Personalized 360-Degree Video Streaming: A Meta-Learning Approach

Yiyun Lu
Shanghai Jiao Tong University
lyy_ji@sjtu.edu.cn

Yifei Zhu*
Shanghai Jiao Tong University
yifei.zhu@sjtu.edu.cn

Zhi Wang
Tsinghua University
wangzhi@sz.tsinghua.edu.cn

ABSTRACT

Over the past decades, 360-degree videos have attracted wide interest for the immersive experience they bring to viewers. The rising of high-resolution 360-degree videos greatly challenges the traditional video streaming systems in limited network environments. Given the limited bandwidth, tile-based video streaming with adaptive bitrate selection has been widely studied to improve the Quality of Experience (QoE) of viewers by tiling the video frames and allocating different bitrates for tiles inside and outside viewers' viewports. Existing solutions for viewport prediction and bitrate selection train general models without catering to the intrinsic need for personalization. In this paper, we present the first meta-learning-based personalized 360-degree video streaming framework. The commonality among viewers of different viewing patterns and QoE preferences is captured by efficient meta-network designs. Specifically, we design a meta-based long-short term memory model for viewport prediction and a meta-based reinforcement learning model for bitrate selection. Extensive experiments on real-world datasets demonstrate that our framework not only outperforms the state-of-the-art data-driven approaches in prediction accuracy and QoE improvement, but also quickly adapts to users with new preferences with significantly less training epochs.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Networks** → **Application layer protocols**.

KEYWORDS

360-degree video streaming, personalization, viewport prediction, bitrate selection, meta-learning

ACM Reference Format:
Yiyun Lu, Yifei Zhu, and Zhi Wang. 2022. Personalized 360-Degree Video Streaming: A Meta-Learning Approach. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3503161.3548047

*Corresponding author

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548047

1 INTRODUCTION

In recent years, with the development of Virtual Reality (VR) devices and the advancement of video capture technology, 360-degree videos have attracted a large number of viewers because of the immersive experience it brings. According to the statistics, the cumulative installed base of VR headsets worldwide is expected to surpass 34 million by 2024, increased by 106% from 2021 [18]. In 360-degree videos, a view in every direction is captured simultaneously when recording; Viewers can freely control the viewing direction when watching the video. Compared with traditional videos, 360-degree videos have higher requirements on video resolutions in order to ensure viewer experience. The rising of high-resolution 360-degree videos greatly challenges the traditional video streaming protocols in limited network environments.

Viewport is a specific area in the video frame that viewers usually focus on when watching a video. It is usually less than 20% of the full video scene [14]. Given the limited viewport that the viewers are focusing on, still transmitting a whole frame of video with the same quality brings little benefit to the viewers' experience and introduces a significant bandwidth wastage. Therefore, in recent years, tile-based video streaming framework for 360-degree video streaming is proposed and is widely being studied [2, 27]. In tile-based video streaming framework, each video frame is divided into a specific number of non-overlapping tiles. Tiles inside the viewport are usually allocated with a better video quality compared to tiles outside the viewport, so that the overall bandwidth can be utilized more effectively.

In the existing tile-based video streaming frameworks, viewport prediction and bitrate selection are two fundamental building blocks. Viewport prediction aims at predicting viewers' viewport in the next time period according to their head movements trajectories and video contents [4, 9, 10, 15]. Commonly used approaches include average [15], linear regression [9], machine learning [10], and recently deep learning approaches [11]. With the information of the predicted viewport, adaptive bitrate selection algorithms dynamically determine the bitrates for all the tiles in varying network conditions [11, 19, 24, 27]. Commonly used approaches include heuristic approaches, control-based approaches, and recently reinforcement learning-based approaches.

While researchers have made tremendous efforts in improving the Quality of Experience(QoE) of viewers by proposing new viewport prediction and bitrate selection algorithms, viewers' heterogeneous viewing patterns and diverse QoE requirements have not been fully captured and satisfied yet. Existing approaches mostly treat the entire population as a homogeneous entity and propose general models catering for the whole population, which may not be accurate for each individual in the real life. In practice, heterogeneity in viewing preferences and QoE requirements naturally

exists[7, 20]. For the first one, some people prefer to focus on a small area in the scene when viewing the 360-degree videos while others prefer to look around. For the latter one, some people may prefer high-quality videos, while others may prefer a smooth watching experience without rebuffering. The diverse requirements and preferences among viewers are hard to be captured by a single general model. Thus, a huge gap exists between the current approach of applying one task model for all viewers and the intrinsic need for personalization among viewers.

Although providing personalized 360-degree video streams is crucial to improving viewers' QoE, it is non-trivial to achieve. Essentially, in the tradeoff of generalization and personalization, relying on a general model to capture all types of viewers good-enough requires a tremendous amount of training data with rich statistical diversity. Training a separate model for each individual viewer also lacks sufficient data in practice and may suffer from over-fitting, leading to even worse performance. Grouping the similar viewers first and training models separately later seem to be plausible. However, native training of the independent distinct model for each group results in prohibitive training cost and calls for a complete retraining when a new type of viewers emerges. Conventional knowledge transfer techniques may lessen the burden for extra training data, but still rely on carefully selected pre-trained models.

In this paper, we propose a meta-learning-based 360-degree video streaming framework that can efficiently extract high-level knowledge among viewers, and serve personalized models for viewers to maximize their quality of experience simultaneously. Specifically, we apply the framework of meta-learning into two major building blocks of 360-degree video streaming, and propose a novel meta-based viewport prediction model and a meta-based bitrate selection model to offer personalized streaming services. For the viewport prediction, we leverage the state-of-the-art Long-Short Term Memory (LSTM) model as the building block. We design a meta-LSTM cell to extract common knowledge and predict parameters for the basic-LSTM cells specified for various tasks. For the bitrate selection, we evolve the state-of-the-art reinforcement learning-based bitrate approaches with a meta-LSTM model so that the heterogeneous QoE requirements can be captured. To the best of our knowledge, this is the first work that applies the meta-learning framework to offer personalized 360-degree video streaming. In summary, the contributions of our paper are summarized as follows:

- We propose a novel meta-learning-based 360-degree video streaming framework that provides personalized video streaming for viewers with diverse viewing preferences and QoE requirements.
- We design a novel meta-network that contains two carefully designed LSTM cells to extract common knowledge of viewers and predict viewers' viewport according to their viewport preferences.
- We propose a meta-based deep reinforcement learning algorithm that adaptively selects bitrates for tiles to maximize QoE objectives under various QoE preferences of viewers.
- Extensive experiments on real-world datasets demonstrate that our framework outperforms the state-of-the-art data-driven approaches in prediction accuracy by 11% on average and improves QoE by 27% on average, and significantly reduces the required epochs by 67%-88% on average when meeting new viewer preferences.

