Mosaic: Advancing User Quality of Experience in 360-Degree Video Streaming with Machine Learning

Sohee Park†, Arani Bhattacharya§, Zhibo Yang†, Samir R. Das†, Dimitris Samaras†
†Stony Brook University, §IIIT-Delhi
soheekim@cs.stonybrook.edu, arani@iiitd.ac.in, {zhibyang, samir, samaras}@cs.stonybrook.edu

***Abstract*—Conventional streaming solutions for streaming 360-degree panoramic videos are inefficient in that they download the entire 360-degree panoramic scene, while the user views only a small sub-part of the scene called the viewport. This can waste over 80% of the network bandwidth. We develop a comprehensive approach called *Mosaic* that combines a powerful neural network-based viewport prediction with a rate control mechanism that assigns rates to different tiles in the 360-degree frame such that the video quality of experience is optimized subject to a given network capacity. We model the optimization as a multi-choice knapsack problem and solve it using a greedy approach. We also develop an end-to-end testbed using standards-compliant components and provide a comprehensive performance evaluation of *Mosaic* along with five other streaming techniques – two for conventional adaptive video streaming and three for 360-degree tile-based video streaming. *Mosaic* outperforms the best of the competitions by as much as 47-191% in terms of average video quality of experience. Simulation based evaluation as well as subjective user studies further confirm the superiority of the proposed approach.**

***Keywords*—360-degree video streaming, adaptive video streaming, MPEG-DASH, Machine Learning, Convolutional Neural Network (CNN), 3DCNN, Recurrent Neural Network (RNN)**

I. Introduction

With video streaming proliferating on the Internet [1] interest is growing for immersive video applications. An important application in this space is 360-degree video [2]. 360-degree video is a panoramic video recorded using omni-directional cameras [3]. It is then projected onto 2D using one of the available mapping techniques (e.g. equirectangular, cube, and pyramid). Typically, the user watches the 360-degree video using head mounted display (HMD) or commodity mobile devices (e.g., [4]).

Regardless of the actual mapping used, the existing video streaming ecosystem delivers the full 360-degree scene, while the user views only part of the scene at a given time, called *viewport* (VP). A viewport is about 90° − 120° horizontally, 90° vertically, less than 20% of the full 360-degree scene.

This amounts to a significant wastage of network bandwidth by fetching bits that is never used in actual viewing. Thus, with the prevalent Internet connection speeds [5] the video can only be viewed at a compromised quality. Fig. 1 shows the total download video file size in MB, consumed by conventional 360-degree video streaming compared with the download consumed by only the current viewport. This is shown for multiple video qualities (or bit rates). The highest of the bit rates used is equivalent to the 4K video quality.

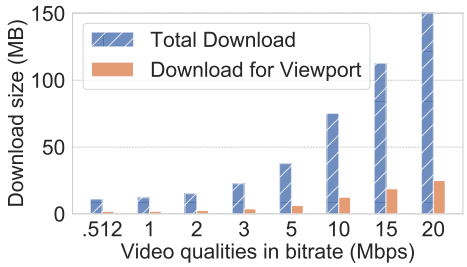


Fig. 1. Total downloaded size vs. download size for only the viewport watched (in MB) for an actual 360-degree video streaming experiment with video encoded in different bit rates. The video is about 1 min long. Further details about the experiments are discussed in Section III.

Clearly, the perfect solution is a video streaming system that fetches *only the information related to the viewport and no more*. The challenge here is as follows: 1) In a video streaming system the video frames are fetched in advance of playing and thus the user's viewport must be predicted in advance in order to do this. 2) The viewport depends on user's attention and thus cannot always be predicted perfectly in advance though several techniques have been proposed recently with varying prediction performance [6]–[9]. Imperfect prediction must be handled adequately. For example, with imperfect prediction, part of the viewport may be missing and thus the user's quality of experience (QoE) will drop – causing the player to either ignore these missing parts or stall until these missing parts are fetched. 3) Viewport prediction must be integrated seamlessly with bit rate control.

As an example, consider Fig. 2. The Subfigure 2(a) shows the scenario where the entire frame is downloaded at a low quality (video bit rate) subject to the available network bandwidth. Subfigure 2(b) shows a scenario when only the viewport portion is downloaded but now at a much higher bit rate. However, if the viewport prediction is imperfect the user may miss part of the viewport (Subfigure 2(c)) leading to a poor quality of experience. The alternative we pursue in this work is to use utilize the viewport prediction to determine which parts of the frame to be fetched at what quality. Thus,

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

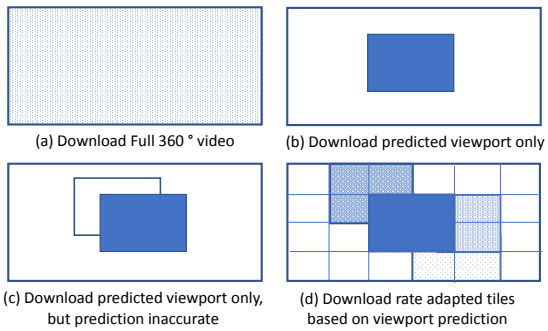


Fig. 2. Example illustrating the tradeoff between accuracy of viewport prediction and video resolution.

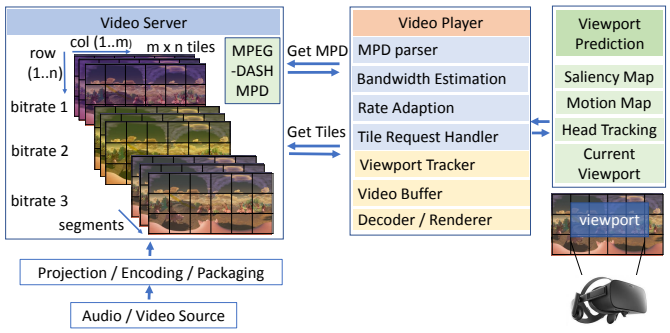


Fig. 3. Overview of *Mosaic* design: a tile-based adaptive 360-degree video Streaming using user viewport prediction.

the portion of the frame that is highly probable is fetched at a higher quality and other less probable areas are still fetched, but at a lower quality and other lower probability areas are not fetched at all (Subfigure 2(d)).

Even for conventional videos, adequate bit rate control is a challenging problem and is still a matter of active research interest [10]–[16]. With 360-degree video a new dimension is now added to this: *a temporal domain problem now transforms to a spatio-temporal problem*. As the example with Fig. 2 has illustrated, appropriate video bit rates both across space and time must now be determined.

While a number of studies have recently looked into optimizing 360-degree video streaming [17]–[24], these have not developed a complete end-to-end solution including bit rate allocation/control across both space and time along with advanced viewport prediction. Though a more recent study [25] proposed such a system, in this study the server has to manage client status, decide which client HTTP request of video it should respond to, making them difficult to scale. It also suffers from low prediction accuracy. In contrast, *Mosaic* does not require any changes to the MPEG-DASH standard. Moreover, *Mosaic* also uses state-of-the-art vision techniques such as 3D-Convolutional Neural Networks (3D-CNN) [26] to ensure that it can achieve accurate prediction for much longer segments than previous studies. This significantly reduces both the computation and network overhead. See a comprehensive review in Section VII.

In this work, we develop an end-to-end 360-degree video delivery system, *Mosaic*, that streams spatial subparts of the video at appropriate encoding rates to maximize the estimated user's quality of experience subject to the prevalent network capacity. (See Fig. 3 for an overview of the major components.) To achieve this, *Mosaic* spatially partitions video frames into rectangular regions called *tiles*, encodes them in in multiple bitrates, and packages them for adaptive 360-degree video streaming.

