Adaptive Bitrate with User-level QoE Preference for Video Streaming

Xutong Zuo*, Jiayu Yang†, Mowei Wang*, Yong Cui*‡
*Tsinghua University, China
†Beijing University of Posts and Telecommunications, China

***Abstract*—Recent years have witnessed tremendous growth of video streaming applications. To describe users' expectations of videos, QoE was proposed, which is critical for content providers. Current video delivery systems optimize QoE with ABR algorithms. However, ABR is usually designed for an abstract "average user" without considering that QoE varies with users. In this paper, to investigate the difference in user preferences, we conduct a user study with 90 subjects and find that the average user can not represent all users. This observation inspires us to propose Ruyi, a video streaming system that incorporates preference awareness into the QoE model and the ABR algorithm. Ruyi profiles QoE preference of users and introduces preference-aware weights over different quality metrics into the QoE model. Based on this QoE model, Ruyi's ABR is designed to directly predict the influence on metrics after taking different actions. With these predicted metrics, Ruyi chooses the bitrate that maximizes user-specific QoE once the preference is given. Consequently, Ruyi is scalable to different user preferences without re-training the learned models for each user. Simulation results show that Ruyi increases QoE for all users with up to 65.22% improvement. Testbed experimental results show that Ruyi has the highest ratings from subjects.**

***Index Terms*—video streaming, bitrate adaptation, quality of experience, deep learning**

I. Introduction

With the emergence of new applications such as video conferences and 4K videos, the volume of video streaming traffic has increased rapidly in recent years [1]. As reported by Cisco, video traffic could make up as much as 82% Internet traffic by 2022 [2]. Meanwhile, user demand on video quality has been on the rise. In order to increase revenue, content providers make great efforts to meet users' expectation with the limited network resources.

Quality of Experience (QoE) is used to describe user expectations and has been a rapidly evolving research topic in adaptive video streaming. Aiming to describe users' perception, many QoE models are proposed, including learning-based methods [3], [4] and parametric methods [5], [6]. Learning-based methods can automatically generate the desired models while parametric methods are simple in form and do not require much data. With the QoE model as the optimization objective, Adaptive BitRate (ABR) algorithms have been proposed and make great breakthroughs [5], [7]–[9]. The effectiveness of ABR algorithms is attributed to the fact that ABR splits the videos into chunks, encodes chunks into several quality levels and selects the bitrate for each chunk according to network conditions.

As a simple observation, users have different preferences towards various QoE metrics. When the bandwidth resources are limited, the videos will be distorted with different levels of rebuffering, visual quality and switching, etc. In this condition, users usually have different tolerances for the above distortion types. In this paper, we **refer to the tradeoff among these distortion types as user preference**. For example, some users would rather tolerate the rebuffering than watch low visual quality videos, whereas some users are on the opposite. However, user preferences have mostly been neglected in existing QoE models and ABR algorithms, which may hinder the performances. Specifically, most QoE models assume that all users have the same preferences. They calculate the Mean Opinion Score (MOS) and regard it as the score of an "average user" [3], [4], [10]. Although some works take user preferences into consideration of video images [11], temporal impairments, like rebuffering, which are key factors in adaptive video streaming are not included. Besides, traditional ABR algorithms are also agnostic to user preferences, because they usually optimize towards a fixed QoE model [5], [7]–[9].

With the simple observation that QoE preferences vary with users, a natural question arises: how different users' QoE preferences are? To answer this question, we first conduct a user study of 90 subjects in Section II-A, followed by the key findings (Section II-B). Notably, we find non-negligible differences among QoE preferences, and thus an average subject can not represent all users.

Based on the findings in the user study, we argue that user preferences on QoE should be considered. To achieve this, ABR should optimize for each user with the preference-aware QoE model. To achieve this, we propose *Ruyi*[1], a video streaming system that incorporates user preferences into both the QoE model and the ABR algorithm.

We first profile the unique user preference of video quality and try to improve prediction accuracy of QoE model. For compatibility, Ruyi takes a pragmatic method, which supports adaptable changes to the existing QoE models. Particularly, Ruyi is based on the representative additive QoE models which account for a large proportion of existing QoE models [5], [6],

‡Yong Cui is the corresponding author.

[1]Ruyi, which means as you wish in mandarin, is a talisman symbolizing good fortune.

[8], [10], [12], [13]. It can consist of some meta metrics (e.g., VMAF, rebuffering, smoothness) and weights over them [5], [8], [14]. For Ruyi, we propose preference-aware weights. To estimate the weights, we collect quality ratings directly from real users with in-lab rating. The subjects are required to watch different rendered videos distorted by various network conditions as well as bitrate adaptation algorithms and rate them. Based on the ratings, we can infer the weights. With the preference-aware weights, the prediction accuracy of our QoE model is improved compared to that of the QoE model designed for the average user.

Then we consider to incorporate preference-aware ABR into a video streaming system. However, existing state-of-the-art ABR algorithms fail to achieve this. Rule-based approaches (e.g., buffer-based approach [6], [7] and rate-based algorithms [15], [16]) inherently can not be generalized over different QoE preferences, since their optimization objectives are predetermined and fixed during the design phase. Data-driven methods (e.g., MPC [5], Fugu [9] and Pensieve [8]) can support flexible QoE objectives but lack scalability. In offline phase, they often learns a representation of the state-decision mapping (table or neural network) that is optimized for the predetermined QoE objective. In online phase, the decision can be quickly obtained according to the representation. However, each representation only supports one objective unless reconstructed, and thus it is infeasible to support a huge number of QoE objectives.

As defined above, user preference refers to the tradeoff among different metrics. Nevertheless, the state-of-the-art data-driven methods usually map the states and QoE metrics into a scalar reward (Pensieve based on reinforcement learning (RL) [8]) or the best action (MPC [5] or Fugu [9]). The feedback of the applied bitrate lacks detailed information of metrics. For this reason, and inspired by [17], [18], we leverage **supervised learning (SL) perspective on learning to act**, which is superior to RL when temporally vectorial feedback is available. Specifically, we directly predict the influence on each metric of taking different actions with SL under different user preferences. Our aim is that when the preference-aware QoE model is given, the preference-aware ABR algorithm optimizes for a specific user scalably. To achieve this, Ruyi's ABR algorithm is trained with multiple QoE objectives offline. In the online inference phase, Ruyi can explicitly make bitrate decisions according to the given QoE objective. In this way, the ABR achieves scalability as it is independent on user size and thus can be applicable to any number of QoE objectives without re-training.

We integrate Ruyi in our chunk-level simulator and implement Ruyi on Dash.js. Evaluation results show that preference-aware QoE model can improve the prediction accuracy compared with the QoE model designed for the average user. Ruyi achieves an improvement of more than 43.52% and 67.20% for half of the users and the users with top 30% improvement. As for end-to-end QoE, compared to the state-of-the-art algorithms, Ruyi increases QoE for all users with up to 65.22%. In testbed experiments, subjects are recruited to rate rendered videos generated by Ruyi and other comparison algorithms. Experimental results show that Ruyi have the highest ratings from subjects.