2 RELATED WORK

In this section, we review the related work in viewport prediction and bitrate selection. The viewport-assisted tile-based 360-degree video streaming tiles the video chunk and decides the bitrate for each tile within and outside the viewport given the viewport information. Extra reliance on the viewport information makes the algorithms heavily affected by the accuracy of the viewport predictor. Van der Hooft et al. [19] propose a heuristic-based approach that gradually increases the bitrate for tiles inside the viewport region until the bandwidth is not available. Xie et al. [24] allocate the rates for tiles in the viewport according to a probabilistic optimization model. A number of recent studies start to use reinforcement learning (RL) with different features to dynamically decide bitrate for tiles inside the viewport, like DRL360 [27], SRL360 [2], and Plato [11]. These methods prove to outperform the previous heuristics attempts, probabilistic models as well as other deterministic models.

While most of the works focus on general models for all viewer population, several papers have also started to explore viewers' personalized behavior and propose preliminary personalized viewport prediction approaches. Ban et al. [1] cluster viewers with the K-nearest neighbors algorithm and use a separate linear regression model for each viewer cluster. Wang et al. [22] advance this by training an LSTM for each viewer cluster. These personalization-aware methods demonstrate significant improvements in viewport prediction performance. However, they still need to build a distinct model for each of the viewer group with little parameter sharing. A complete retraining is also required when facing new viewer groups, slowing the serving speed. Another line of research resorts to knowledge transfer to handle the training data deficiency in viewport prediction. However, the conventional knowledge transfer techniques rely on carefully selected pre-trained models or representative domains to avoid negative transfer [13][28][25]. Zhang et al. [26] adopt a personalized federated learning framework for viewport prediction so that the need for data privacy and personalization can both be satisfied. Our work advances the study of personalized 360-degree video streaming and innovates both the viewport prediction model and the bitrate selection model by proposing a novel mata-learning-based solution. Our proposed models can automatically learn the common knowledge from diverse viewers and help the streaming systems with better prediction and QoE performances and a faster adaptation speed.

3 SYSTEM MODEL AND PROBLEM FORMULATION

In this section, we introduce the framework of our personalized 360-degree video streaming system, model the key components in our system, and formally formulate the studied problem.

3.1 Personalized 360-degree Video Streaming

Our 360-degree video streaming framework is illustrated in Fig. 1. We have a tile-based 360-degree video streaming system between

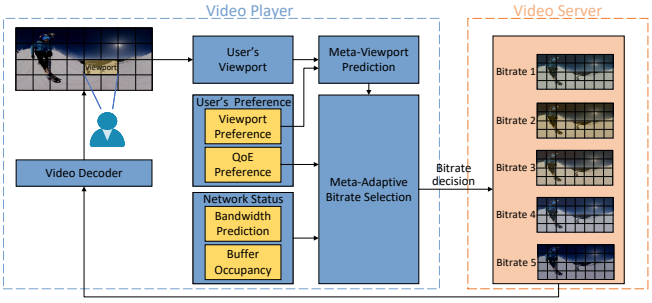


Figure 1: The meta-learning-based personalized 360-degree video streaming framework

the video player and the server. Viewers possess different preferences in viewing patterns and QoE requirements. The video is divided into C segments with the same time duration, named *chunks*. Each video chunk is divided into $m \times n$ tiles, where m is the number of rows and n is the number of columns. We define $\mathcal{I} = \{1, 2, ..., m\}$ as the candidates of the row index and $\mathcal{J} = \{1, 2, ..., n\}$ as the candidates of the column index. After downloading the video chunk from the server, the viewer interacts with the video through a head-mounted display device and the viewport is recorded. The viewport prediction model predicts the viewer viewport for the next video chunk corresponding to the viewer's viewport preference. For the c-th chunk, the indicator variable $v_{ij}^c \in \{0, 1\}$ is 1 if the tile at location (i, j) is in the viewport, and is 0 otherwise. With the viewport information, the adaptive bitrate selection model decides the bitrate for tiles inside and outside the predicted viewport for the next video chunk corresponding to the viewer's QoE preference. For the c-th chunk, $r_{ij}^c \in \mathcal{R}$ is the bitrate for the tile at location (i, j) where $\mathcal{R}$ is the candidate bitrate set. The client then downloads the next video chunk from the server with the selected bitrate.

3.2 Quality of Experience Model

We first model the QoE of viewers incorporating three commonly used critical factors, average viewport quality, rebuffering time, and average viewport quality variation. To be specific, for the c-th chunk, these three factors are defined as follows.

Average Viewport Quality. Average Viewport Quality, $QoE_{1,c}$, is defined as the average bitrate of the tiles inside the viewport, namely,

$$QoE_{1,c} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^c r_{ij}^c}{\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^c}. \tag{1}$$

Rebuffering Time. When players download the chunk, if the buffer was used up before the current chunk is downloaded, it would cause rebuffering. Assume that D_c is the downloading time of the c-th chunk; s_{ij} is the storage size of the tile at location (i, j); N_c is the average bandwidth when downloading the c-th chunk. We can first calculate D_c as:

$$D_c = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} s_{ij}^c}{N_c}. \tag{2}$$

With the downloading time, we can describe the relationship between the start time for downloading two consecutive video chunks t_{c-1} and t_c as:

$$t_c = t_{c-1} + D_{c-1}. \tag{3}$$

We denote the buffer occupancy before downloading the c-th chunk as $B_c \in [0, B_{max}]$. The buffer has B_{max} as its maximum size. If the buffer is used up before the current chunk is downloaded, it would cause rebuffering. B_c thus is calculated as:

$$B_c = (B_{c-1} - D_{c-1})_+ + T_c, \tag{4}$$

where T_c is the duration of the c-th chunk; Function $(x)_+ = max\{x, 0\}$ makes the term non-negative.