To utilize the existing video streaming ecosystem, we use MPEG-DASH with spatial relation description (SRD) [27]. We predict the user's viewport based on features such as head tracking data, motion map and saliency map using advanced machine learning methods. We then use a variant of the knapsack problem to choose an optimal video resolution for each tile, given available network capacity and prediction. To handle the prediction error, we factor in a penalty of missing tiles and incorporate this cost within the rate adaptation equation. We evaluate *Mosaic* for 360-degree video streaming on the Internet with a rate-limited client network link and compare its performance with state-of-the-art algorithms for both conventional video streaming and 360-degree video streaming. *Mosaic* provides about 50% better median user perceived video quality than its nearest competitor in our evaluations. Median rebuffering is 3 times less than that of the competitor over WiFi and similar of 4G/LTE emulated network.

A previous version of this paper was published in [28]. The current version adds substantially to the content, including:

- utilization of a more representative metric of QoE as opposed to only the bitrates (Section II-B),
- addition of more experiments to evaluate system performance under various network conditions (Section V), such as latency and bandwidth constraints,
- experiments to justify the choices of parameters used (e.g. the tile size and segment length used for prediction),
- experiments to show the performance of each unit of the system separately, such as QoE modeling, viewport prediction module and adaptive bitrate (ABR) algorithm,
- addition of a more representative baseline technique, and
- addition of subjective user study (Section VI), with illustrations to give an intuitive understanding of the quality.

The rest of the paper is structured as follows. We discuss the background and develop the *Mosaic* system design in Section II. Section III reports the experiment methodology and explains our implementation. We describe the experimental results and our observations in Section IV and Section V. We describe subjective user studies in Section VI. We discuss related works in Section VII and conclude in Section VIII.

II. Mosaic System Design

A. Background

The first step of adaptive streaming of 360-degree videos is to divide the video across both space and time. Across space, the 360-degree frame is split into multiple *tiles* after an equirectangular projection [18], [27]. Across time, each tile is split into multiple chunks of fixed duration, called *segments*. We make a design choice in our work regarding segment duration and number/layout of tiles. We choose 4×6 tiling and 2 sec long segments. Note that larger tiles (i.e., fewer in number) provide a better encoding efficiency, but

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

may cause additional bits to be downloaded as larger tiles may increasingly extend beyond the viewport. Prior work [23], [25] has indicated that 4×6 tiling work well in this regard. Similarly, both too long and too short segment duration could be detrimental to performance. Interested reader can refer to Appendix B for additional study on segment size, where we establish that 2 sec is a good choice.

A <tile,segment> is the unit of encoding, storage or network communication. Each <tile,segment> is encoded in multiple qualities (i.e., resolutions or video bit rates) at the video server. The client dynamically chooses the playback bitrate of each segment based on the network capacity and/or client player buffer levels and sends HTTP requests to the server to prefetch the future segments of selected set of tiles. Ideally, the only tiles that need to be fetched are those corresponding to the user's viewport. We assume that the tiles are smaller than the viewport. In other words, multiple tiles are needed to cover the viewport. Since the viewport at the (future) playback time is unknown when video data is being fetched we first predict the viewport corresponding to the segment being fetched. Given the prediction is always imperfect, *the prediction is modeled as a probability distribution over all possible tiles.* This in turn provides a probability distribution over viewports for the 360-degree video frame. Given this input, our task is to *select the tiles along with their playback bit rates for the next segment to be fetched subject to the prevalent network capacity.* This is fundamentally an optimization problem – maximizing the user's quality of experience subject to the network capacity.

B. Modeling Quality of Experience

In this section, we develop a model for the proposed tile-based adaptive video streaming system for 360-degree videos. We use a control theoretic approach similar to [29].

We model the quality of experience metric of an individual segment, QoE, as consisting of four distinct components.

User Perceived Video Quality: The first component considers the average bitrate perceived by the user and the quality penalty when tiles are missed. Note that higher bitrate improves the quality, whereas missed tiles hurt the quality. The user perceived video quality of the segment is modeled as a linear combination of the bitrate perceived and tiles missed.

Assume that N tiles cover the 360-degree scene, T_j is the j^{th} tile of the segment, R_j is the rate selected for T_j and $D(R_j)$ is the size of the T_j for rate R_j. R_{min} and R_{max} are the minimum and maximum rate available respectively. If T_j is not to be fetched, then $R_j = 0$. Viewports are indicated by V_i. Since the viewport spans over some tiles, quality of tiles in turn defines viewport quality. Then the user's perceived video QoE is assumed to be proportional to the sum of the qualities of the tiles composing the viewport. Mathematically, we define the estimated user perceived video bitrate as:

$$B_i = \sum_{j=1}^{N} P_j q(R_j) O_{ij} \tag{1}$$

where $q(R)$ is a function that maps a video bitrate R to the perceived quality and O_{ij} is overlapping ratio of viewport V_i and T_j. P_j is the probability indicating how likely the user is to view the tile T_j. The set of tiles that are not fetched but still overlap with user viewport are called missing tiles. They cause QoE degradation either by projecting poor quality video (part of the viewport missing) or by stalling to fetch these missing tiles. We define penalty function for such missing tiles as:

$$S_i = \sum_{j=1}^{N} P_j q(R_{min}) L_j, \tag{2}$$

where $L_j = 1$ if T_j is skipped (not fetched), else $L_j = 0$.

Combining the above two models, the estimated quality is modeled as:

$$Q_i = \gamma_1 B_i - \gamma_2 S_i \tag{3}$$

where γ_1 and γ_2 are weights modeling the relative importance of tiles to be fetched vs. not fetched.

Quality Variation Within Viewport: Unless all tiles are fetched at the same quality level, it is possible that the tile quality levels vary across different tiles within the user's viewport during playback. This is because no special mechanisms have been used to ensure same qualities. This may impact the QoE. However, because tiles are being fetched for future playback, it is unknown which tiles will overlap with the future viewport. Therefore we use the standard deviation of qualities of all tiles instead (i.e. $O_{i,j} = 1$), proportional to the probability indicating how likely the user is to view the tile (P_j). Mathematically,

$$E_i = \text{StdDev}\left[q(R_j)P_j |\ j = 1, \ldots, N\right] \tag{4}$$

Quality Variation Across Segments: Quality variation across segments during playback also impacts QoE. As the third component, similar to a metric used for regular adaptive video streaming [29], we use G_i to quantify quality change at each segment transition.

$$G_i = |Q_i - Q_{i-1}| \tag{5}$$

Rebuffering: While we run rate adaptation within the estimated network capacity, network delays and capacity fluctuation could cause the playback buffer to deplete and the client player to stall until the content for the player is downloaded. This stall/rebuffering duration is an important QoE metric. We model the forth component, rebuffering, as follows:

$$T_i = \left(\frac{\sum_{j=1}^{N} D(R_j)(1 - L_j)}{W_i} - B_i\right)_+ \tag{6}$$

where W_i is estimated network bandwidth and B_i is the playback buffer level when the player segment starts to download i^{th} segment. $(a)_+ = a$ if $a > 0$, else $(a)_+ = 0$.

Normally, the initial start time would also be an important QoE metric. We do not present this here, however, as this is similar across all mechanisms we evaluated, since all tiles are to be downloaded for the first segment in our implementation of the client regardless of the mechanism used.

Now we define the quality of experience, QoE, as linear combination of the four distinct components as follows:

$$QoE = \mu_1 Q_i - \mu_2 T_i - \mu_3 G_i - \mu_4 E_i \tag{7}$$

where the μ's are constants modeling contributions of video bitrates, rebuffering and quality smoothness on the QoE. A large μ_1 indicates the user cares for the high quality video

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

bitrates. A large μ_2, μ_3 and μ_4 indicates the user prefers low rebuffering, low quality variance across segment and within viewport respectively.

Our objective is to select and assign rate R_j for each tile T_j (if the tile is not selected to be fetched, $L_j = 1$ and $R_j = 0$) such that QoE is maximized, subject to the estimated download capacity C. So the optimization problem to be solved is:

$$\text{Maximize } QoE \text{ subject to } \sum_{j=1}^{N} D(R_j)(1 - L_j) < C \quad (8)$$

We determine download capacity C as function of the estimated network capacity W_i and/or the player buffer level β_i. $f(W_i, \beta_i) = C$. Similar to previous works [25], [29], we consider several different download capacity estimate functions $f(.)$ as follows:

- $f(W_i, \beta_i) = \delta W_i$.
- $f(W_i, \beta_i)$: Similar to buffer based approach [12], if $\beta_i <$*reservoir* $\delta_1 W_i$, else if $\beta_i <$*cushion* $\delta_2 W_i$, else $\delta_3 W_i$
- $f(W_i, \beta_i) = \delta W_i \beta_i$.