Our key contributions are:

- We conduct a user study of 90 subjects showing that there are nonnegligible differences in terms of preference among different users in Section II.
- We profile the user preference and propose the preference-aware QoE model in Section III-A.
- We propose an ABR algorithm which can deal with user preferences scalably in Section III-B to validate the effectiveness of the preference-aware QoE model.

II. Motivation

With the simple observation that QoE preferences vary with users, a natural question arises: how different users' QoE preferences are? This question is critical as diverse user preferences indicate the necessity of designing user-level QoE models and ABR algorithms. To answer this question, we first conduct a user study to collect data and then do data processing. After that we present key findings from different perspectives. Notably, we find non-negligible differences among QoE preferences, and thus an average user can not represent all users.

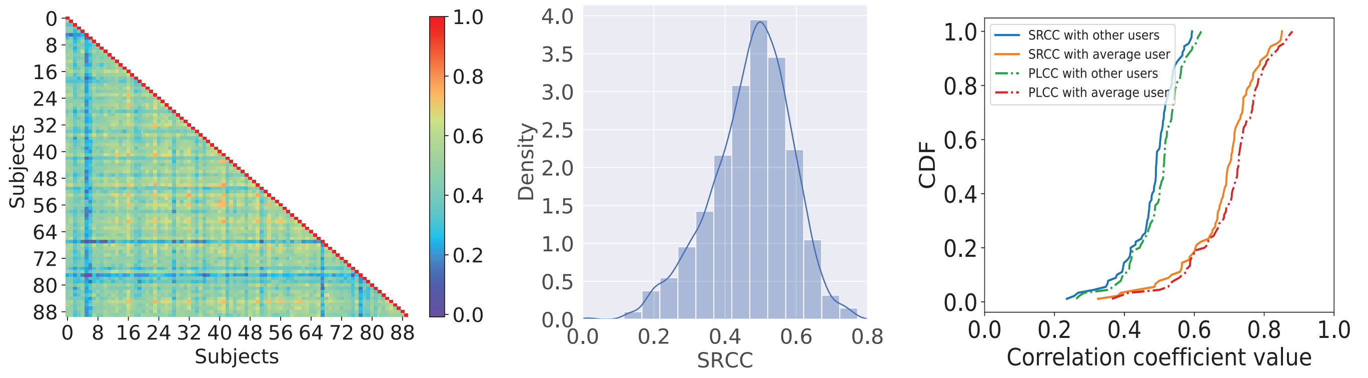
A. User study

To explore whether users have a difference in the perception of video quality, we conduct a user study in this section. We first create a set of distorted videos and then recruit some subjects at college to watch the videos and rate them.

We create a video set of 12 source videos randomly selected from two public video datasets [19], [20] of four content genres (animation, sports, nature and game) and 7 network throughput traces randomly selected from the HSDPA dataset [21] which contains actual cellular traces and is suitable for modeling challenging, low bandwidth network conditions. The chosen traces cover various network behaviors of bandwidth and variation to cause different video quality distortions. We use Traffic-Control [22] to replay the traces in Dash.js [23] and emulate the real streaming process using three ABR algorithms with different behaviours: BB (buffer-based) [7] , RB (throughput-based) [24] and Pensieve (hybrid) [8]. Finally, we get 252 ($12 \times 7 \times 3$) distorted videos.

With these distorted videos, we conduct a single-stimulus continuous quality evaluation study at college using Psychopy [25] which automates the process of playing videos and rating. We collect retrospective QoE ratings in range [1,100] on 1440p 16:9 computer monitors from a total of 90 subjects to get the individual overall QoE of each video. Overall, we gather $90 \times 252 = 22680$ retrospective ratings for analysis. More details can be found in Section III-A.

Each subject watches and rates the same video set, so we can calculate the correlation of ratings for each pair of subjects to explore how different subjects feel about the same video. For the given 252 videos, each subject has 252 ratings and



(a) Rating correlation (SRCC) between subjects.

(b) Probability Distribution Function of SRCC between subject pairs.

(c) SRCC and PLCC between subjects and the average subject.

Fig. 1. Overview of rating correlation.

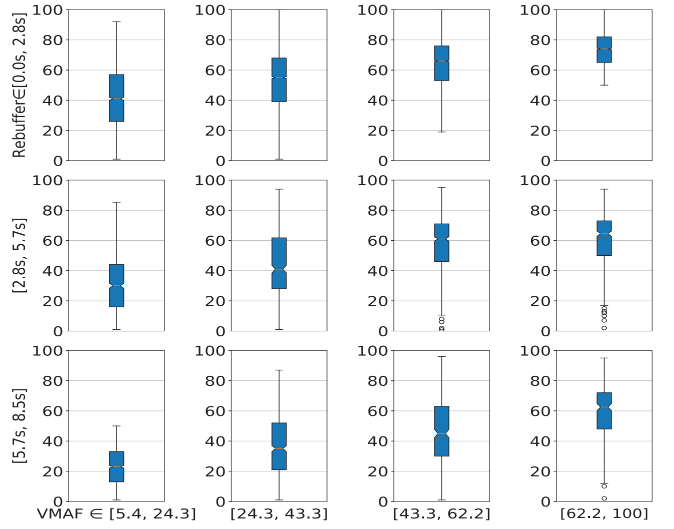


Fig. 2. Visualization of subjects' ratings by dividing the videos into different parts according to the metric levels.

the ranking of these ratings are of concerned. Spearman rank-order correlation coefficient (SRCC) can be used to indicate the correlation of the ranking. We calculate the SRCC for the subjects' rating sequences of distorted videos. The higher the SRCC, the higher the correlation between the ratings of the two subjects, the closer the subjects' perception of the video quality, and the more similar the preferences of the subjects. If the ranking of two ratings sequence is exactly the same, the SRCC is 1.0. The data processing is introduced in III-A, and only the results are introduced here.

B. Key findings

(1) Almost 90% of subject pairs are not strongly correlated. We first present the SRCC of ratings for each pair of subjects in Figure 1(a). The color of the small square at the intersection of row i and column j represents the SRCC between subject i and subject j. Higher SRCC indicates more similar preferences between subjects. Considering the symmetry of SRCC heatmap, only the part below the diagonal is shown. As shown in Figure 1(a), aside from SRCC between subject i and itself (i.e., 1.0), the largest SRCC between other pairs of subjects is 0.77. An interesting observation is that some subjects, like Subject16, share relatively similar preferences with other subjects while some subjects are not (e.g. Subject5).

We describe the strength of the correlation using the following guide for the absolute value of SRCC [26]: very weak ($SRCC < 0.2$), weak ($0.2 \leq SRCC < 0.4$), moderate ($0.4 \leq SRCC < 0.6$), strong ($0.6 \leq SRCC < 0.8$) and very strong ($SRCC \geq 0.8$). As Figure 1(b) shows, almost 90% of subject pairs have a rank correlation below 0.6, indicating most of subject pairs are not strongly correlated. Specifically, only 11% pairs of subjects are strongly correlated while almost a quarter of subjects are weakly correlated. The remaining 65% are moderately correlated.