The rebuffering time, $QoE_{2,c}$, for c-th chunk thus is defined as the value of subtracting the buffer occupancy before downloading the chunk from this chunk's downloading time. When the result is negative, it means that there is no rebuffering so the rebuffering time is zero. Formally, it can be presented as

$$QoE_{2,c} = (D_c - B_c)_+. \tag{5}$$

Average Viewport Quality Variation. Average Viewport Quality Variation, $QoE_{3,c}$, captures the variation between the average viewport quality of the current chunk and that of the previous chunk. It is calculated as:

$$QoE_{3,c} = |\frac{\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^c r_{ij}^c}{\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^c} - \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^{c-1} r_{ij}^{c-1}}{\sum_{i=1}^{m}\sum_{j=1}^{n} v_{ij}^{c-1}}|. \tag{6}$$

In summary, the QoE at the c-th chunk can be modeled as:

$$QoE_c = \alpha_1 QoE_{1,c} - \alpha_2 QoE_{2,c} - \alpha_3 QoE_{3,c}, \tag{7}$$

where $\alpha_1, \alpha_2, \alpha_3$ are non-negative weights for each component of the QoE metrics. Similar models have also been used in [2, 27].

3.3 Preference Model

We then model the viewers' personal preferences in both viewport and QoE in the 360-degree video streaming. In this paper, we assume that viewers with different viewing patterns and QoE requirements can be clustered into different groups as also being assumed in [1, 22]. Practically, building personalized models for such a system is also more feasible and reliable due to the limits in individual data size and the avoidance of over-fitting.

Viewport Preference. Viewport preference V_u indicates a group of viewers sharing the similar viewing behavior. For example, some groups of viewers prefer to watch dynamic objects while others prefer to focus on static objects. Assume there are N viewers in total. N_v is the number of videos; C_n is the chunk numbers of the n-th video; $E(c)$ indicates whether the two viewers focus on the same viewport at the c-th video chunk of the same video:

$$E(c) = \begin{cases} 1, & \text{if } u_a(c) = u_b(c), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

We define a straightforward similarity score function to measure the similarity between two viewers' viewports. It also allows us to retrieve viewport preference. The similarity score between two viewers, u_a and u_b, is calculated as:

$$S(u_a, u_b) = \frac{\sum_{n=1}^{N_v}\sum_{c=1}^{C_n} E(c)}{N_v}. \tag{9}$$

For viewer u_a, we thus have the feature vector, $[S(u_a, u_1)...S(u_a, u_N)]$. Based on the constructed feature vectors, we can cluster viewers into different groups, where each group represents viewers with a

specific type of viewing pattern. Two similar viewers have high similarity not only in their reciprocal viewing pattern (i.e. displaying the same viewport) but also in their similar distance to the rest of the users. Technically, to achieve so, we calculate the similarity between two viewers based on their distance in the high-dimensional feature space, which reflects each user's viewport commonality with respect to all the other users, including its pair.

QoE Preference. Different viewers have different preferences on QoE[7, 20]. For example, some viewers prefer to watch the video with the highest quality, even though the video is slightly stalled in the middle; some viewers cannot tolerate the rebuffering time and they would rather watch a fluent video of lower quality while others require that the video quality does not change frequently in the watching experience. Therefore, we categorize viewers into different groups with each group represented by a different setting of $\alpha_u = [\alpha_1, \alpha_2, \alpha_3]$ in Equation 7. This indicates different types of viewers' preference on average viewport quality, rebuffering time, or average viewport quality variation. Viewers in the same group follow the same QoE preference setting.

3.4 Problem Formulation

For a given viewer with viewport preference V_u, α_u, we target at maximizing its overall QoE over all video chunks by deciding the bitrate for each tile. Formally, our personalized QoE maximization problem is formulated as follows:

$$\max_{\{r_{ij}^c\}_{\mathcal{I}\times\mathcal{J}}} \sum_{c=1}^{C} QoE_c \tag{10}$$

$$\text{s.t. } t_c = t_{c-1} + D_{c-1} \tag{11}$$

$$B_c = (B_{c-1} - D_{c-1})_+ + T_c \tag{12}$$

$$r_{ij} \in \mathcal{R} \tag{13}$$

Constraint (11) captures the throughput dynamic; Constraint (12) captures the buffer dynamic; Constraint (13) describes the discrete range of the bitrate.

4 PRELIMINARIES ON META-LEARNING

In this section, we introduce the general meta-learning framework to prepare for our design of meta-based viewport prediction and bitrate selection models.

Meta-learning, commonly understood as "learning-to-learn", provides a framework that learns the high-level knowledge that is task-agnostic, and uses this experience to improve the task-specific learning performance. The extracted knowledge also helps the model to adapt quickly and accurately when it meets a new task. The process of meta-learning can be concluded into two steps. During the base-learning step, an inner learning algorithm solves a specific task with an inner objective such as minimizing classification loss or maximizing the reward function. During the meta-learning step, an outer learning algorithm learns through the episodes of base tasks and updates the inner learning algorithm with an outer objective such as the general performance of the inner algorithm.

In a meta-learning framework, we have datasets of M tasks that are used for training, denoted as $\mathcal{D} = \{\mathcal{D}^{train(i)}, \mathcal{D}^{val(i)}\}_{i=1}^{M}$. We define $\mathcal{L}^{meta}$ as the objective of the outer algorithm, $\mathcal{L}^{task}$ as the objective of the inner algorithm, and θ as the task-specific model parameters. We want to train the meta-knowledge ω that can generate $\theta^{*(i)}$ which performs well on the validation sets for a specific task i after the training. Formally, the training process of meta-learning can be formulated as follows:

$$\omega^* = \arg\min_{\omega} \sum_{i=1}^{M} \mathcal{L}^{meta}(\theta^{*(i)}(\omega), \omega, \mathcal{D}^{val(i)}) \tag{14}$$

$$\text{s.t. } \theta^{*(i)}(\omega) = \arg\min_{\theta} \mathcal{L}^{task}(\theta, \omega, \mathcal{D}^{train(i)}), \tag{15}$$

where (14) represents an outer learning algorithm and (15) represents an inner learning algorithm.