Similar to [15], [29], we consider several different quality functions $q(.)$:

- Linear: $q(R) = R$
- Ratio: $q(R) = R/R_{min}$ or R/R_{max}
- Index: $q(R)$ is an index into a table of R.

Evaluating QoE in terms of PSNR or similar methods (V-PSNR, WS-PSRN) is expected to be expensive for tiled 360-degree video rate adaptation because of the evaluation overhead that needs to run in real-time. Studies such as [8], [23] do use the V-PSNR to evaluate QoE, but there this is done only after the playback/streaming is complete. Note that our algorithm is generic in nature and works for any quality function. We present the results with the linear function in this paper. Additional experimental results with the index function appears in the Appendix C. We model the optimization problem in Equation 8 as a *multi-choice knapsack problem* and solve it using a greedy algorithm.

Algorithm 1 Tiled Adaptive Video Streaming

```
1: Initialize
2: for k ← 1 to M do /* M segments in video*/
3:     Estimate current download capacity C
4:     Estimate tile probabilities P_j, ∀j = 1...N
5:     [R_1..R_N] = SelectRates ([P_1..P_N], C)
6:     Download the tiles in segment k with video rates [R_j]
```

C. Greedy Algorithm

Algorithm 1 selects tiles and their bit rates for each segment to be downloaded. We use the network capacities observed while downloading the previous k segments and use harmonic mean of k observed network capacities as a proxy for the current network capacity. We acknowledge that more sophisticated network capacity estimation techniques may possibly provide better results. The tile probabilities P_i are estimated based on (offline) analysis of the video and user's head tracking data upto this point of the video. The function *SelectRates* (Algorithm 2) is called with C, and P_i's as inputs to determine the selection and rates for each tile T_j.

Algorithm 2 *SelectRates*: Viewport Based Rate Adaptation

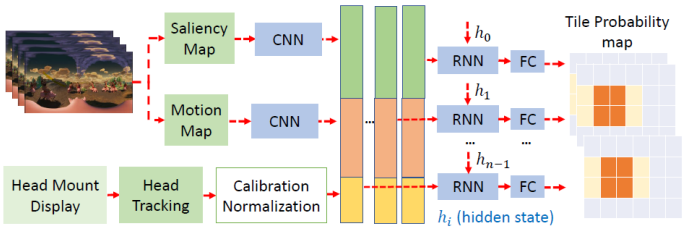
```
   Input: tile probabilities P_j, capacity estimated C
   Output: Rates, bitrates selected for tiles
1: max ← −∞
2: Rates ← {0, 0, ..0} /*initial condition */
3: isSelected ← False
4: while Rates is updated do
5:     R_tmp ← Rates
6:     for i ← 1 to V do /* viewport index*/
7:         R ← R_tmp
8:         for j ← 1 to N do /* tile index*/
9:             if O_{i,j} > 0 then
10:                R_j ← Increase to next rates available
11:        if Σ_{j=1}^{N} D(R_j)(1 − L_j) > C & isSelected then
12:            continue; /* too aggressive.*/
13:        else
14:            QoE ← μ_1 Q_i − μ_2 T_i − μ_3 G_i − μ_4 E_i
15:            if QoE > max then
16:                isSelected ← True
17:                max ← QoE
18:                Rates ← R /*update optimal rates*/
19: return Rates
```

We implement a greedy knapsack-based solution in Algorithm 2. We choose a greedy algorithm as opposed to more sophisticated dynamic programming based solution to minimize the overhead. Our algorithm works as follows. In each iteration (line 7-18), the goal is to find the viewport V_i that maximizes the user perceived quality given C. Best rate selection is at $Rates$. The precondition for each viewport rate selection is at R_{temp}. For each V_i, we increase the quality of the overlapping tiles R_j, if T_j overlaps with V_i, $O_{i,j} > 0$ (line 9). If downloading tiles based on rate selection R exceeds the capacity C (line 11), we discard the selection (line 12).

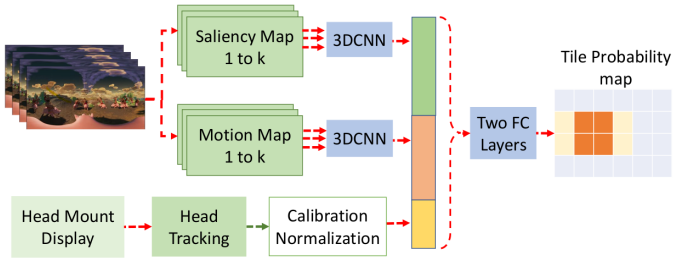
If rate selection is within the capacity estimate C (line 13), we evaluate the expected quality of experience QoE (line 14) and check if it is greater than the best quality so far, max (line 15). The quality function evaluated using Equation 3. If the new rate selection has the maximum quality, we assign this rate R to $Rates$ as the optimal rate that provides the highest expected quality so far (line 18). We evaluate the next viewport in same manner (line 7-18). We repeat this while $Rates$ updates (line 4) and total download is within C.

To compute the time complexity of Algorithm 2, we note that we iterate over each tile (line 8) and over each viewport(line 6), thus having $O(V \times N)$ number of steps, where V is the number of viewports and N is the number of tiles. Since there is only a small number of available rates for each tile, we consider this as a constant. Thus, the time complexity of *Mosaic*'s algorithm is equal to $O(V \times N)$ per segment. In *Mosaic*, we have a relatively small number of viewports with different overlapping tiles. Thus, as shown in the evaluation, the overhead of rate adaptation is relatively small.

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.



(a) Architecture of CNN+RNN (LSTM) based prediction.



(b) Architecture of 3DCNN based prediction.

Fig. 4. Mosaic Prediction Algorithms: CNN+RNN (LSTM) and 3DCNN.

D. Viewport Prediction

We develop two different viewport prediction mechanisms. The output of the viewport prediction system is the probabilities of different tiles indicating how likely the user is to view the tile. One can assume that the viewport is defined by its center point inside the 360-degree frame. The basis of viewport prediction is that users tend to look at interesting (salient) features in the scene that captures their attention. Video analysis can reveal these features. In addition, temporally meaningful correlation exists in the viewer's attention. Overall, a combination of video analysis (static) and head tracking for the user in the past (real time, dynamic) can be used to predict the user's viewport in near future (say, next several seconds). We use suitable machine learning (ML) algorithms to predict the viewport. The saliency and motion maps of the video and user head tracking trace are used as input for the prediction.

The saliency of a pixel indicates how much this pixel stands out from its neighboring pixels, and thus saliency is directly related to how likely a pixel/part of an image can attract the viewer's attention. The motion map captures the movement of each pixel in two consecutive images in a video. In particular, each pixel of a motion map describes how much the corresponding pixel has moved from the previous frame in the original video.

The choice of the actual algorithms are important. Our initial approach used simpler approaches such as linear regression and SVM (similar to prior work [8]). However, this resulted in poor accuracy (<65%) for lookaheads of more than 1 sec. This led us seek out more sophisticated models using neural networks that can capture latent features of the video. In the work we present here, we use convolutional neural networks (CNNs) to capture spatial features in the video and Recurrent Neural Networks (RNN) to capture the temporal features. We take a similar approach to [7], where we combine CNN and RNN with motion and saliency maps of the video. However, they create two separate networks making the prediction cumbersome and slow. Instead, we extract the saliency and motion maps of the video via off-line analysis and embed them with the viewport trajectory of the current user (this is obtained from the head tracking data at the client) and use this combined information as an input to a single network. We also propose an alternative design using 3DCNN to improve the prediction performance. The two prediction mechanisms used are described below.