From Figure 1(a), we observe that some subjects share relatively similar preferences with other subjects while some subjects are not. In order to characterize this difference among subjects, we calculate the average SRCC of each subject and other subjects, whose distributions are plotted as the blue line in Figure 1(c). In particular, most subjects are only moderately correlated with others, and no subjects are strongly correlated. For completeness, Pearson linear correlation coefficient (PLCC) results (i.e., green line in Figure 1(c)) are also shown, which have the same trend as SRCC.

Apart from the above analysis on overall *rating correlations*, we also investigate *rating distributions* according to different QoE metrics. Specifically, we divide the distorted videos according to VMAF and rebuffering. VMAF is divided into 4 levels, and rebuffering is divided into 3 levels. As a result, these distorted videos are divided into 12 parts. Then for videos in each part, the corresponding ratings of all subjects are shown in Figure 2. Apparently, it can be observed that subject ratings have different distributions in distinct metric levels. This observation is within expectation. Intuitively, for the videos with very high/low quality, users tend to give relatively consistent high/low ratings. Contrarily, for videos with medium quality, subject ratings may have large variances. As exemplified in Figure 2, the box in the first row and fourth

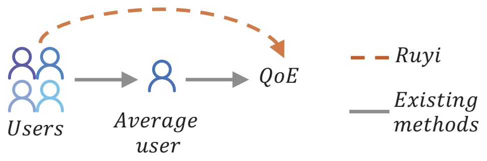


Fig. 3. Comparison of Ruyi and existing methods.

column has the highest mean score (72) with the least standard deviation (11.24). Besides, the box in the third row and third column has the largest standard deviation (21.04).

(2) The average user can not represent all users. Existing QoE models and ABR algorithms try to optimize QoE based on an average user as Figure 3 shows. Here, we further investigate whether the average user can represent all users well. First, we construct an average user whose rating for a video is the average rating of all the subjects. We calculate SRCC and PLCC between each subject and the average user, whose distributions are plotted as the red lines in Figure 1(c). More than 50% of subjects have a relatively low correlation (lower than 0.7) with the average user, indicating that the average user can not represent all users. This further enhances our motivation: when considering user preference, there should be ample room for QoE improvement. To unleash this potential, we propose the user-aware QoE model and ABR algorithm.

III. Ruyi's Design

So far we have shown that user preference is a key factor for the improvement of user QoE. To boost QoE, we propose Ruyi, a video streaming system consisting of two components (Figure 4): Preference-aware QoE model and user-specific ABR algorithm. Next we introduce these two components in Section III-A and Section III-B respectively.

A. Modeling preference-aware QoE

Different from existing average QoE models, we build a specific QoE model for each subject. Apparently, ratings are needed for each specific subject. In this work, we obtain these ratings based on the measurement study in Section II. The procedure of our preference-aware QoE modeling is described as follows. First, we describe the quality control of subject ratings. After that, we show the data processing of subject ratings. Finally, we encode user sensitivities towards different QoE metrics into the preference-aware QoE model.

Quality control of subject ratings. Since the subject ratings may be noisy, we take several principled measures to control rating quality. First, before the rating process, the subjects are required to fill out a questionnaire about QoE preferences. In this way, subjects are able to get familiar with the QoE metrics and their preferences. Besides, subjects are required to rate based on consistent criteria. To get the subjects familiar with the video quality ranges, two reference videos with the best and worst quality are played. In rating process, for different subjects, the distorted videos are randomly played to eliminate biases caused by the viewing order. To avoid subject fatigue in the nearly two-hour rating process, each subject can take a five-minute break after rating a third of

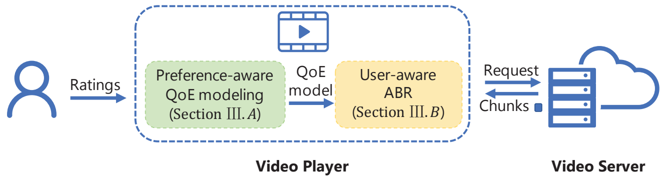

Ratings
Preference-aware QoE modeling (Section III.A)
QoE model
User-aware ABR (Section III.B)
Request
Chunks
Video Player
Video Server

Fig. 4. System overview of Ruyi.

distorted videos. Furthermore, we set up some rejection criteria to get reliable quality ratings. Specifically, we insert a video with the highest VMAF and no rebuffering as a reference video. If the subject does not give this video the highest rating, then the ratings of this subject will be rejected.

Following data collection, we process the collected ratings to eliminate biases. First, we analyze the rating sequence of each subject and find that the ratings conform to normal distribution. Then, we apply normalization to the collected ratings to prepare for the following QoE modeling. The ratings of different subjects usually fall in different ranges. For example, subjects who are accustomed to rate highly may give all ratings higher than 60. In order to normalize the subject's rating in the same range, for the rating sequence x_i, we apply $(x_i - x_{min}) \times 100/(x_{max} - x_{min})$ to each rating where x_{max} and x_{min} represent the maximum and minimum values of a subject's rating respectively.

Encode user sensitivities towards different QoE metrics. As mentioned above, Ruyi is based on the representative additive QoE models. As a result, the perceived QoE of video j for user i can be written as:

$$QoE_{ij} = \boldsymbol{w_i} \cdot \boldsymbol{q_j}, \tag{1}$$

where $\boldsymbol{w_i}$ represents the preference weights of user i and $\boldsymbol{q_j}$ denotes the QoE metrics of video j. In the additive QoE model, overall QoE is the sum of the QoE of all chunks. So we can write Equation 1 as:

$$QoE_{ij} = \boldsymbol{w_i} \cdot \sum_{k=1}^{N} \boldsymbol{m_k}, \tag{2}$$

where $\boldsymbol{m_k} = (v_k, r_k, s_k)$ is the vectorial meta metrics of chunk k and N is the number of chunks of video j. v_k, r_k and s_k represent the VMAF, rebuffer and quality switch for chunk k respectively. The range of VMAF is [0,100] and the rebuffering is counted in frame numbers. Since $\boldsymbol{w_i} = (w_{iv}, w_{ir}, w_{is})$ represents the QoE model of user i and is the same to all chunks:

$$QoE_{ij} = \sum_{k=1}^{N} (w_{iv}, w_{ir}, w_{is}) \cdot (v_k, r_k, s_k). \tag{3}$$

Next we present how to infer the weights $\boldsymbol{w_i}$ for each user. Given M rendered videos distorted by different network conditions, we get M ratings of user i, which represent the corresponding perceived QoE. With the rating QoE_{ij} and video metrics (v_k, r_k, s_k) known in advance, we can write M equations for each user i, $QoE_{ij} = \boldsymbol{w_i} \cdot \boldsymbol{q_j}$ where $j = 1, 2, ..., M$. Finally, we can infer w_i using the linear regression for user i.

B. Ruyi's ABR design

Existing ABR algorithms ignore different user preferences, which may hinder QoE improvement. To this end, we propose Ruyi's ABR algorithm in this section. Ruyi leverages the supervised learning perspective on learning to act. Specifically, Ruyi directly predicts the influence on each metric after taking different actions. In this way, Ruyi's ABR can optimize towards given user preferences.