There are two widely used meta-learning methods. One is the optimization-based method, where the inner task (15) is literally solved as an optimization problem. The representative of this method is MAML [6]. Another one is the model-based method. In the latter one, rather than explicitly optimize the objectives as in (14) and (15), the learning step is wrapped up in a single model with the feed-forward manner, illustrated as follows:

$$\omega^* = \arg\min_{\omega} \sum_{i=1}^{M} \sum_{(x,y)\in\mathcal{D}^{val(i)}} [(x^T g_\omega(\mathcal{D}^{train(i)}) - y)^2], \tag{16}$$

where g_ω is the model that embeds the training set $\mathcal{D}^{train}$ to a weight vector and the model should performs well in tasks drawn from the task set $\mathcal{T}$. The typical architectures of the models are recurrent networks [5, 21] and hypernetworks [3, 8]. Compared to the optimization-based methods, the model-based methods have simpler optimization without requiring second-order gradients. Therefore, we also adopt the model-based meta-learning framework.

5 META VIEWPORT PREDICTION

In this section, following the framework, we present our proposed meta-learning-based LSTM model as the viewport prediction model. Network architectures are carefully designed to facilitate successful meta knowledge extraction from various types of viewers.

In the personalized viewport prediction scenario, we define a task as predicting viewport for viewers with the same viewport preference, i.e. in the same cluster. The flow of the meta-network is illustrated in Fig. 2. We assign a basic-LSTM cell to each task, while sharing a meta-LSTM cell network among tasks. To simplify the diagram, we only draw the basic-LSTM cell for task k given inputs from task k. At timestep t, given the hidden state of the meta-LSTM cell, $\hat{h}_t$, hidden state of the basic-LSTM cell for task k, h_t^k, and the viewport data for task k, x_t^k, as inputs, the meta-LSTM cell generates weight z_{t+1}^k for the basic-LSTM cell. The output of the basic-LSTM cell h_{t+1}^k is further fed into a fully-connected (FC) layer to match the dimension of the input. The output $\hat{h}_{t+1}$, h_{t+1}^k, x_{t+1}^k are forwarded as the inputs at the next timestep. This can be concluded as follows:

$$[\hat{h}_{t+1}, z_{t+1}^k] = \text{Meta-LSTMCell}(x_t^k, \hat{h}_t, h_t^k; \theta_m), \tag{17}$$

$$h_{t+1}^k = \text{Basic-LSTMCell}(x_t^k, h_t^k; z_{t+1}^k, \theta_k) \tag{18}$$

$$x_{t+1}^k = \text{FC}(h_t^{k+1}), \tag{19}$$

where θ_m, θ_k are parameters for the meta-LSTM model and the basic LSTM model for task k, respectively. The meta-network captures can learn the meta knowledge of different tasks to predict parameters for the basic task-specific network. In the following parts, we explain the update of the basic-LSTM cell and the meta-LSTM cell in detail.

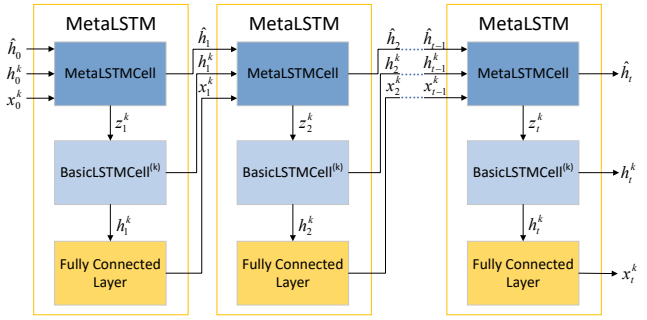


Figure 2: The structure of our meta-LSTM model for viewport prediction

Basic-LSTM cell. For each specific task k, we use a basic LSTM for encoding the input. We denote $W(z_t^k)$ and $b(z_t^k)$ as dynamic parameters controlled by z_t^k generated by Meta-LSTM. The basic LSTM for task k can be specified as:

$$\begin{bmatrix} g_t^k \\ o_t^k \\ i_t^k \\ f_t^k \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W(z_t) \begin{bmatrix} x_t \\ h_{t-1}^k \end{bmatrix} + b(z_t^k) \right) \tag{20}$$

$$c_t^k = g_t^k \odot i_t^k + c_{t-1}^k \odot f_t^k \tag{21}$$

$$h_t^k = o_t^k \odot \tanh(c_t^k), \tag{22}$$

where i_t is the input gate, f_t is the forget gate, o_t is the output gate, c_t is the memory cell state, σ represents the logistic sigmoid function and $\odot$ represents element-wise multiplication. It is worth noting that, different from the traditional LSTM model, where the parameters are independent of other networks, in our meta network, the parameters of the basic-LSTM cell is controlled by a meta vector generated by the meta-LSTM cell.

Meta-LSTM cell. The Meta-LSTM cell depends on the input x_t and the previous hidden state h_{t-1}^k of the basic LSTM cell and its own hidden state to generate parameters z_t^k for the basic LSTM for task k. The Meta-LSTM cell can be specified as:

$$\begin{bmatrix} \hat{g}_t \\ \hat{o}_t \\ \hat{i}_t \\ \hat{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W_m \begin{bmatrix} x_t \\ \hat{h}_{t-1} \\ h_{t-1}^k \end{bmatrix} + b_m \right) \tag{23}$$

$$\hat{c}_t = \hat{g}_t \odot \hat{i}_t + \hat{c}_{t-1} \odot \hat{f}_t \tag{24}$$

$$\hat{h}_t = \hat{o}_t \odot \tanh(\hat{c}_t) \tag{25}$$

$$z_t^k = W_z \hat{h}_t, \tag{26}$$

where W_m, b_m, W_z are affine transformation parameters. Different from the traditional LSTM cell, our meta-LSTM cell takes both the hidden state for the meta-LSTM cell and the basic-LSTM cell as the input and outputs the extra weight matrix for the basic-LSTM cell.