CNN+RNN (LSTM): We use the motion and saliency maps along with users' head tracking data, feed it to a CNN model combined with RNN. More specifically, we collect motion map and saliency map for each frame, encode in a common vector and consider it as single input instance to the CNN. We use a pre-trained ResNet-101 model [30] to extract the spatial features. The CNN's spatial features are then combined with the head tracking data and fed into an RNN. In this way, this model computes the tile probabilities by modeling the spatial region that users are likely to pay more attention. Fig. 4a shows the architecture of our CNN+RNN model. As an RNN, we use Long Short Term Memory (LSTM) that captures long term dependencies among the frames.

TABLE I
HYPERPARAMETERS OF LSTM AND 3DCNN-BASED VIEWPORT PREDICTION

LSTM	Hidden Units	256
	Learning rate	0.01
	Updater	Stochastic Gradient Descent
	Loss	Binary Cross Entropy
3DCNN	Learning rate	0.01
	FC layer 1/2 Hidden Units	1024 / 200
	Activation	ReLU
	Loss	Binary Cross Entropy

3DCNN: The CNN+RNN model learns the spatial feature and the temporal feature of the video independently and separately. However, in reality, there often exists spatio-temporal correlation in a user's gaze movement (e.g., a user's attention following a flying bird in the sky as time goes by). Inspired by the works in action recognition [31], we adopt a 3DCNN model to extract the spatio-temporal feature from the videos. Fig. 4b depicts the architecture of our 3DCNN approach. Specifically, we use the pre-trained I3D [26] as our 3DCNN component and apply a simple two-layer fully connected (FC) network to map from the feature space to the tile probability map space. Notably, compared with the CNN+RNN model, the prediction latency of 3DCNN is significantly lower as the 3DCNN applies a simple two FC layers for prediction while CNN+RNN applies an RNN model (e.g., LSTM) which requires much more computation. Both models are trained off-line using the hyperparameters listed in TABLE I.

III. Evaluation Testbed

We evaluate *Mosaic* and compare its performance with other state-of-the-art methods using a testbed. The testbed consists of a video server, a client and the viewport prediction system (see Fig. 5). In the following we describe these system

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

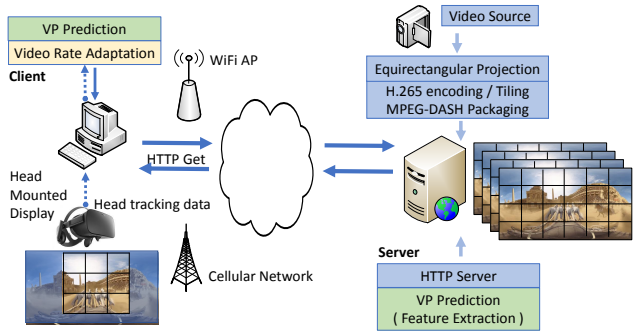


Fig. 5. Evaluation setup.

components and also the data set to be used for evaluation. A high-level block diagram appears in Fig. 3.

A. System Components and Data Set

The server uses MPEG-DASH compliant HTTP adaptive streaming [32] on a Linux platform (Ubuntu 16.04). The player is implemented based on an open-source tile-based adaptive streaming video player, MP4Client [33]. The viewport prediction system interfaces with the player. The training part of the viewport prediction is done off line and only the inference runs on the client. The overhead of the viewport prediction and rate control algorithms (Algorithms 1 and 2) are negligible in our implementation.

For evaluation we use ten popular 360-degree videos of different categories for which head tracking dataset is available (50 users for each video) [34]. Each trace in the head tracking data set contains a user's head position (yaw, pitch and roll) for every frame. From this the viewport and viewport-specific tiles are derived. For each of these videos, we also extract the saliency and motion maps as described in Section II. The videos are typically about 1 min long. Each video is split temporally in 2 sec long segments.[1]

Each video is encoded and projected using equirectangular projection using 4K quality (3840×1920). Since we need multiple qualities for our experiments, we transcode the video in six different bit rates – 512 Kbps, 2 Mbps, 5 Mbps, 10 Mbps, 15 Mbps and 20 Mbps. We use the open source HEVC encoder `kvazaar` [35] for the transcoding. HEVC 'motion constrained'[2] 6×4 tiling is used in the transcoding. Using the GPAC MP4Box [36], we package the HEVC encoded videos in MP4 containers and generate the necessary MPD (Media Presentation Description) including SRD (Spatial Relational Description) [27]. The MPD file serves as the manifest which client references as the video catalog for HTTP requests for various tiles for video segments. The MPD file contains two sets of elements— *AdaptationSet* and *RepresentationSet*. One *AdaptationSet* in the MPD specifies each tile with tile index, encoded bitrates each at a *RepresentationSet*, bandwidth required, and the media (audio/video) file name. Client requests the video tiles with specific qualities selected by the rate adaptation algorithm. This enables the standard MPEG-DASH SRD capable server to work with *Mosaic*, without server side modification.

We implement the CNN+RNN (LSTM) and 3DCNN based prediction using PyTorch [37] and train the network using the saliency map, motion map, and head tracking dataset [34]. Specifically, we use 6 videos out of 10 available and randomly sample 100 video segments from each video. We randomly select 12 users' head tracking data of which 80% is used for training and 20% for validation. For testing, we use the other 4 videos and the head tracking data of 20 different users (neither the videos nor the users' head tracking data are seen during the training phase). Given 30 video frames and head tracking data, the prediction system outputs the tile probabilities of next 30 frames (assuming 30 frames/sec framerate).

To implement the client, we extend the MP4Client [36] adding an interface to the prediction system, rate adaptation and additional logging for evaluating the QoE metrics (to be described in the next subsection). Note that MP4Client requires downloading all the tiles of the first segment and the first tile of each subsequent segment to use the information for decoding. Without modification, we use the existing MP4Client modules such as scheduling and downloading of MPD file and tiles, MPD parser, video buffer management, decoding and rendering. Bandwidth estimation directly affect the rate control and the buffer management determines how much to pre-fetch to avoid rebuffering, which are crucial part of rate adaptation. We use harmonic means of previous k segments download trajectories. We plan to enhance the rate adaptation using more advanced data-driven machine learning techniques in the future. We empirically determine the weight parameters, γ_1 for B_i and γ_2 for S_i, 0.9 and 0.3 respectively (Equation 3).

B. Network Setup

The client connects to the server over the Internet through a WiFi network using commodity Access Point (2.4 GHz band) capable of providing a throughput upto 60 Mbps. We use `tc` to emulate different challenging network conditions and configure multiple realistic traffic control settings: bandwidth limit of 3, 5, 7, and 10 Mbps and average delay about 20-40 ms. To evaluate rate adaptation using realistic network condition, we use throughput traces of existing 4G/LTE dataset [38]. A number of studies have evaluated the system over emulated network using public throughput traces [15], [25], [29]. To reflect prevalent Internet connection speed [5], we shape the bandwidth by scaling it down to average bandwidth around 16.68 Mbps and standard deviation of 6.6 Mbps. To save space, we present the results over WiFi with 10 Mbps bandwidth limit and emulated network using five traces of the 4G/LTE dataset.

C. QoE Evaluation Metrics

While the rate adaptation algorithm uses prediction probabilities, user perceived quality is measured in deterministic fashion. In this section, we define several QoE metrics and empirically evaluate the performance of *Mosaic* based on these metrics:

[1] We have experimented with other segment lengths. But 2 sec segments work well. See Appendix B for more analysis on this topic.

[2] This signifies that the decoding and rendering can be done independently on the tiles [23].

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

User Perceived Video Quality: We define user perceived video quality as the sum of the qualities ($q(.)$ function, II-B) of viewed tiles during playback. If a tile is only partially in view, the overlapping ratio is used to weight the contribution of this tile. Missing tiles within the viewport contribute zero, $q(0) = 0$. Note that we use linear quality function of video bitrate. We evaluate this metric with subjective evaluation in Section VI.

