameter settings and observe similar results, but the plots cannot be shown here due to the space limit.

Individual QoE Metrics. We also analyze the performance of individual QoE metrics of difference techniques. Figure 3d shows the average bitrate perceived by viewers for different algorithms, where a higher bitrate is more desirable. On average, the bitrate of *ATRIA* (and similarly for *ATRIA*-2) is 1.3, 1.5, and 1.7 times higher than the bitrates of DRL360, Mosaic, and Flare respectively.

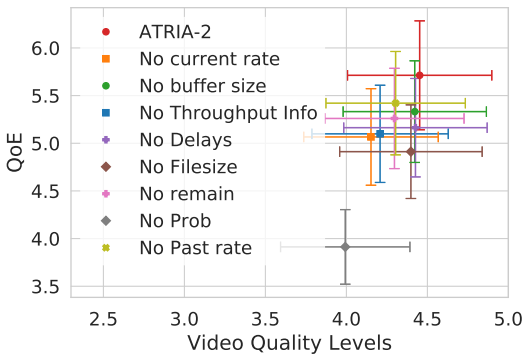


Figure 4: Ablation study for the importance of individual feature types. The QoE and video quality values decrease if any input feature is not used.

Figure 3e shows the ratio of rebuffering for different streaming techniques, where a lower ratio indicates a better QoE. All streaming methods have a similar buffering duration, which is about 1–4% of the video duration or 0.6–2.4 seconds of 60-second video. The quality variance is another important indicator for the QoE, which measures the smoothness of the video quality over time (lower is better). Figure 3f shows the standard deviation across segments of a video. We note that *ATRIA*-2 and DRL360 have similar quality variance values. Figure 3g plots the quality variation within the viewport. While *ATRIA* and *ATRIA*-2 are not the best in terms of smoothness metric, they have higher bitrates and overall QoE values. *ATRIA*-2 has higher QoE than *ATRIA* because *ATRIA* allows re-downloading of tiles to maximize the reward (i.e., previously downloaded tile is not used). The addition of a simple rule (the order of tile to download) helps the agent learn faster while ensuring that agent explores different actions and exploits what it has learned.

4.3. Ablation Studies

As described in Section 3.1, the state representation of our RL streaming algorithms incorporates eight types of features. We evaluate the impact of each feature type by excluding it from the state representation. Figure 4 plots the resulting video quality and QoE values for *ATRIA*-2. Removing any feature type would reduce both video quality and the overall QoE. The most important feature is the viewport prediction probabilities.

4.4. Generalization Ability

In all experiments described above, we have evaluated the performance of all methods on the test videos that are different from the videos used for training. But for many situations, videos at a streaming server will be watched by multiple viewers, and it not unpractical to use those videos to train a customized streaming algorithm. Thus one might wonder if there is any performance gap for streaming a

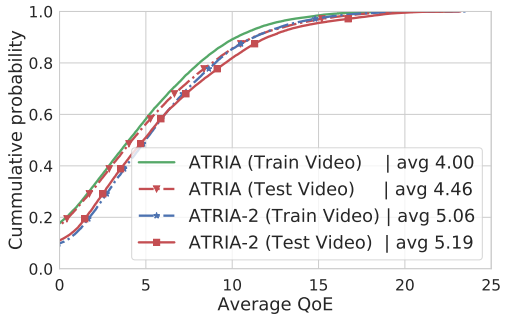


Figure 5: Comparing the average QoE values for streaming videos used and not used in the training of the RL policies for adaptive streaming.

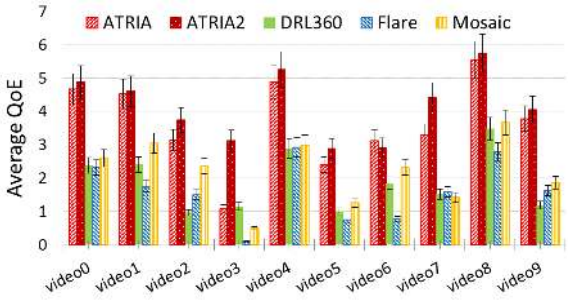


Figure 6: QoE values for streaming different videos. In this experiment, only Video0 is used for training the adaptive streaming algorithms *ATRIA* and *ATRIA*-2.

video that is among the training set and a video that is outside the training set.

Figure 5 compares the average QoE for streaming videos used in training and the average QoE for streaming videos not used in training. Interestingly, the two performance curves are similar, suggesting that it is unnecessary to train *ATRIA* on any specific video and that *ATRIA* can generalize well on unseen videos. This is understandable because *ATRIA* makes the streaming decision based on the predicted user behavior, the network conditions, and the state of the buffer, not the content of the video. Notably, in all experiments, the training and testing environments are independent, emulated based on disjoint sets of real network traces.

Given the good generalization ability of *ATRIA*, we experiment with an extreme situation where *ATRIA* is trained with a single video, Video0. We compute the QoE for streaming Video0 and other videos, and the results are shown in Figure 6. *ATRIA* obtains higher QoEs on Video4 and Video8 than on the training video Video0. This further confirms the good generalization ability of *ATRIA*. Figure 6 also compares the performance of *ATRIA* with other state-of-the-art streaming algorithms. As can be seen, *ATRIA* outperforms these algorithms, even when it is trained with a single video.