The training and adaptation process for the meta viewport prediction model can be concluded in Alg. 1. In the meta-training process, tasks are sampled from the task set. For each task k, the hidden state of both the meta-LSTM cell and the basic-LSTM cell for k is initialized. In each timestep t, the viewport for $t+1$ is predicted. Then the parameters of the model are updated to minimize the prediction loss. The output $\hat{h}_{t+1}$, h_{t+1}^k, x_{t+1}^k is fed as the input of $t+1$. The meta-adaptation process is similar to the meta-training process except that the model is trained with a new task.

Algorithm 1 Meta-Based Viewport Prediction

Input: Learning rate α; Task number N; Maximum training timesteps T_{max}, Dataset $\mathcal{D}$; Task set $\mathcal{T}$; New task T_{new}

Meta-training

Initialize $\theta_{meta-viewport}$
for each epoch **do**
  Sample N tasks from $\mathcal{T}$
  for $k = 1, ..., N$ **do**
    Initialize hidden state $\hat{h}_0, h_0^k$, input x_0^k
    for $t = 1, ..., T_{max}$ **do**
      Retrive input $\hat{h}_t, h_t^k, x_t^k$ for the current timestep from timestep $t-1$
      Calculate output $\hat{h}_{t+1}$, h_{t+1}^k, x_{t+1}^k as the input for timestep $t+1$ through (17)-(19)
      Retrieve the prediction result from x_{t+1}^k
      Update network parameter $\theta_{meta-viewport}$
    end for
  end for
end for

Meta-adaptation

for each epoch **do**
  Select T_{new} as the task
  Initialize hidden state $\hat{h}_0, h_0^{new}$, input x_0^{new}
  for $t = 1, ..., T_{max}$ **do**
    Retrive input $\hat{h}_t, h_t^{new}, x_t^{new}$ for the current timestep from timestep $t-1$
    Calculate output $\hat{h}_{t+1}, h_{t+1}^{new}, x_{t+1}^{new}$ as the input for timestep $t+1$ through (17)-(19)
    Retrieve the prediction result from x_{t+1}^{new}
    Update network parameter $\theta_{meta-viewport}$
  end for
end for
return $\theta_{meta-viewport}$

6 META ADAPTIVE BITRATE SELECTION

For a typical deep reinforcement learning process, after the state space, action space and a QoE objective are defined, the agent learns the optimized policy by exploring the environment so that it can select the most appropriate bitrate at a certain state. However, as we have illustrated, different viewers can have different preferences

on QoE. classical DRL is not appropriate in our scenario because it is trained for one QoE objective only. When encountering new QoE objectives, the previous learning parameters are no longer useful under the new objective, thus the model must be retrained. To avoid unnecessary retraining and quick adaptation, we design a meta-learning-based DRL model in this part.

Our model consists of one LSTM model as the meta-network, another LSTM model for bandwidth prediction, and an actor-critic network as the policy learner. The overall structure is illustrated in Fig. 3. The configuration of our meta-DRL model is very similar to a traditional DRL model, except that the history including last reward r_{t-1} and the last action a_{t-1} are also incorporated as the input into the model in addition to the current state s_t. Then these features are fed forward to an extra LSTM network compared to the traditional RL model. The LSTM's hidden states can serve as a memory to internalize the dynamics and track characteristics between rewards, states, and actions in different tasks as we feed the history into it. Thus, our model is able to adapt and find a strategy to optimize the reward for a new task quickly with the learned meta knowledge through LSTM.

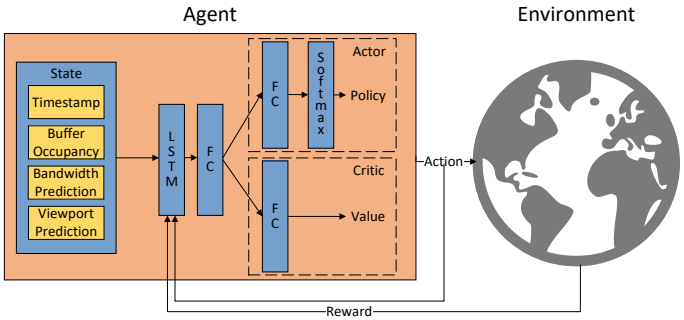


Figure 3: The structure of our meta-DRL model for adaptive bitrate selection

We formulate adaptive bitrate selection as a Markov Decision Process which calls for the design of state, action, and reward. We present our design for each of them as follows.

State. The state after downloading chunk c is represented as follow:

$$s_c = (t_c, B_c, N_{t_{c+1}+1}, ..., N_{t_{c+1}+\tau}, V_{c+1}),$$

where $N_{t_{c+1}+1}, ..., N_{t_{c+1}+\tau}$ is the predicted bandwidths in the next τ seconds, V_{c+1} is the predicted viewport for the next chunk.

Action. The model determines the bitrate for tiles inside and outside the viewport for the next chunk. The action is represented as $a_c = \{a_{c,out}, a_{c,in}\}$ where $a_{c,out}, a_{c,in}$ are the selected bitrates for tiles outside viewport and tiles inside viewport.

Reward. The reward for the c-th chunk is calculated as $r_c = QoE_c$ defined in (7).

Network architecture. The agent take actions based on the policy $\pi_{\theta_a} : (s_c, a_c) \rightarrow [0, 1]$, representing the probability of taking an action a_c at state s_c. θ_a are the parameters of the policy network. We use the advantage actor critic method [12] as our training algorithm. Given θ_a, the gradient of the accumulated discounted reward

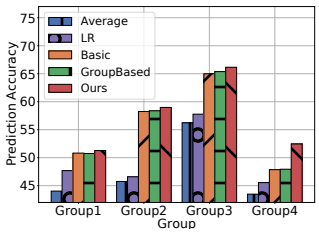


Figure 4: Viewport prediction accuracy comparison

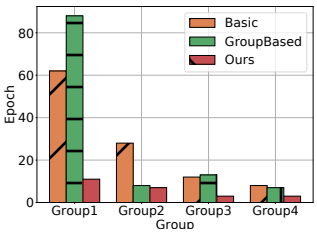


Figure 5: Number of epochs for fast adaptation

can be calculated as:

$$\nabla_{\theta_a} \mathbf{E}_{\pi_{\theta_a}}[\sum_{c=0}^{n} \gamma^c r_c] = \mathbf{E}_{\pi_{\theta_a}}[\nabla_{\theta_a} log_{\pi_{\theta_a}}(s, a) A^{\pi_{\theta_a}}(s, a)], \quad (27)$$

where $\gamma \in (0, 1]$ is the discounting factor on future rewards, $A^{\pi_{\theta_a}}(s, a)$ is the advantage function indicating how much advantage the action a selected has over other actions at state s, calculated as:

$$A^{\pi_{\theta_a}}(s_c, a_c) = r_c + \gamma V_{\theta_v}^{\pi_{\theta_a}}(s_{c+1}) - V_{\theta_v}^{\pi_{\theta_a}}(s_c), \quad (28)$$

where $V_{\theta_v}^{\pi_{\theta_a}}(\cdot)$ is the output of the critic network.