Mathematically, this is somewhat similar to Equation 1 but only relates to the actual viewport used by the user and considers what happens during playback. If there are M segments, the average user perceived quality during playback is given by,

$$Q = \frac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{N} q(R_j)O_{i,j} \tag{9}$$

where R_j is the rate of tile T_j in this interval, $O_{i,j}$ is overlapping ratio of this interval's viewport and T_j. Note that we reuse the notations with a slightly different definition.

Quality Variation Within Viewport: To evaluate the quality variation among tiles within a viewport during playback, we use E_i, standard deviation of weighted qualities of tiles that overlaps the viewport of the segment during playback. Mathematically, this is somewhat similar to Equation 4, Section II-B but only relates to the actual viewport used by the user and considers what happens during playback. This is averaged over all segments, V_t.

$$V_t = \frac{1}{M}\sum_{i=1}^{M} \text{StdDev}\left[q(R_j)O_{i,j}|\ j = 1, \ldots, N\right] \tag{10}$$

Quality Variation Across Segments: To evaluate quality variation across segments during playback, we use V_s for quantify quality change at each segment transition, averaged over all such transitions. If there are M segments as follows:

$$V_s = \frac{1}{M-1}\sum_{i=2}^{M} |\sum_{j=1}^{N} q(R_j)O_{i,j} - \sum_{j=1}^{N} q(R_j)O_{i-1,j}| \tag{11}$$

Rebuffering: We quantify the duration of rebuffering where playback buffer depletes and the client player stalls until the content for the player is downloaded. If there are M segments as follows, rebuffering averaged over all segments as defined as follows:

$$T = \frac{1}{M}\sum_{i=1}^{M} T_i \tag{12}$$

where T_i is the rebuffering for segment i as defined at Equation 6.

Quality of Experience: So far, we presented the *Mosaic* evaluation methods using different QoE metrics. However, often in many literature [15], [25], [29], QoE is presented in one quantitative representation, combining multiple QoE metrics. Similarly, we define a average user perceived quality of experience, over M segments, QoE_1^M, as the linear sum of user perceived bitrates, rebuffering, and smoothness for all video segments as follows:

$$QoE_1^M = \mu_1 Q - \mu_2 T - \mu_3 V_s - \mu_4 V_t \tag{13}$$

where the μ's are constants modeling contributions of video bitrates, rebuffering and quality smoothness on the QoE The approach we have used in evaluating the quality of experience is similar to the existing literature [15], [29]. QoE is to present a quantitative measurement of user perceived quality of experience. To save space, we present the results using 3, 4, 1, and 2 for μ_1, μ_2, μ_3 and μ_4 respectively.

IV. Evaluation Results

The goal in this section is to evaluate *Mosaic* in our testbed and compare its performance vis-a-vis other state-of-the-art and baseline methods. We also separately evaluate the performance of the viewport prediction method. The algorithms we consider for comparative evaluation are described first. Then we proceed with the evaluation results.

A. Suite of Algorithms

We consider two state-of-the-art algorithms for regular, non-tiled adaptive video streaming [12], [14]. These provide the baseline for our comparisons. We also consider three variations of tiled adaptive video streaming algorithms [7], [23], [25]. In our knowledge these are three most recent works closest to *Mosaic*. The five algorithms are described briefly below.

BBA (Buffer-Based Algorithm): BBA [12] is a buffer-based rate adaptation algorithm for conventional adaptive video streaming with no tiling. The entire frame is downloaded regardless of user's viewport. We use BBA implemented in the GPAC MP4Client described before and assume that the entire frame is just a single tile (i.e., 1×1 tiling).

BOLA (Buffer Occupancy-Based Lyapunov Algorithm): BOLA [14] also does not use tiling and is another example of state-of-the-art in adaptive video streaming. BOLA formulates bitrate adaptation as a utility maximization problem that uses Lyapunov optimization techniques. We use BOLA in our testbed using a similar approach as BBA by downloading a single (or, 1×1) tile in a segment with a selected quality.

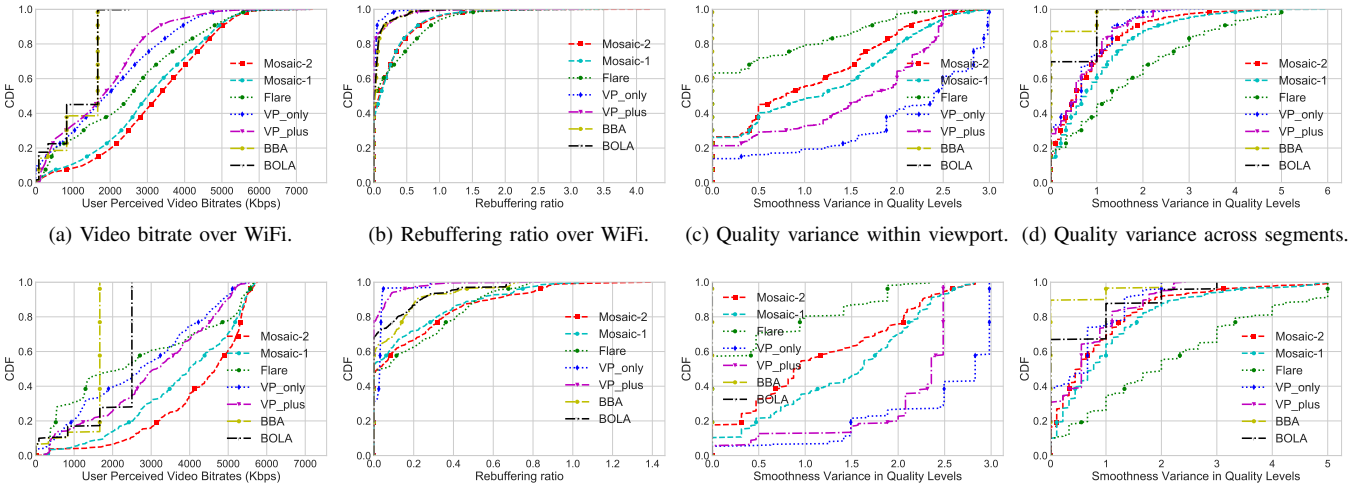
VP_Only (Viewport Only): This is an implementation of a recent work [7] that downloads the tiles in the viewport as predicted by the LSTM-based technique similar to what we described before. All tiles with probability of appearing in the viewport greater than a threshold (0.5) are downloaded in highest quality possible given network capacity and the other tiles are not downloaded.

VP_Plus (Viewport Plus): This technique downloads tiles with a high probability of appearing in the viewport with the highest quality possible given network capacity and all other tiles in the lowest quality [23].[3]

Flare [25]: Using previous segment viewing history, it predicts the viewport center using simple machine learning techniques

[3]The evaluation presented in [23] uses a 'static' viewport which obviously presents very poor results in a real situation. Their evaluation has a different goal. We do not present any viewport prediction method. So, we apply the same viewport prediction as used for VP_Only for ease of comparison.

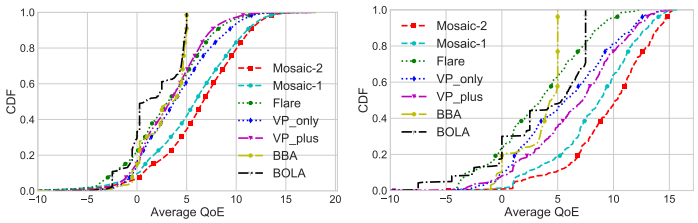
This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.



(a) Video bitrate over WiFi. (b) Rebuffering ratio over WiFi. (c) Quality variance within viewport. (d) Quality variance across segments.

(e) Video bitrate over LTE traces. (f) Rebuffering ratio over LTE traces. (g) Quality variance within viewport. (h) Quality variance across segments.

Fig. 6. Video bitrates, rebuffering ratio, quality variations with viewport and across segments for different methods. (a)–(d) over WiFi network with bandwidth shaping of maximum 10 Mbps using Linux *tc* utility. (e)–(h) over emulated network with LTE traces.