(a) Coaster using *ATRIA*-2 (b) Coaster using DRL360 (c) Game using *ATRIA*-2 (d) Game using DRL360

Figure 7: Frame captures of video session for two videos each with *ATRIA*-2 and DRL360. Video frames at (b) and (d) show that parts of the viewport have poor image quality (yellow rectangular areas) for DRL360.

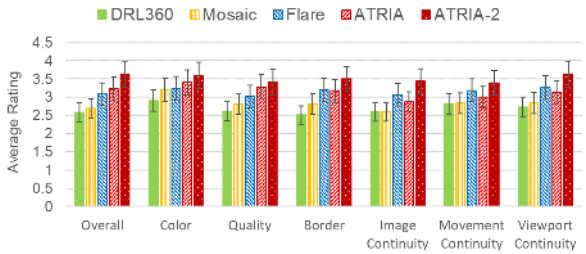


Figure 8: Average user ratings for seven quality categories.

4.5. Subjective Evaluation: User Study

We also compare the actual quality of experience with an user study. We follow the recommendations for *subjective video quality assessment methods for multimedia applications* in [20]. We use a recommended method, called Absolute Category Rating (ACR) to evaluate different algorithms and to rank the video system performance and quality levels [20]. Each user participant is presented with test sequences, one at a time with five different methods in randomized order. The participant is asked to evaluate the overall quality of each video, in a five-level scale: 5-Excellent, 4-Good, 3-Fair, 2-Poor, 1-Bad. The participant is also asked to evaluate the quality of: image color, compression quality, borders, image continuity, and movement continuity. Unless all tiles are downloaded in a same compression quality level, the user could view a scene with different quality levels. The participant is asked if they notice any discontinuity within a viewport and asked to rate in a five-level scale: 5-Imperceptible, 4-Perceptible but not annoying, 3-Slightly annoying, 2-Annoying, and 1-Very Annoying. The higher the rating is, the better image continuity within the viewport is observed.

We recruit fourteen participants (11 male, 4 female, from 10–50 year old) for our user study. Each participant views ten videos with five different methods (*ATRIA*, *ATRIA*-2, Flare, Mosaic, DRL360) for a total of 50 videos.

Figure 7 shows some frames presented to the participants in the user study. As can be seen, the frames from the DRL360 (Figure 7b and 7d) have much lower quality than the frames from *ATRIA*-2 (Figure 7a and 7c), especially inside the yellow rectangular regions.

Figure 8 shows the average user ratings for several quality categories for five streaming methods. In terms of Overall Quality, both *ATRIA*-2 and *ATRIA* exhibit superior performance compared with other methods. *ATRIA*-2 is consistently rated highest in all evaluation categories. *ATRIA* has higher Overall Quality, Image Color, and Image Quality than Flare, Mosaic, and DRL360. Recall from Figure 3d, *ATRIA* has the highest average user perceived bitrate. However, the smoothness variance across segments and within the viewport is higher than that of Flare (Figure 3f and 3g). This explains why *ATRIA* has lower ratings than Flare in terms of Image Continuity, Movement Continuity, and Viewport Continuity.

5. Conclusions

We have presented two methods for adaptive streaming of 360-degree videos. Unlike prior works that use pre-determined rules for rate adaptation, our methods are based on deep reinforcement learning, and they can dynamically determine which tiles to download at what qualities and when, depending on the network conditions. We have evaluated our methods in realistic settings that emulate the real network conditions. We have compared our methods against state-of-the-art 360-degree tiled video streaming techniques, and showed that our methods outperform the other methods by a factor of 1.6–3.4 in terms of average QoE and a factor of 1.3–1.7 in terms of perceived bitrates. We have also performed a subjective user study and found that our methods have the highest overall ratings among all methods. Further improvement can be obtained by increasing the smoothness within the viewport, and it will be a good direction for future work.

Acknowledgments

This research was partially supported by Intelibs, Inc. Minh Hoai was supported by NSF Award IIS-1763981. The authors acknowledge the time and effort of the volunteer user study participants.

References

[1] Federal communications commission. 2016. raw data - measuring broadband america. (2016). Technical report.

[2] TensorFlow. https://www.tensorflow.org, 2018.

[3] FFmpeg. https://ffmpeg.org/, 2019.

[4] Shahryar Afzal, Jiasi Chen, and KK Ramakrishnan. Characterization of 360-degree videos. In *Workshop on VR/AR Network*. ACM, 2017.

[5] Akamai. Akamai's: 2018 State of the Internet / Connectivity Report . 2018.

[6] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. Oboe: auto-tuning video ABR algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018.

[7] Yanan Bao, Huasen Wu, Tianxiao Zhang, Albara Ah Ramli, and Xin Liu. Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. In *Big Data (Big Data), IEEE International Conference on*. IEEE, 2016.

[8] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[9] Federico Chiariotti, Stefano D'Aronco, Laura Toni, and Pascal Frossard. Online learning adaptation strategy for DASH clients. In *International Conference on Multimedia Systems*, 2016.