We use a general setting where an agent interacts with an environment like in RL. At each time step t, the agent receives the observation o_t and chooses an action a_t to maximize the objective. o_t consists of two parts s_t and m_t, where s_t includes network conditions as well as video player measurements, m_t represents the meta metrics directly affecting users' QoE and a_t represents the bitrate. After applying the action, the state of the environment transits to s_{t+1} and the agent receives the influences of that action on QoE meta metrics m_t which acts as the temporally vectorial feedback.

With the definition of the preference-aware QoE model in Section III-A, the total optimization objective of Ruyi $u(\boldsymbol{m}, \boldsymbol{w})$ can be expressed as follows:

$$u(\boldsymbol{m}, \boldsymbol{w}) = \boldsymbol{w} \cdot \boldsymbol{m}, \tag{4}$$

where $\boldsymbol{w}$ represents the user preference and $\boldsymbol{m} = \langle \boldsymbol{m_t}, ..., \boldsymbol{m_{t+n}} \rangle$ represents present and future QoE metrics of different temporal offsets.

The key design is that we directly predict future measurements, i.e., meta QoE metrics $\boldsymbol{m}$, with a neural network as our evaluation function $Q(o, a, w)$. As shown in Figure 5, Ruyi is designed under the general framework for value function approximators [27]. We decouple the evaluation and decision process by explicitly introducing the user preference w into Ruyi to enable scalability. During training, Ruyi's ABR gradually learns to make better ABR decisions with preference-aware QoE objectives. During online inference phase, Ruyi chooses the best bitrate that maximize an preference-aware QoE objective according to the linear combination of the output of evaluation function and the preference w. For different users with distinct QoE models, we just modify the corresponding parameters of QoE model without modifying the ABR logic or retraining the ABR.

1) Evaluation function approximator: Leveraging the concept of Universal Value Function Approximator [27], [28], we design a customized neural network (valueNet) as our evaluation function.

Inputs: After the download of each chunk t, Ruyi's agent takes state inputs $s_t = (\boldsymbol{x_t}, \boldsymbol{\tau_t}, \boldsymbol{n_t}, \boldsymbol{v_t}, b_t, c_t, l_t)$ to its neural networks. $\boldsymbol{x_t}$ is the network throughput measurements for the past k video chunks; $\boldsymbol{\tau_t}$ is the download time of the past k video chunks, which represents the time interval of the throughput measurements; $\boldsymbol{n_t}$ is a vector of available sizes for the next video chunk; $\boldsymbol{v_t}$ is a vector of available VMAF values for the next video chunk; b_t is the current buffer level; c_t is the number of chunks remaining in the video; and l_t is the bitrate at which the last chunk was downloaded. Besides, video player measurements of the last downloaded chunk m_t are as the second part of the inputs. It includes the VMAF of the last downloaded chunk, the rebuffering caused by downloading the last chunk and the quality switch. We also add the user preference w as input which influences the optimal strategy and the final evaluation result.

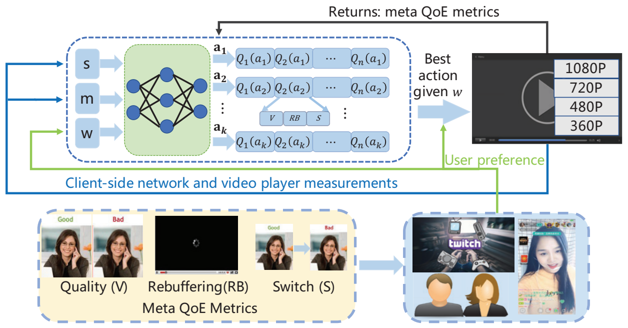

Returns: meta QoE metrics
s
m
w
$\mathbf{a_1}$
$\mathbf{a_2}$
$\mathbf{a_k}$
$Q_1(a_1)$ $Q_2(a_1)$ ··· $Q_n(a_1)$
$Q_1(a_2)$ $Q_2(a_2)$ ··· $Q_n(a_2)$
V RB S
$Q_1(a_k)$ $Q_2(a_k)$ ··· $Q_n(a_k)$
Best action given w
1080P
720P
480P
360P
User preference
Client-side network and video player measurements
Good
Bad
Quality (V)
Rebuffering(RB)
Switch (S)
Meta QoE Metrics
twitch

Fig. 5. ABR framework of Ruyi. The inputs are observations (o), meta metrics (m) and user preferences (w). The output Q is the prediction of meta metrics on different actions.

Outputs: In this work, we *directly predict the future QoE metrics*. Since we need to maximize the user's long-term QoE, we *adopt the value concept to model the meta metrics i.e.,* $\boldsymbol{m}$. To explicitly optimize for a given QoE objective, we split the conventional action value $Q(o, a)$ into several action-metric values, which only model the value for one meta metric. Thus we can predict the QoE of each action under any preference w. Recall that the standard value function is defined as the expected cumulative discounted reward over all future steps, denoted as Q_∞. However, since the network condition changes dynamically, the long-term uncertainty may hurt the prediction accuracy. Thus we use enough short-term value predictions $\{Q_t | t = 1, 2..., n\}$ as the output, where t denotes the look-ahead step. Combing the above two modifications, the output of the valueNet is $Q(o, a, w) = \{Q_t(o, a, w) | t = 1, 2, .., n\}$, where n can be adjusted according to the network condition.

Neural network architecture: The predictor Q is a deep network and the structure is shown in Figure 6. To conduct feature extraction more efficiently, we use the convolution layer for the vectorial state input and merge its output with other scalar state and measurement inputs. For the preference input w, we use fully-connected layers and merge the output with other features extracted from state inputs. Then we use a dueling architecture [29] to enhance the value prediction. Specifically, we split the features into two stream: an expectation stream E which predicts the value over this observation and an action stream A reflects the fine differences between actions. Finally, the expectation stream E is added to each of action stream A_k where k is the dimension of actions. The detailed parameters are shown in Section IV.

2) Training and inference: We train the valueNet using a variant of Deep Q-learning algorithm [30] with the experience replay technique. The agent interacts with the environment and collects a set of experiences D where $D = \{(o_j, a_j, w_j, f_j)\}_{j=1}^{T}$. (o_j, w_j) is the input of the neural network and a_j indicates the action that we are predicting

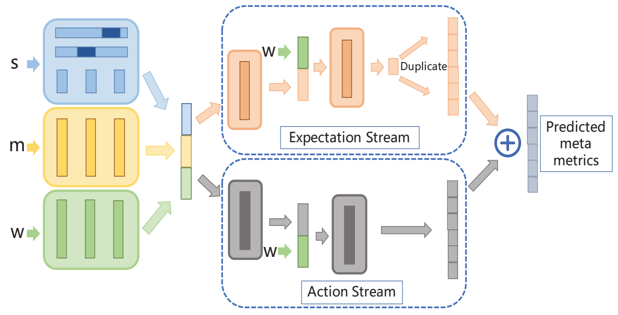


Fig. 6. Neural network structure. Inputs: network states (s), meta metrics (m) and user preferences (w). To enhance user preference, w is contacted in each neural network layer.

the subsequential influence for. f_j is the label of trajectory j which consists of multiple metric triples where each triple f_{jt} is constructed as follows:

$$f_{jt}(a) = \begin{cases} m_{j+1}, & t = 1, \\ m_{j+1} + Q_{t-1}(o_{j+1}, a, w_{j+1}), & t > 1. \end{cases} \tag{5}$$

The direct optimization objective is to minimize the following regression loss:

$$\mathcal{L}(\theta) = \sum_{j=1}^{T} \|Q(o_j, a_j, w_j) - f_j(a')\|^2 , \tag{6}$$

where $a' = \arg\max_a \; \sum_{t=1}^{n} f_{jt}(a)$.