(a) Average QoE over WiFi. (b) Average QoE over LTE traces.

Fig. 7. Quality of experience (a) over WiFi network with bandwidth shaping of maximum 10 Mbps using Linux *tc* utility, (b) over emulated network with LTE traces.

such as LR (Linear Regression), Ridge Regression (RR), and Supporting Vector Machine (SVM). It then applies two-step rate adaptation: first, classification of tiles then assigning bit rates for each class. We implement the LR for prediction and apply 4 level classification.

***Mosaic*-1, *Mosaic*-2:** In the evaluation, we implement *Mosaic* using two prediction techniques indicated by labels *Mosaic*-1 and *Mosaic*-2 for the CNN+RNN (LSTM) and 3DCNN prediction respectively.

B. Results for Viewport Prediction

We first evaluate the viewport prediction component of *Mosaic*. We evaluate LSTM and 3DCNN-based predictions and compare with LR baseline technique in two categories given the available ground truth head tracking data [34] that provides the instantaneous user viewport: i) testing and training data from the same videos and the same users, but each video segment randomly sampled, ii) testing and training data randomly sampled from different videos and different users.

We use four different metrics to evaluate the performance of the viewport prediction techniques.

Accuracy: *Accuracy* is the ratio of number of tiles correctly predicted to be viewed or not viewed to the total number of tiles. While accuracy is used to measure the performance of

TABLE II
PREDICTION ACCURACY

LSTM	Accuracy	Precision	Recall	F1
Same videos/same users	89.80	73.69	59.16	0.66
Diff videos/diff users	88.43	67.04	56.03	0.61
3DCNN				
Same videos/same users	92.21	83.85	83.83	0.79
Diff videos/diff users	91.69	83.99	71.92	0.77
Linear Regression				
All videos	79.46	49.82	49.13	0.49

prediction techniques [6], [25], it is often not sufficient to evaluate a prediction model. Especially when the cost of false negative or false positive is high.

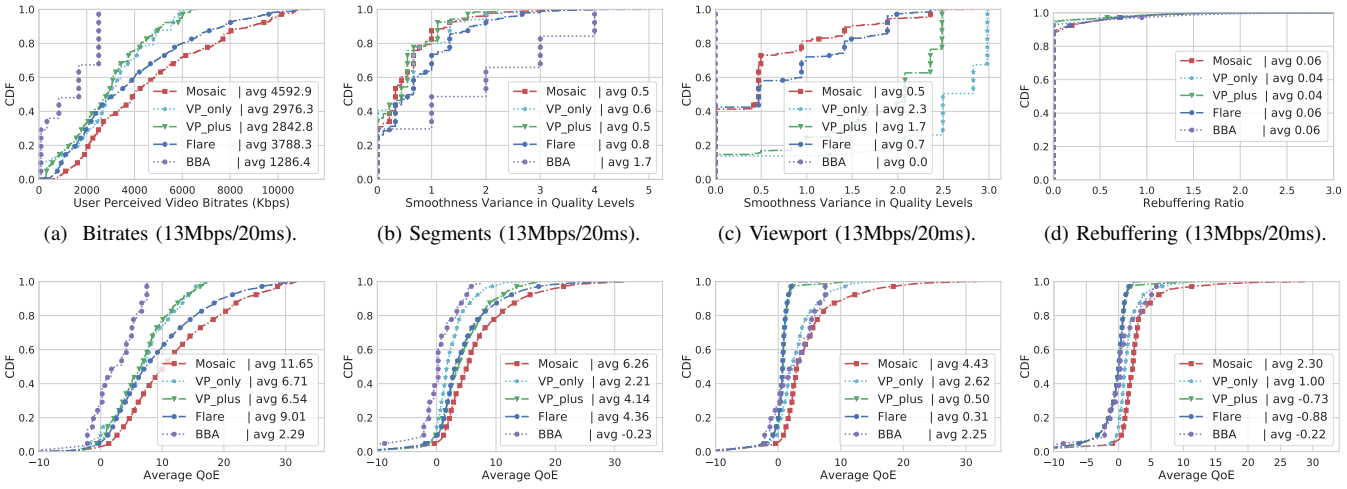
Precision: *Precision* is the ratio of number of tiles correctly predicted to be viewed to the number tiles to be viewed both correctly and incorrectly. The higher the precision is, the lesser tiles are incorrectly predicted to be viewed. Thus, having a high precision is necessary to provide high bitrates to the viewed tiles, especially when the bandwidth is limited.

Recall: *Recall* calculates the ratio of number of tiles predicted to be viewed over number of tiles actually viewed. The higher the value of recall is the smaller number of tiles are incorrectly predicted not to be viewed. Having a high recall is necessary to ensure fewer tiles are missed.

TABLE II shows the prediction accuracy, precision, recall and F1-score of the LSTM and 3DCNN-based predictions. We also add the Linear Regression results to compare with LSTM and 3DCNN. Note that as expected the 3DCNN-based method performs somewhat better than LSTM in general. 3DCNN and LSTM methods outperform the other method, in all scores, accuracy, predicion, recall and F1 scores. Training and testing with different videos/users (as opposed to the same) have negligible impact on performance. This shows that these methods have tremendous potential. The video server just need to collect enough training data to be effective. This does not necessarily need to be from the same users.

The prediction computation is efficient. Excluding training of the neural networks which can be done offline on the

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.



(a) Bitrates (13Mbps/20ms). (b) Segments (13Mbps/20ms). (c) Viewport (13Mbps/20ms). (d) Rebuffering (13Mbps/20ms).

(e) QoE (13Mbps/20ms). (f) QoE (6Mbps/20ms). (g) QoE (13Mbps/80ms). (h) QoE (6Mbps/80ms).

Fig. 8. Evaluation over simulated network with average 13 Mbps and 6 Mbps and average RTT 20ms and 80 ms. User Perceived bitrates (a), quality variance across segments (b), quality variance within viewport (c), rebuffering ratio (d), and quality of experience (e)–(h). When RTT is high, both the quality degradation in Flare and VP_Plus is more serious and they download the tiles in lower qualities most of time, resulting in poor user perceived bitrates. This is one reason for poor QoE of Flare and VP_Plus in (g) and (h).

server, the prediction takes about 0.6ms for 3DCNN, 12.7ms for LSTM on a CPU (Intel Xeon E5-1650 v3 Haswell-EP 3.5GHz). The speed improves for LSTM when a GPU is used. On Nvidia Titan X GPU, it takes 0.7ms for 3DCNN, 2.8ms for LSTM. This is fast enough for a seamless video experience. While we have not evaluated the performance on mobiles, we expect them to perform well. For example, a recent study [39] shows similar or better performance on mobile GPUs relative to a Xeon CPU similar to what we also have used. We also expect mobile GPU performance to improve significantly in future and neural network accelerators to be available. The trained model is small, about 38 MB for 3DCNN and 28 MB for LTSM.

C. Results for Streaming Performance

Fig. 6a and Fig. 6e plot the user perceived video quality during playback in a CDF for all evaluated algorithms over WiFi and emulated network using the 4G/LTE dataset respectively. With *Mosaic* LSTM and 3DCNN-based predictions yield average user perceived bitrates of 3.0 and 3.21 Mbps over WiFi, and 3.78 and 4.24 Mbps using the 4G/LTE dataset respectively. This is in comparison to Flare, VP_Only and VP_Plus which offer in the average about 2.46, 1.97 and 1.77 Mbps over WiFi and 2.58, 2.67 and 3.01 Mbps over the 4G/LTE emulated network, respectively. Thus, *Mosaic* with 3DCNN offers almost 30-81% improvement over WiFi in quality at the average versus the state-of-the-art. It also offers 40-65% improvement over 4G/LTE emulated network. As expected, BBA and BOLA both perform somewhat poorly as they download the entire panoramic frame.