[10] Ching-Ling Fan, Jean Lee, Wen-Chih Lo, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. Fixation prediction for 360 video streaming in head-mounted virtual reality. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2017.

[11] Vincent Francois-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4), 2018.

[12] Jun Fu, Xiaoming Chen, Zhizheng Zhang, Shilin Wu, and Zhibo Chen. 360SRL: A sequential reinforcement learning approach for ABR tile-based 360 video streaming. In *IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019.

[13] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella. D-DASH: A deep q-learning framework for DASH video streaming. *IEEE Transactions on Cognitive Communications and Networking*, 3(4), Dec 2017.

[14] Mario Graf, Christian Timmerer, and Christopher Mueller. Towards bandwidth efficient adaptive streaming of omnidirectional video over http: Design, implementation, and evaluation. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 2017.

[15] Yu Guan, Chengyuan Zheng, Xinggong Zhang, Zongming Guo, and Junchen Jiang. Pano: Optimizing 360 video streaming with a better understanding of quality perception. In *Proceedings of the ACM Special Interest Group on Data Communication*. 2019.

[16] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. Rubiks: Practical 360-degree streaming for smartphones. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 2018.

[17] Mohammad Hosseini and Viswanathan Swaminathan. Adaptive 360 VR video streaming: Divide and conquer. In *Multimedia (ISM), IEEE International Symposium on*. IEEE, 2016.

[18] Tianchi Huang, Rui-Xiao Zhang, Chao Zhou, and Lifeng Sun. Qarc: Video quality aware rate control for real-time video streaming based on deep reinforcement learning. *arXiv:1805.02482*, 2018.

[19] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. *ACM SIGCOMM Computer Communication Review*, 44(4), 2015.

[20] P ITU-T RECOMMENDATION. Subjective video quality assessment methods for multimedia applications. *International telecommunication union*, 1999.

[21] Nuowen Kan, Junni Zou, Kexin Tang, Chenglin Li, Ning Liu, and Hongkai Xiong. Deep reinforcement learning-based rate adaptation for adaptive 360-degree video streaming. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[22] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 360 video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. ACM, 2017.

[23] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017.

[24] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 2016.

[25] Omar A Niamut, Emmanuel Thomas, Lucia D'Acunto, Cyril Concolato, Franck Denoual, and Seong Yong Lim. MPEG DASH SRD: spatial relationship description. In *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016.

[26] Sohee Park, Arani Bhattacharya, Zhibo Yang, Mallesham Dasari, Samir R Das, and Dimitris Samaras. Advancing user quality of experience in 360-degree video streaming. In *IFIP Networking Conference (IFIP Networking)*. IEEE, 2019.

[27] Sohee Kim Park, Arani Bhattacharya, Mallesham Dasari, and Samir R Das. Understanding user perceived video quality using multipath TCP over wireless network. In *IEEE 39th Sarnoff Symposium*. IEEE, 2018.

[28] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of MobiCom*. ACM, 2018.

[29] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*. ACM, 2013.

[30] Lucile Sassatelli, Marco Winckler, Thomas Fisichella, and Ramon Aparicio. User-adaptive editing for 360 degree video streaming with deep reinforcement learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, 2019.

[31] S. Sengupta, N. Ganguly, S. Chakraborty, and P. De. HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning. In *ICNP*, 2018.

[32] Iraj Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE multimedia*, 18(4), 2011.

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

[34] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2^{nd} edition, 2018.

[35] Boyu Wang, Lihan Huang, and Minh Hoai. Active vision for early recognition of human actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[36] Gongwei Xiao, Xu Chen, Muhong Wu, and Zhi Zhou. Deep reinforcement learning-driven intelligent panoramic video bitrate adaptation. In *Proceedings of the ACM Turing Celebration Conference-China*, 2019.

[37] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming Guo. 360ProbDASH: Improving QoE of 360 video streaming using tile-based HTTP adaptive streaming. In *Proceedings of the ACM on Multimedia Conference*. ACM, 2017.

[38] Lan Xie, Xinggong Zhang, and Zongming Guo. CLS: A cross-user learning based system for improving QoE in 360-degree video adaptive streaming. In *2018 ACM Multimedia Conference on Multimedia Conference*. ACM, 2018.

[39] Zhibo Yang, Lihan Huang, Yupei Chen, Zijun Wei, Seoyoung Ahn, Gregory Zelinsky, Dimitris Samaras, and Minh Hoai. Predicting goal-directed human attention using inverse reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[40] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. *ACM SIGCOMM Computer Communication Review*, 45(4), 2015.

[41] Gregory Zelinsky, Zhibo Yang, Lihan Huang, Yupei Chen, Seoyoung Ahn, Zijun Wei, Hossein Adeli, Dimitris Samaras, and Minh Hoai. Benchmarking gaze prediction for categorical visual search. In *CVPR Workshop - Mutual Benefits of Cognitive and Computer Vision*, 2019.

[42] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and XiaoMing Li. DRL360: 360-degree video streaming with deep reinforcement learning. In *Proceedings of IEEE Infocom*. IEEE, 2019.