With this objective, we train the valueNet with supervised learning to explicitly approximate the future metrics. Then we can get the best action to maximize the total optimization objective $u(\boldsymbol{m}, \boldsymbol{w})$ of our problem. Specifically, in the offline training, the user preference w for each episode is generated at random. Each value is sampled uniformly from a predefined reasonable range. The agent follows an ϵ-greedy policy: it acts greedily according to the current user preference w with probability 1-ϵ, and selects a random action with probability ϵ. The value of ϵ is initially set to 1 and is decreased during training according to a fixed schedule. In online phase, Ruyi takes the action that yields the best predicted outcome:

$$\arg\max_{a} \; w^{\top} \cdot \sum_{t=1}^{n} Q_t(o, a, w). \tag{7}$$

where $w^{\top} \cdot \sum_{t=1}^{n} Q_t(o, a, w)$ can be treated as a discounted form of the optimization objective Equation (4).

IV. Implementation

Video system implementation. We implement Ruyi based on DASH.js [23] which is a widely used open source video player. Prominent ABR algorithms, (i.e., BB, MPC and Pensieve) are configured and can run in DASH.js. We use a PC (Intel(R) Core(TM) i3-2120 CPU@3.30GHz) as the video server and a laptop (ASUS-N551JW) as the client where ABR algorithms are located. The first time to fetch a new video, the manifest file is downloaded. We augment it by adding VMAF of chunks in a new XML field which is used in the following bitrate decision. We use *tc* [22] on server to control the egress traffic and emulate the network variation. The weights representing user preference needed by Ruyi's ABR algorithm is configurable for each user at client and can be re-loaded if updated.

Parameters setting of Ruyi's ABR. Here we overview the hyperparameters in training ABR algorithm in Ruyi. Ruyi uses $k = 6$ past chunks as bandwidth measurements. The look-ahead step is $n = 3$. There are $3 \times 3 = 9$ neurons of both the expectation stream and the action stream representing 3 meta QoE metrics in 3 time offsets. There are 64 neurons in each hidden layers. The activation function is Leaky Relu and the learning rate is set to 10^{-3}. These hyperparameters are used without particularly fine-tuned. We implement Ruyi's ABR algorithm with TensorFlow [31] and TFLearn [32].

V. Evaluation

A. Experimental Setup

Network and video traces. As mentioned in Section II, we replay the streaming video in 7 traces with various bandwidth and variance from HSDPA [21] to construct the 252 distortion videos. We train the preference-aware QoE model on a subset of 216 videos obtained in the randomly selected 6 network traces (12 source videos, 6 network traces, 3 ABR algorithms, $216 = 12 \times 6 \times 3$) and test them on the remaining one network trace with medium average bandwidth, i.e., the rest 36 videos ($36 = 12 \times 1 \times 3$), to evaluate the accuracy of preference-aware QoE model. Besides, the network traces used for training two learning ABR algorithms (Pensieve and Ruyi) and testing all ABR algorithms are randomly selected from HSDPA dataset [21]. Our evaluation use the test video selected from the LIVE-NFLX-II video dataset [19]. This video is encoded by the H.264/MPEG-4 codec at bitrates in {300, 750, 1200, 1850, 2850, 4300}kbps (which correspond to video levels in {240, 360, 480, 720, 1080, 1440}p).

Comparison algorithms. We compare Ruyi with the following algorithms with different adaptive strategies:

- Buffer-based adaptation (BB) [7] without specific QoE models: chooses the chunks according to the client buffer.
- MPC [5] (with the average QoE model): chooses the chunks to maximize the QoE model of the average user in our dataset over a horizon of 5 future chunks.
- Pensieve [8] (with the average QoE model): chooses the chunks with the pre-trained learning-based ABR algorithm towards the average user. We re-train Pensieve with the average QoE model with our collected data.

Performance metrics. To show the improvement of QoE model when considering user preference, we evaluate the performance of our model with Pearson's Coefficient (PLCC) and Spearman's Coefficient (SRCC). Besides, we use QoE and QoE gain ((($Q_1 - Q_2$)/Q_2)) of one ABR (Q_1) over another (Q_2) to show the efficiency of Ruyi system.

B. Effectiveness analysis of preference-aware QoE model

We show the preference-aware weights of all 90 subjects we recruit (the red points) as well as the average user (the blue point) in Figure 7 (derived as Section III-A shows).

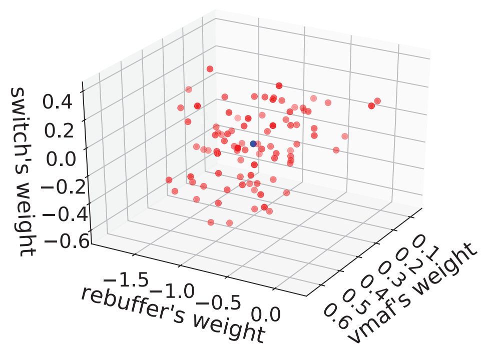


Fig. 7. QoE models of all users, which are expressed by preference-aware weights. The blue point represent the average user and red points represent all other users.

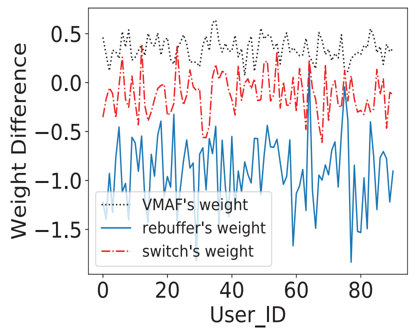


Fig. 8. Differences of the preference-aware weights between the average user and other users.

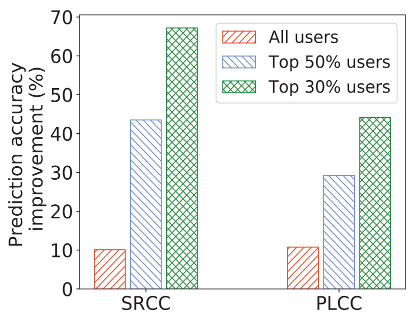


Fig. 9. QoE prediction accuracy improvement when comparing preference-aware QoE model to the average QoE model.