Fig. 6b and 6f plot rebuffering ratio (fraction of playback time spent in stalls) for 1-minute video playback. Note that the average rebuffering ratio of *Mosaic* is about 13-15% less than Flare over WiFi and upto 15% less than Flare over 4G/LTE. While *Mosaic* does not show the best performance in rebuffering and quality variance within viewport, it achieves overall the highest QoE.

Fig. 6c–6d and Fig. 6g–6h plot several other metrics – quality level variation within a viewport and across the segments over WiFi and over 4G/LTE respectively. For quality variation within viewport, the spatial quality changes, *Mosaic* with LSTM and 3DCNN are better than VP_Only and VP_Plus. Note, however, BBA and BOLA do not have any variation within viewport as they do not use tiling but bring in the entire frame always. For variation across segments, *Mosaic* is better than Flare. Here, however, BBA performs better (almost negligible variation) due to a conservative rate control.

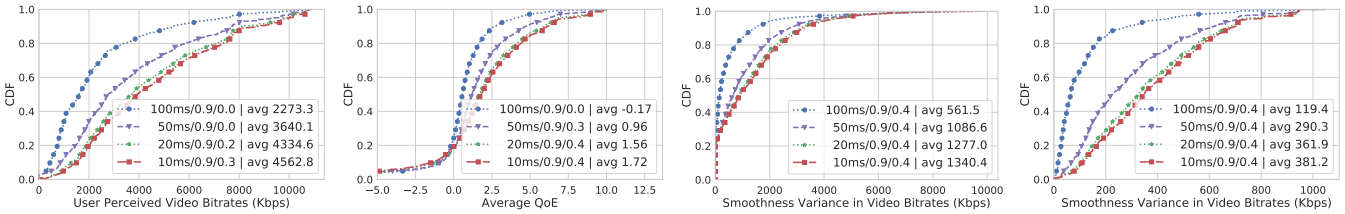
Fig. 7 plots the average QoE per segment over all videos considered over WiFi and over emulated network with 4G/LTE traces. *Mosaic* has 58-107% higher QoE over WiFi and 47-191% higher over emulated network with 4G/LTE traces than VP_Only, VP_Plus, and Flare. Overall, *Mosaic* (specifically with 3DCNN) provides a significantly better video quality during playback relative to other state-of-the-art 360-degree video streaming.

We also measure the overhead of running the bitrate adaptation algorithm of *Mosaic*. In general, we find an overhead of $0.1 - 0.4$ ms per segment. We note that this overhead is quite small, and thus our technique is feasible even on smartphones.

V. Simulation

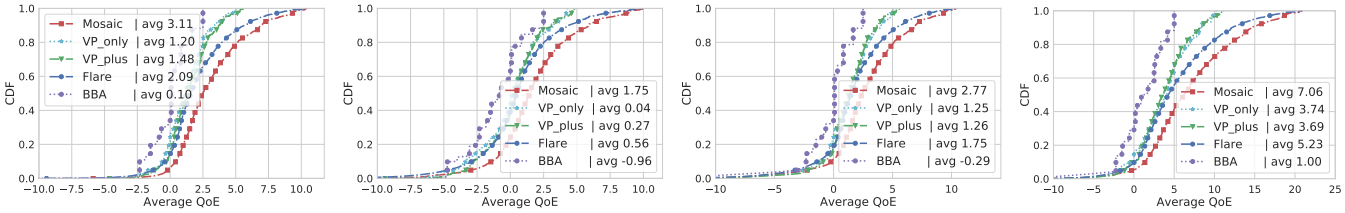
We further evaluate *Mosaic* and compare its performance with other techniques using simulation environment. To benchmark algorithms under different configurations and different network settings, simulation makes it more efficient to evaluate the performance than with a real video player. It is because the real player would have to wait until the actual segment download is complete, or even entire video download is complete before evaluating the quality of experience for each video playback. Details of the environment is discussed at Section V-A). We compare the performance of *Mosaic* with the performance of one conventional video streaming technique (BBA) and three tile based 360-degree video streaming

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.



(a) Bitrates. (b) Quality of Experience. (c) Segment Quality Variance. (d) Viewport Qualify Variance.

Fig. 9. User perceived bitrates, rebuffering, and quality of experience of *Mosaic* using different weight parameters (γ_1/γ_2) over different network delays. Different weigh parameters performs best at different network delays and quality metrics. Over average 100ms RTT network, γ_2=0 gives the highest bitrates and QoE where as γ_2 = 0.3 works best for low delay network. Smoothness variance over segment and within viewport is best when γ_2 = 0.4.



(a) (1,1,1,1). (b) (1,1,2,2). (c) (1,4,1,1). (d) (2,4,1,2).

Fig. 10. QoE with different weight parameter μ's (μ_1, μ_1, μ_3, μ_4). *Mosaic* has the highest average QoE values proving it outperforms VP_Only, VP_Plus, and Flare in all conditions.

technique (VP_Only, VP_Plus, and Flare). We use *Mosaic* with 3DCNN method as it shows the better performance than *Mosaic* with LSTM method (Details of the evaluation results is at Section IV). We skip BOLA as we have already shown that tiled based approaches are superior than conventional video streaming techniques (BBA and BOLA in our evaluation) and BBA performs better than BOLA in our evaluation. Details of the baseline techniques in comparison is at Section IV-A.

A. Simulation Environment

The simulator faithfully models the dynamics of the tiled video player and the network. It simulates the adaptive video streaming over HTTP. It maintains the video player buffer level and assigns download time of each tile of a segment, $< segment, tile >$ unit, based on the chosen tile quality and instantaneous network throughput that is determined by the input trace. Then it simulates the download of the video tile and the storing of the tile to the player buffer. The simulator also models the player buffer by decreasing for the playback of videos and increasing of buffer upon the download complete of a segment.

For each pair of video and the head tracking data, we can emulate different streaming conditions and calculate the quality of experience. The simulated environments are based on real network traces of public datasets: FCC broadband dataset [40] and a mobile dataset [41]. We linearly increase the bandwidth to reflect prevalent Internet connection speed [5]. Many existing studies also use such simulated environment as ours [15], and/or use network traces to simulate the real network condition [12], [15], [29]. We use harmonic means of the previous network capacities observed while downloading the previous segments (upto 8) as a proxy for the current network capacity.

B. Evaluation Results

To understand how network conditions such as capacities and delays affect the user quality of experience of 360-degree video streaming, we simulate different network conditions and evaluate how *Mosaic* performs in comparison with other techniques.

Network condition and QoE: Fig. 8 plots QoE metrics of the *Mosaic* and baseline techniques over 4 different network capacities and delays. For brevity, we provide the drill-down results for one case and final QoE plots for all 4 cases. Fig. 8a through Fig. 8c plot the user perceived bitrates, the quality variance across segments and within viewport, respectively, for one such case. Note that *Mosaic* has the highest bitrates and lowest variations. Fig. 8d plots the rebuffering ratio for the same case. All the methods has about 4–6% rebuffering ratio, which makes about 1 sec difference in a 60 sec video. Fig. 8e – Fig. 8h show the final QoE plots for the all 4 cases studied.

From left to right (Fig. 8e – Fig. 8h), as network changes from higher capacity / low RTT network (13Mbps/20ms) to lower capacity / higher RTT, the quality of experience degrades more in Flare and VP_Plus than in VP_Only or *Mosaic*. This is because higher RTT further decreases the performance when there are more number of tile downloads (i.e. larger number of HTTP Request/Response). BBA, for example, has one HTTP Request/Response (video segment is in a file) for each segment, which shows the impact is less even when streaming over higher RTT network. This lead us to conclude that, in deploying tiled 360-degree video streaming over existing adaptive video streaming ecosystem, high delays significantly cause more quality degradation in 360-degree video streaming than in conventional video streaming. If player has to bring in more tiles (such as VP_Plus or Flare which brings all tiles of entire 360-degree scene), it adds up the delays even further.

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

(a) Coaster using VP_Plus (b) Coaster using *Mosaic* (c) Ride using VP_Plus (d) Ride using *Mosaic*

Fig. 11. Frame captures of video session for two videos each with *Mosaic* and VP_Plus. Video frames at (b) and (d) show that parts of the viewport have poor quality (red rectangular areas) for VP_Plus.