We add an entropy of policy $H(\cdot)$ used in [12] in the update of the actor network to discourage converging to a sub-optimal policy. The actor network parameters can be updated as below:

$$\theta_a = \theta_a + \eta_a \sum_c \nabla_{\theta_a} log_{\pi_{\theta_a}}(s_c, a_c) A^{\pi_{\theta_a}}(s_c, a_c) + \delta \nabla_{\theta_a} H^{\pi_{\theta_a}}(s_c), \quad (29)$$

where η_a is the learning rate for the actor network and δ is the entropy coefficient. Following the temporal difference method, the parameters of the critic network can be updated as:

$$\theta_v = \theta_v - \eta_v \sum_c \nabla_{\theta_v} [A^{\pi_{\theta_a}}(s_c, a_c)]^2, \quad (30)$$

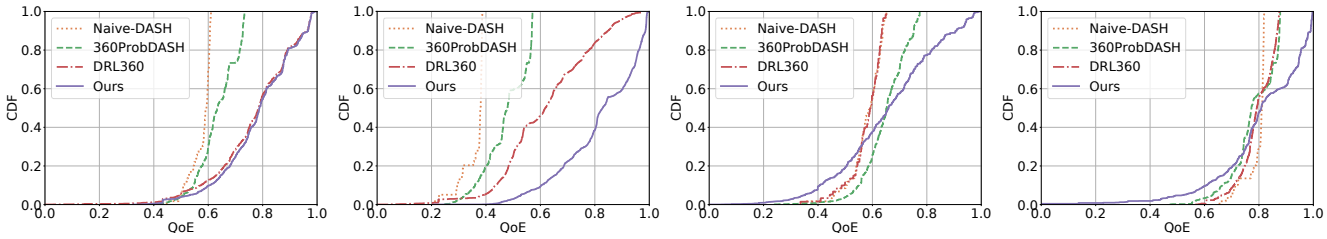
where η_v is the learning rate for the critic network.

The training and adaptation process can be concluded in Alg. 2. The bandwidth prediction model θ_{band} is first trained on the bandwidth traces. In the meta-training process, tasks are sampled from the task set. For each task, the hidden state of the LSTM network is reset at first. Then the model is trained with this task for an episode. In each episode step, the agent interacts with the environment according to the policy and gets the reward. The parameters of the model are updated according to (29) and (30) to maximize the sum of rewards over all steps. The meta-adaptation process is similar to the meta-training process except that the model is trained with a new task.

7 PERFORMANCE EVALUATION

7.1 Experiment Setup

Dataset. For videos and viewport trajectories, we have eight 360-degree 30-FPS videos with an average length of 164 seconds from the dataset [23]. It contains viewport trajectories from 48 viewers collected on the head-mounted display devices. We split the video into 1-second chunks. For each chunk, we divide the video segment into 8×8 tiles. We randomly select one video for validation, one for testing and the rest for training. We set five candidate rates: 1Mbps



(a) (1,1,1) (b) (1,0.25,0.25) (c) (1,4,1) (d) (1,1,4)

Figure 6: The CDFs of the normalized QoE under four QoE preferences.

Algorithm 2 Meta-Based Actor-Critic Algorithm

Input: Learning rate α; Task number N; Maximum episode step C_{max}; Dataset $\mathcal{D}$; Task set $\mathcal{T}$; Viewport prediction network parameter $\theta_{meta-viewport}$; Bandwidth prediction network parameter θ_{band}; New task T_{new}

Train θ_{band} on $\mathcal{D}$
Meta-training
Initialize network parameter $\theta_{meta-ac}$
for each epoch **do**
  Sample N tasks from $\mathcal{T}$
  for $k = 1, ..., N$ **do**
    Initialize hidden state h_k
    for $c = 1, ..., C_{max}$ **do**
      Predict $v'(c+1), N_{t_{c+1}+1}, ..., N_{t_{c+1}+\tau}$
      Retrieve $s_c, t_c, a_{c-1}, r_{c-1}$
      Sample a_c from π_θ, and calculate r_c
      Update network parameter $\theta_{meta-ac}$
    end for
  end for
end for
Meta-adaptation
for each epoch **do**
  Select T_{new} as the task
  Initialize hidden state h_{new}
  for $c = 1, ..., C_{max}$ **do**
    Predict $v'(c+1), N_{t_{c+1}+1}, ..., N_{t_{c+1}+\tau}$
    Retrieve $s_c, t_c, a_{c-1}, r_{c-1}$
    Sample a_c from π_θ, and calculate r_c
    Update network parameter $\theta_{meta-ac}$
  end for
end for
return $\theta_{meta-ac}$

(360p), 5Mbps (720p), 8Mbps (1080p), 16Mbps (2K) and 35Mbps (4K). Each video is encoded by FFMPEG with encoder X.264. For bandwidth dataset, we select 40 bandwidth traces from the dataset[16] with various fluctuation patterns.

Viewport Dataset Pre-processing. To identify viewers with different viewing patterns for simulation, we first use the method suggested in [4] to transform the unit quaternions to equirectangular viewport. We then calculate the feature vector following (9) and cluster viewers into different groups using the Euclidean distance as the distance measure and the Ward's method as the linkage criterion. The elbow method is used to determine the number of clusters. As a result, we preprocess viewers into four groups and thus split the original dataset into 4 sub-datasets for each viewer group. For each sub-dataset, we select one video for testing, one video for validation, and the remaining six videos for training.