Apparently, it can be observed that the average QoE model is located in the center of other models. This observation is within expectation. Besides, we observe that the QoE model varies with users and it is hard to divide them into several categories. This result indicates that preference-aware QoE improvement is needed and that the average user can not represent all users. What's more, we show the difference of the three weights between each user and the average user in Figure 8. We observe that, compared with the average user, other users show the largest difference in the rebuffer weight and the smallest difference in the VMAF weight. It indicates that users show the largest preference variety in rebuffer perception while the smallest in VMAF.

Then, we present the improvement of QoE model prediction accuracy of the subjects in Figure 9. We calculate SRCC and PLCC to measure the increase, and they show the same trend. We use SRCC in the following description. For each subject, we test the preference-aware QoE model and the average model on the test dataset of 36 videos. For the test video sequences, we get the scores predicted with the preference-aware QoE model and calculate the SRCC of these scores and the ratings of subjects i ($SRCC_i$). We get the SRCC of average model in the same way ($SRCC_a$). Then the difference can be calculated as $SRCC_i - SRCC_a$ and presented in Figure 9 to show the QoE model prediction accuracy. Compared with the average QoE model, using the preference-aware QoE model brings an increase of more than 43.52% and 67.20% of QoE model prediction accuracy for half of the users and the top 30% users (the top 30% of users most impacted). The results show the effectiveness of user-aware QoE model.

C. End-to-end QoE evaluation

We evaluate Ruyi with other ABR algorithms in both simulation environment and DASH.js.

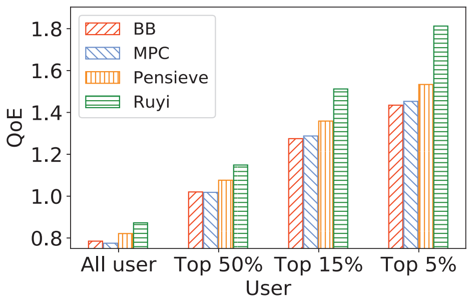
1) Simulation:

Overall QoE gains. We present the overall QoE and QoE gains in Figure 10. We find that Ruyi outperforms other algorithms which equipped with the average QoE model (MPC and Pensieve) and without QoE models (BB). Figure 10(a) shows the QoE of Ruyi, BB, MPC and Pensieve across all test network traces. From left to right are the QoE for all users, the improvement top 50% users, top 15% users and top 5% users. For top 5% users, compared to BB, MPC and Pensieve, Ruyi achieves improvements of 26.3%, 24.7%, and 18.2% respectively. Moreover, the average performance improvement decreases with the number of users increases. The reason is that some users who are close to the average user have less improvement than users who are far from the average user. For all users, compared to the baseline BB, Ruyi has 10.6% QoE improvement on average.

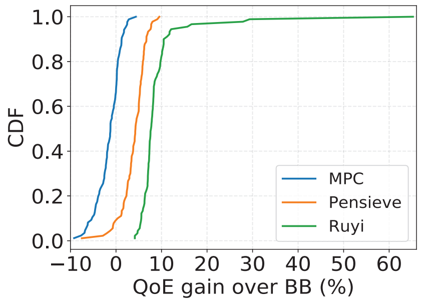
Figure 10(b) provides more detailed results which shows the distributions of QoE gains for all users. Ruyi achieves a positive gain on all users whereas MPC achieves a positive gain on 33.3% users. For Ruyi, all users achieve a performance improvement of more than 4% whereas Pensieve achieves that on 55.6% users. What's more, Ruyi obtains over 10% QoE gains on about 17.8% users with up to 65.22%. We also present that Ruyi can reach a wide range of performance by flexible adjusting the user's preference in Figure 10(c). We choose 4 example subjects whose QoE models are different ($Ruyi_1$, $Ruyi_2$, $Ruyi_3$ and $Ruyi_4$ for Subject2, 59, 85 and 41 respectively) and run experiments on 1 example testing network trace. The preference weights over different QoE metrics of the 4 subjects are (0.33, -0.92, -0.46), (0.40, -0.60, 0.32), (0.55, -0.41, -0.27) and (0.65, -0.34, 0.15). Other comparison ABR algorithms optimize towards the average user and can reach only one result point shown as red points in Figure 10(c). Meanwhile, Ruyi can reach a wide range as long as the user preferences are given.

QoE breakdown. To better understand the QoE gains obtained by Ruyi, we analyzed the achieved meta metrics for each user. Specifically, we present the preference-aware weights of each user and the achieved QoE metrics, i.e., VMAF, the rebuffering time and switches of VMAF, in Figure 11. Each dot in the figure represents a user and the value is the average of metric on all testing traces.

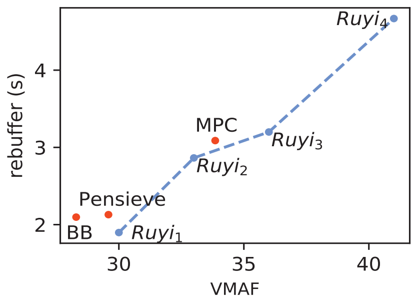
As shown in Figure 11, there is an approximate linear relationship between the achieved VMAF and the VMAF weight of different users. The larger the VMAF weight is, the more the users care about VMAF and the larger the value achieved



(a) Overall QoE of users.

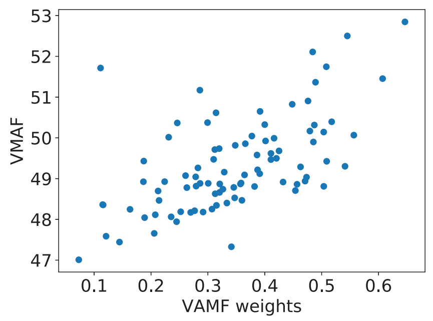


(b) QoE gains over BB of different users.

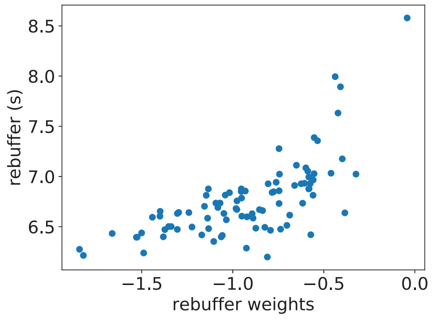


(c) Ruyi achieves a wide range of video quality flexibly.

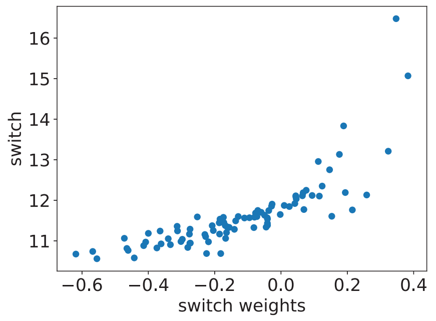
Fig. 10. Overall performance of Ruyi.



(a) VMAF



(b) Rebuffering



(c) Switch

Fig. 11. QoE breakdown of Ruyi for all users.

by the algorithm is. The achieved average VMAF across all test traces ranges from 47 to 53. Similarly, for rebuffering, when the weight is less than 0, the greater the absolute value of the weight, the less users prefer the rebuffering. Consequently, the resulted rebuffering time is smaller. Switching shows the similar results as rebuffering. Besides, we find the difference between the achieved VMAF values of users is less than that of the other two metrics, i.e., the rebuffering and the switch. This result also corresponds to Figure 8, where the weight difference of VMAF for different users is the smallest among the three metrics. Figure 11 directly proves that Ruyi has the ability to optimize towards different preferences.