Benchmarks. For the viewport prediction, we compare ours with four other approaches:

- Average [15], which predicts the viewport as the average viewport in the last two seconds.
- Linear Regression (LR) [24], which uses an LR model to prediction the viewport using data in the last two seconds.
- Basic LSTM (Basic), which is a commonly used neural network model to predict the viewport in the future in multiple works e.g., DRL360 [27].
- Viewer Group Based Viewpoint Recommendation (Group-based), which groups viewers first and trains a separate LSTM for each viewer group [22].

For the adaptive bitrate selection, we compare ours with three other related approaches from the literature:

- Linear rate prediction without viewport awareness (Naive-DASH) [17], which does not use tiling and selects the same bitrate for the entire chunk. It assigns each chunk with the maximum bitrate under the predicted bandwidth.
- 360ProbDASH [24], which uses LR as the viewport prediction model and introduces a control-based method with a buffer constraint to prevent rebuffering in bitrate selection.
- DRL360 [27], which applies a deep reinforcement learning method with the actor-critic architecture to decide the bitrate inside the viewport.

Metrics. For viewport prediction, we examine the prediction accuracy of all methods on each viewer group. We also compare the number of epochs to reach an accuracy threshold value for a new viewer group. We also compare the number of epochs to reach a QoE threshold value on a new QoE preference. Note that the Average, LR methods in viewport prediction, and Naive-DASH and 360ProbDASH in bitrate selection are excluded from the adaptation test because they are not neural network-based algorithms.

Parameter Settings. For the meta viewport prediction model, the hidden size for the basic-LSTM cell h is 8; The hidden size for the meta-LSTM cell $\hat{h}$ is 16; The embedding size z is 16; The number of LSTM layers is 2; The learning rate during meta-training process

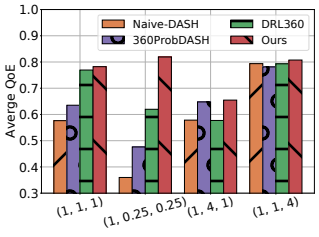


Figure 7: Average normalized QoE comparison

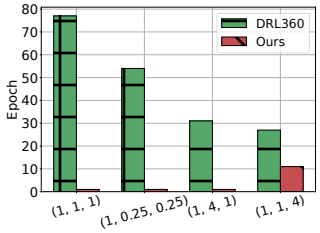


Figure 8: Number of epochs for fast adaptation

is 0.01; The learning rate during the meta-adaptation process is 0.005. For the meta DRL bitrate adaptation model, the prediction duration of bandwidth τ is 2; The maximum buffer occupancy is 4; The hidden size of the LSTM network is 48; The number of the layer of the LSTM network is 1; All the FC layers have the size of 128; The learning rate for the actor network η_a in both meta-training process and meta-adaptation process is 0.0001. The learning rate for the critic network η_v is set the same as η_a. The entropy coefficient δ is 0.01; The discount factor γ is 1.

QoE Objectives. We define four QoE preferences in our experiments, which are (1, 1, 1), (1, 0.25, 0.25), (1, 4, 1), (1, 1, 4) that indicates viewers' preference on a balanced watching experience, high video quality, low rebuffering time, and low video quality various between consecutive video chunks, respectively.

7.2 Evaluation Results

Viewport Prediction. Fig. 4 presents the viewport prediction accuracy for all the testing groups. As can be seen, our method significantly improves the prediction accuracy over the Average method and the LR method by up to 29% and 27%, respectively. In addition, our meta-learning-based LSTM model also outperforms the basic LSTM approach and the group-based approach by 1%, 1%, 1%, 10% on these groups, respectively, which indicates the effectiveness of the meta knowledge. To further explain the reasons that group 3 has the highest accuracy across all groups, we measure how often one viewer changes its viewport in the total time span. We find that the average viewport change frequency of group3 is significantly lower than that of other groups (0.28 as opposed to 0.40-0.42). This indicates that users in group3 have a stable viewport pattern, which leads to overall good performance.

We then examine the number of epochs required for the training-based methods to reach a certain threshold when a new viewer group emerges in Fig. 5. The threshold is set one percentage below the lowest viewport prediction accuracy of the three methods. Compared to basic LSTM, our method reduces 82%, 82%, 75%, 63% of adaptation epochs on setting group1, group2, group3, and group4, as the new group, respectively. Though the basic LSTM is trained with dataset with different tasks, it does not extract the shared knowledge as our method. Thus, when it encounters a new task, it still cannot quickly adapt to predict accurately. Compared to the group-based method, our method reduces 88%, 13%, 77%, and 57% adaptation epochs in these groups respectively. Since the group-based method trains an LSTM for each task, a separate model needs to be trained from zero for the new task, leading to a slow adaptation process compared to our approach. Experiment results show that the meta knowledge helps our method not only outperform other methods in predicting viewports for the known tasks, but also it can reduce considerable adaptation epochs to a new group of viewers.

Adaptive Bitrate Selection. We next examine the performance of our approach in bitrate selection. Fig. 6 illustrates the CDFs of the normalized QoE under four QoE preferences. The QoE of Naive-DASH and 360ProbDASH is concentrated in one small range. In contrast, a large proportion of our model's QoE is concentrated in the range of larger values. Except the similar performance of our model with DRL360 in the first viewer group, our model keeps generating higher QoE values for a larger proportion of viewers. According to the average normalized QoE shown in Fig. 7, our method performs up to 128%, 72%, 32% better than Naive-DASH, 360ProbDASH, DRL360, respectively, across all groups. The results demonstrate that benefited from the learned meta knowledge, our meta-DRL model also improves the QoE on various QoE objectives over the traditional DRL model.

We next evaluate the number of epochs required for extra training to reach a certain threshold when facing a new QoE group. The threshold is set five percentage below the lowest QoE of all methods. Each time, we pick a QoE preference group as the new task in the meta-adaptation process and the remaining three as the training tasks in the meta-training process. According to the result shown in Fig. 8, compared to DRL360, our method reduces 99%, 98%, 97%, 59% adaptation epochs on QoE when meeting preference (1, 1, 1), (1, 0.25, 0.25), (1, 4, 1), (1, 1, 4) as new, respectively. While the traditional DRL model struggles in finding the optimal strategy when it encounters a new QoE objective, our model quickly optimizes the strategy in only a few epochs.

8 CONCLUSION

Existing solutions for tile-based video streaming with adaptive bitrate selection in 360-degree video streaming cannot capture the heterogeneous preferences among viewers in viewing patterns and QoE objectives. In this paper, we present the first work to incorporate meta-learning into personalized 360-degree video streaming services. Our proposed meta-viewport prediction and meta-bitrate selection models can efficiently extract the common knowledge from diverse viewers and quickly adapt to new viewer preferences using the learned meta-knowledge. Extensive experiments demonstrate that, compared with the state-of-the-art data driven approaches, our personalized 360-degree video streaming framework can significantly improve the viewport prediction accuracy by 11% on average and reduces 67% of training epochs when meeting new viewers; it can also improve the QoE by 27% on average and reduces 88% of training epochs when models adapt to viewers with new preferences.

ACKNOWLEDGMENT

Yifei Zhu's work is supported by SJTU Explore-X grant. The work of Zhi Wang is supported by NSFC 61872215, Shenzhen Science and Technology Program RCYX20200714114523079.

REFERENCES

[1] Yixuan Ban, Lan Xie, Zhimin Xu, Xinggong Zhang, Zongming Guo, and Yue Wang. 2018. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *Proc. IEEE ICME*. 1–6.

[2] Jiawen Chen, Miao Hu, Zhenxiao Luo, Zelong Wang, and Di Wu. 2020. SR360: boosting 360-degree video streaming with super-resolution. In *Proc. ACM NOSSDAV*. 1–6.

[3] Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *Proc. AAAI*, Vol. 32.

[4] Lovish Chopra, Sarthak Chakraborty, Abhijit Mondal, and Sandip Chakraborty. 2021. PARIMA: Viewport Adaptive 360-Degree Video Streaming. In *Proc. ACM WWW*. 2379–2391.

[5] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779* (2016).

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*. 1126–1135.

[7] Yun Gao, Xin Wei, and Liang Zhou. 2020. Personalized QoE improvement for networking video service. *IEEE Journal on Selected Areas in Communications* 38, 10 (2020), 2311–2323.

[8] David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).

[9] Yuxiang Hu, Yu Liu, and Yumei Wang. 2019. Vas360: Qoe-driven viewport adaptive streaming for 360 video. In *Proc. IEEE International Conference on Multimedia & Expo Workshops*. 324–329.

[10] Mohammadreza Jamali, Stéphane Coulombe, Ahmad Vakili, and Carlos Vazquez. 2020. LSTM-based viewpoint prediction for multi-quality tiled video coding in virtual reality streaming. In *Proc. IEEE ISCAS*. 1–5.

[11] Xiaolan Jiang, Yi-Han Chiang, Yang Zhao, and Yusheng Ji. 2018. Plato: Learning-based Adaptive Streaming of 360-Degree Videos. In *Proc. IEEE LCN*. 393–400.

[12] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proc. ICML*. 1928–1937.

[13] Anh Nguyen, Zhisheng Yan, and Klara Nahrstedt. 2018. Your attention is unique: Detecting 360-degree video saliency in head-mounted display for head movement prediction. In *ACM MM*.

[14] Sohee Park, Arani Bhattacharya, Zhibo Yang, Mallesham Dasari, Samir R. Das, and Dimitris Samaras. 2019. Advancing user quality of experience in 360-degree video streaming. In *2019 IFIP Networking Conference (IFIP Networking)*. 1–9.

[15] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proc. ACM MobiCom Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.

[16] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *Proc. ACM MMSys*. 114–118.

[17] Thomas Stockhammer. 2011. Dynamic adaptive streaming over HTTP– standards and design principles. In *Proc. ACM MMSys*. 133–144.

[18] Thomas Alsop. 2022. VR headset unit sales worldwide 2019-2024. https://www.statista.com/statistics/677096/vr-headsets-worldwide/. Accessed: 2022-03-29.

[19] Jeroen van der Hooft, Maria Torres Vega, Stefano Petrangeli, Tim Wauters, and Filip De Turck. 2019. Optimizing adaptive tile-based virtual reality video streaming. In *Proc. IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 381–387.

[20] Fangxin Wang, Cong Zhang, Jiangchuan Liu, Yifei Zhu, Haitian Pang, Lifeng Sun, et al. 2019. Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe. In *Proc. IEEE INFOCOM*. 910–918.

[21] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763* (2016).

[22] Xueting Wang, Yu Enokibori, Takatsugu Hirayama, Kensho Hara, and Kenji Mase. 2017. User Group Based Viewpoint Recommendation Using User Attributes for Multiview Videos. In *Proc. ACM MM Workshop on Multimodal Understanding of Social, Affective and Subjective Attributes*. 3–9.

[23] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A dataset for exploring user behaviors in VR spherical video streaming. In *Proc. ACM MMSys*. 193–198.

[24] Lan Xie, Zhimin Xu, Yixuan Ban, Xinggong Zhang, and Zongming Guo. 2017. 360probdash: Improving qoe of 360 video streaming using tile-based http adaptive streaming. In *Proc. ACM MM*. 315–323.

[25] Huaizheng Zhang, Linsen Dong, Guanyu Gao, Han Hu, Yonggang Wen, and Kyle Guan. 2020. DeepQoE: A multimodal learning framework for video quality of experience (QoE) prediction. *IEEE Transactions on Multimedia* 22, 12 (2020), 3210–3223.

[26] Ran Zhang, Jiang Liu, Fangqi Liu, Tao Huang, Qinqin Tang, Shangguang Wang, and F. Richard Yu. 2022. Buffer-Aware Virtual Reality Video Streaming With Personalized and Private Viewport Prediction. *IEEE Journal on Selected Areas in Communications* 40, 2 (2022), 694–709.

[27] Yuanxing Zhang, Pengyu Zhao, Kaigui Bian, Yunxin Liu, Lingyang Song, and Xiaoming Li. 2019. DRL360: 360-degree video streaming with deep reinforcement learning. In *Proc. IEEE INFOCOM*. 1252–1260.

[28] Pengyu Zhao, Yuanxing Zhang, Kaigui Bian, Hu Tuo, and Lingyang Song. 2019. Ladernet: Knowledge transfer based viewpoint prediction in 360○ video. In *IEEE ICASSP*.