Bandwidth savings. To investigate bandwidth savings, we present the QoE obtained with unit bandwidth (denoted as QoE/BW) in Figure 12. For each network trace, we calculate QoE/BW for each user and then get the average of all users on that network trace. For all testing network traces, we show the differences between QoE/BW for Ruyi and the best performing comparison algorithm Pensieve in Figure 12. Ruyi outperforms Pensieve (differences > 0) in about 72.3% traces. The average QoE/BW across all testing network traces of Ruyi and Pensieve are 0.408 and 0.376. That means Ruyi saves 8.51% bandwidth when achieving the same QoE as Pensieve. Furthermore, we calculate the average QoE/BW of the baseline BB and find Ruyi achieves 20.35% bandwidth savings.

2) Testbed Experiment:

We evaluate Ruyi and other ABR algorithms on Dash.js. The network condition is set by *tc* [22] with the random

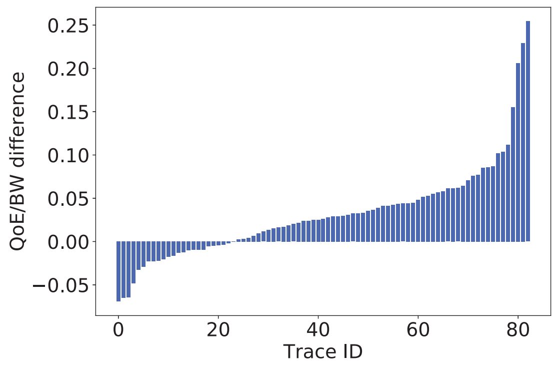
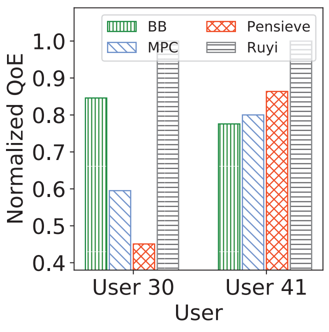
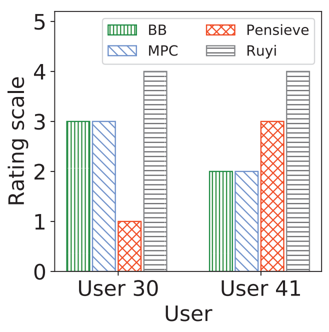


Fig. 12. QoE v.s. bandwidth usage.

selected traces of HSDPA dataset. We replay the videos with all the evaluated ABR algorithms and get the resulted videos. For the resulted videos, we analyze them and calculate the VMAF, rebuffering time and switches. With these metrics and user QoE models, we calculate the QoE and show the normalized QoE (obtained with Ruyi as the standard) for two typical example users: User 30 and User 41 in Figure 13(a). User 30 is more sensitive to rebuffering events, while user 41 is more sensitive to high visual quality. Figure 13(a) shows that Ruyi performs the best. For User 30, Pensieve performs the worst due to the longest rebuffering time. For User 41, BB performs the worst due to the lowest VMAF value. This result shows that the ABR without specific QoE models or the ABR with an average QoE model is not sufficient and will not



(a) Normalized QoE.



(b) User ratings.

Fig. 13. Performance of Ruyi in Dash.js.

consistently perform well towards various user preferences. On average, Ruyi achieves 1.69 times the QoE of Pensieve over the two users (2.23 times on User30, 1.16 times on User41).

In addition to the QoE calculated with the preference-aware QoE models and the meta metrics, we let the corresponding users watch four videos out of the above resulted videos and rate in five scales. The four videos are generated with a same network trace and four evaluated ABR algorithms. The ratings are presented in Figure 13(b) which show the same trend as the calculated QoE.

VI. Related Work

Video QoE models. Video QoE models has two general categories: visual quality assessment (VQA) which focuses on pixel-level perception of users and QoE models considering streaming-related distortion and perception. VQA methods include traditional models (e.g. QP [33], PSNR [34], SSIM [35]) and data-driven models (e.g. VMAF [36], DeepVQA [37]). Besides, streaming-related QoE models for adaptive streaming videos attract many research efforts. The earliest ones start with only rebuffering being considered which are not accurate to model user QoE in adaptive streaming [38], [39]. After that, average bitrate is complemented to improve the prediction accuracy and the QoE is represented as the weighted sum of the average bitrate and the rebuffering where the trade-off weights should be determined [40], [41]. Motivated by observations that frequent quality switches degrade users' QoE [42], some models take quality switching into account [5], [6]. As bitrate is inadequate to model visual perception, many works suggest to replace the average bitrate by video quality assessment models [12], [13]. Thus, we use VMAF instead of bitrate to improve the prediction accuracy of the QoE model. In addition to the above models, machine learning are also used to model QoE [3], [4]. However, these learning models usually overfit the subjective opinion scores and the performance is affected by the limited users' scores.

Adaptive bitrate (ABR) algorithms. The earliest ABR algorithms can be primarily grouped into two classes: rate-based and buffer-based. Rate-based algorithms [15], [16] are hindered by the biases present when estimating available bandwidth on top of HTTP. Buffer-based approaches [6], [7] solely consider the client's playback buffer occupancy when deciding the bitrates. BB [7] leverages a simple but effective method to maintain the buffer occupancy at a safety range. After that, ABR algorithms which combine these two techniques [5], [9], [16] are proposed. MPC [5] employs model predictive control algorithms that use both throughput estimates and buffer occupancy information. However, MPC relies heavily on accurate throughput estimates which are not always available. Fugu [9] is proposed to improve the performance of bitrate adaptation which is based on MPC but replaces its throughput predictor with a deep neural network. A separate line of work is to leverage RL [43]–[46]. These schemes apply RL in a "tabular form", which stores the value function for all states and actions explicitly. Pensieve [8] leverages the deep RL method that represents the ABR algorithms as a neural network. This allows Pensieve to optimize its policy for different network characteristics and QoE metrics from experience. However, Pensieve is trained to optimize for a predefined QoE objective and can not adjust for different objectives online. MPC and Pensieve are objective-based methods, and a recent work combines BB and RL enabling the objective awareness [47]. We leave the extension of these works to support multi-objective as our future work.

VII. Discussion

Influence of content. Regardless of whether user preferences are considered, video content has a great impact on QoE of users. If the user is interested in the video content, or if the video content is enjoyable, then the user's tolerance for poor quality will be higher. A previous work does bitrate adaptation with the consideration of video content [48] for QoE models in additive form. Our work is orthogonal and complementary to this, allowing for more fine-grained user-specific bitrate adaptive control. Furthermore, other dimensions, such as the device, the energy, and the type of videos, considered in user preference-aware bitrate adaptation are left as future work.

Inapplicable situations. For the form of QoE models which are not additive with several preset meta metrics, such as end-to-end learning based QoE models, Ruyi is not applicable due to the design of directly predicting the impact of current actions on those metrics in the future. Instead, if the metrics that affect the users' experience are clear, Ruyi will perform well with these QoE models which occupy a large proportion and are widely used in modern ABR systems.

VIII. Conclusion

In this paper, we investigate the differences in user preferences. We conduct a user study with 90 subjects and find that the average user can not represent all users. Then we propose Ruyi, a video streaming system that incorporates preference awareness into both the QoE model and the ABR algorithm. Our simulation results show that, Ruyi increases QoE for all users. In testbed experiments, results show that Ruyi has the highest ratings from subjects.

IX. Acknowledgement

This work was supported by National Key R&D Program of China Grant (No. 2018YFB1802202) and NSFC (No. 6213000078 and No. 61872211).

References

[1] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan, "Mp-dash: Adaptive video streaming over preference-aware multipath," in *Proceedings of the 12th CoNEXT*, 2016, pp. 129–143.

[2] "Cisco annual internet report (2018–2023) white paper." https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[3] W. Robitza, M.-N. Garcia, and A. Raake, "A modular http adaptive streaming qoe model—candidate for itu-t p. 1203 ("p. nats")," in *2017 Ninth QoMEX*. IEEE, 2017, pp. 1–6.

[4] N. Eswara, S. Ashique, A. Panchbhai, S. Chakraborty, H. P. Sethuram, K. Kuchi, A. Kumar, and S. S. Channappayya, "Streaming video qoe modeling and prediction: A long short-term memory approach," *IEEE TCSVT*, vol. 30, no. 3, pp. 661–673, 2019.

[5] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM SIGCOMM*, 2015, pp. 325–338.

[6] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*. IEEE, 2016, pp. 1–9.

[7] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 187–198.

[8] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the 2017 ACM SIGCOMM*, 2017, pp. 197–210.

[9] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein, "Learning in situ: a randomized experiment in video streaming," in *NSDI 20*, 2020, pp. 495–511.

[10] Z. Duanmu, W. Liu, D. Chen, Z. Li, Z. Wang, Y. Wang, and W. Gao, "A knowledge-driven quality-of-experience model for adaptive streaming videos," *arXiv preprint arXiv:1911.07944*, 2019.

[11] Y. Zhu, A. Hanjalic, and J. A. Redi, "Qoe prediction for enriched assessment of individual video viewing experience," in *Proceedings of the 24th ACM MM*, 2016, pp. 801–810.

[12] A. Bentaleb, A. C. Begen, and R. Zimmermann, "Sdndash: Improving qoe of http adaptive streaming using software defined networking," in *Proceedings of the 24th ACM international conference on Multimedia*, 2016, pp. 1296–1305.

[13] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang, "A quality-of-experience index for streaming video," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 1, pp. 154–166, 2016.

[14] Y. Zhu, S. C. Guntuku, W. Lin, G. Ghinea, and J. A. Redi, "Measuring individual video qoe: A survey, and proposal for future directions using social media," *TOMM*, 2018.

[15] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 1, pp. 326–340, 2014.

[16] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 272–285.

[17] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[18] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," *Cognitive science*, vol. 16, no. 3, pp. 307–354, 1992.

[19] C. G. Bampis, Z. Li, I. Katsavounidis, T.-Y. Huang, C. Ekanadham, and A. C. Bovik, "Towards perceptually optimized end-to-end adaptive video streaming," *arXiv preprint arXiv:1808.03898*, 2018.

[20] Y. Wang, S. Inguva, and B. Adsumilli, "Youtube ugc dataset for video compression research," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2019, pp. 1–5.

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

[22] "tc: Linux advanced routing and traffic control." http://lartc.org/lartc.html.

[23] "Dash.js," https://github.com/ Dash-Industry-Forum/dash.js/wiki.

[24] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive http streaming," in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 169–174.

[25] J. W. Peirce, "Psychopy—psychophysics software in python," *Journal of neuroscience methods*, vol. 162, no. 1-2, pp. 8–13, 2007.

[26] J. Fowler, L. Cohen, and P. Jarvis, *Practical statistics for field biology*. John Wiley & Sons, 2013.

[27] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal value function approximators," in *ICML*, 2015, pp. 1312–1320.

[28] A. Dosovitskiy and V. Koltun, "Learning to act by predicting the future," *ICLR*, 2017.

[29] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *ICML*. PMLR, 2016, pp. 1995–2003.

[30] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th USENIX OSDI'16*, 2016, pp. 265–283.

[32] "Tflearn: Deep learning library featuring a higher-level api for tensorflow, 2017," http://tflearn.org/.

[33] H. ITU-T RECOMMENDATION, "264 "advanced video coding for generic audiovisual services"," 2003.

[34] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 2366–2369.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[36] "Video multimethod assessment fusion." https://github.com/Netflix/vmaf.

[37] W. Kim, J. Kim, S. Ahn, J. Kim, and S. Lee, "Deep video quality assessor: From spatio-temporal visual sensitivity to a convolutional neural aggregation network," in *ECCV*, 2018, pp. 219–234.

[38] K. Watanabe, J. Okamoto, and T. Kurita, "Objective video quality assessment method for evaluating effects of freeze distortion in arbitrary video scenes," in *Image Quality and System Performance IV*, vol. 6494. International Society for Optics and Photonics, 2007, p. 64940P.

[39] R. K. Mok, X. Luo, E. W. Chan, and R. K. Chang, "Qdash: a qoe-aware dash system," in *Proceedings of the 3rd MMSys*, 2012, pp. 11–22.

[40] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang, "A case for a coordinated internet video control plane," in *Proceedings of the ACM SIGCOMM 2012*, 2012, pp. 359–370.

[41] J. Xue, D.-Q. Zhang, H. Yu, and C. W. Chen, "Assessing quality of experience for adaptive http video streaming," in *2014 IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*. IEEE, 2014, pp. 1–6.

[42] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker effects in adaptive video streaming to handheld devices," in *Proceedings of the 19th ACM international conference on Multimedia*, 2011, pp. 463–472.

[43] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online learning adaptation strategy for dash clients," in *7th MMSys*. ACM, 2016, p. 8.

[44] M. Claeys, S. Latré, J. Famaey, T. Wu, V. Leekwijck, D. Turck *et al.*, "Design of a q-learning-based client quality selection algorithm for http adaptive video streaming," in *Proceedings of the 2013 Workshop on Adaptive and Learning Agents (ALA), Saint Paul (Minn.), USA*, 2013, pp. 30–37.

[45] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design and optimisation of a (fa) q-learning-based http adaptive streaming client," *Connection Science*, vol. 26, no. 1, pp. 25–43, 2014.

[46] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 131–138.

[47] T. Huang, C. Zhou, R.-X. Zhang, C. Wu, X. Yao, and L. Sun, "Stick: A harmonious fusion of buffer-based and learning-based approach for adaptive streaming," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 1967–1976.

[48] X. Zhang, Y. Ou, S. Sen, and J. Jiang, "Sensei: Aligning video streaming quality with dynamic user sensitivity," *NSDI*, 2021.