This also explains that VP_Only or *Mosaic* hits less impact by the higher delays than Flare and VP_Plus.

Overall, *Mosaic* has the highest QoE over different network conditions (Fig 8e – 8h) and prove it adapts well to the various network conditions.

Weight Parameters: To further optimize the quality of experience with *Mosaic*, we evaluate how different γ_1 and γ_2 weight parameter values affects the quality of experiences. In other words, in low delayed network, would bringing more tiles outside the predicted viewports give higher QoE? How about in low capacity and high delay network? (Note γ_1 and γ_2 are weight parameters to tiles to download and to skip in computing expected quality in bitrates. Please see Eq. 1, Eq. 2 and Eq. 3 for details). Fig. 9 shows the average bitrates perceived by viewers for different weight parameters, where a higher bitrates indicates a better quality of experience. Different γ_1 and γ_2 achieves the best results under different network conditions. Fig. 9 plots the QoE metrics under different RTTs (average 100ms, 50ms, 20ms and 10ms) and with different γ_1 and γ_2 values. We find in high delay network (avg 100 ms), it has the highest bitrates when γ_2 is zero, whereas in low delay network (avg 20ms, 10ms) it achieves highest bitrates when γ_2 is 0.4. Fig. 9 plots CDFs of QoE metrics over different network and with different weight parameters that gives the best results under the network condition. Note that different weight parameters produce the best results under different network conditions, which can be addressed by applying ML techniques for rate adaptation [42].

QoE **parameters (μ's):** We also evaluate how different μ's affects the *Mosaic* performance. In all condition, *Mosaic* has the highest QoE values compared to the based line techniques. Fig. 10 is four different sets where penalty for quality variation or rebuffering is set, or higher reward for video bitrates. This shows *Mosaic* adapts well to maximize the QoE and is not dependent on the μ's parameter values.

To separately evaluate the ABR algorithms, we also fix the viewport prediction mechanism and study the performance of different ABR algorithms at Appendix A.

VI. Subjective Evaluation: User Study

Modeling the QoE of users has always proved challenging for the computer networks and multimedia community [11], [43], [44]. In fact, there is no clear guidance yet on how to tie measurable metrics to user's perception, especially in the context of 360-degree tiled video streaming. Peak Signal-to-Noise Ratio (PSNR), which compares bits of the original image with the played video image is often inadequate. The approach we have used in evaluating the QoE of Eq. 13 is linear sum of user perceived bitrates, rebuffering, and smoothness for video segments. All related techniques use a very similar heuristic model (e.g. [15], [25], [29], [45]. This model provides all lower-level metrics that might influence viewing experiences, but does not directly evaluate user experience.

Thus, for the sake of a complete evaluation we also perform subjective user study. We follow the recommendations for *subjective video quality assessment methods for multimedia applications* in [46]. We use two recommended methods, one called Absolute Category Rating (ACR) and the other called Pair Comparison (PC). We recruit a total of 13 volunteers[4] for our subjective user study. Each volunteer views ten videos with different methods (*Mosaic*, VP_Only, VP_Plus, Flare) for ACR and ten videos with PC methods. The viewing of is randomized. In total, 40 video sessions are presented to each volunteer. Among the volunteers, 62% of the subjects are male and 38% are female. The distribution of age groups of 20-30, 30-40, 40-50, 50-60 year old is 38%, 31%, 23%, and 8% respectively.

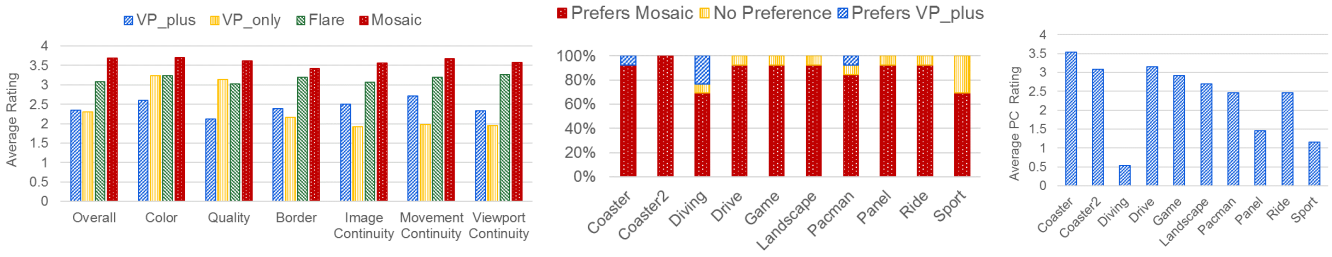
A. Absolute Category Rating (ACR)

We use a recommended method, called Absolute category rating (ACR) to evaluate different algorithms and to rank the video system performance and quality levels [46]. The subject is presented with test sequences, one at a time and rates independently on a category. According to recommendations in [46], the subject is advised to ask if there are any questions about procedure or the meaning of instructions and to observe carefully the entire video sequence before making judgment. Using the five-level scale for rating (5-Excellent, 4-Good, 3-Fair, 2-Poor, 1-Bad), the subject is asked to evaluate overall qualities as well as additional rating components as follows: Overall quality, Image colour, Image quality, Image borders, Image continuity, and Movement continuity.

Unless all tiles are downloaded in a same quality level, the user could view a scene with different quality levels. In the subjective study, we evaluate this aspect – how the user perceives the discontinuities in the qualities within the

[4]According to the ITU-T recommendation, *"four is the absolute minimum number of subjects for statistical reason, while there is rarely any point in going beyond 40." [46]*

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.



(a) Absolute Category Rating (b) Pair Comparison (c) Average Pair Comparison Rating

Fig. 12. Result of subjective user studies using three different techniques. In (a), we show Absolute Category Rating, which is the average rating of *Mosaic* and other techniques for overall quality and six additional rating components. *Mosaic* is able to achieve higher overall quality rating compared to other techniques. In (b), we show Pair Comparison, where the volunteer is asked to compare video streaming sequence shown in *Mosaic* with VP_Plus. In (c), which shows the average Pair Comparison to quantify the average value by which *Mosaic* is preferred by the user across all videos.

viewport. The subject is asked if he/she notices any discontinuity within a viewport and asked to rate in five-level scales (5-Imperceptible, 4-Perceptible but not annoying, 3-Slightly annoying, 2-Annoying, and 1-Very Annoying). The higher the rating, the better image continuity within the viewport is observed.

B. Pair Comparison (PC)

While ACR is easy to implement, the subject may have different interpretation on rating scales. We extend the study using Pair Comparison (PC) method where test videos are presented in pairs and subject is asked to express if he/she prefers the first or second sequence. This is efficient when evaluating if a specific method improves the user quality of experience compared to the other method. We compare the pair of 10 videos using *Mosaic* versus VP_Plus. The subject is also asked to compare the QoE *Mosaic* with VP_Plus in PC rating by giving a rating between -5 and 5, where a higher rating denotes that *Mosaic* has better quality than VP_Plus.

C. Results

Fig. 11 shows frame captures of the video session using two different techniques presented to the volunteers. Fig. 11c and 11a show that the frames using VP_Plus have poor quality at some parts of the viewport. Fig. 11b and 11d show that the video is presented with the highest quality in entire viewport in *Mosaic*.

Fig. 12a shows the subjective qualities in terms of overall quality and six additional rating components. Each bar represents the average rating of a technique used in tiled 360-degree video streaming. *Mosaic* exhibits superior performance in overall quality, with average rating value of 3.7. Notably, *Mosaic* has the highest rating in all additional rating components. *Mosaic* has higher overall quality and image color/quality than other techniques. *Mosaic* also has the highest average user perceived bit rates as is seen at the evaluation results with a real video player as well as in simulation (Fig. 6a, Fig. 6e, and Fig. 8). Note that *Mosaic* has highest viewport continuity, which also confirms the our simulation results in Fig. 8.

Fig. 12b shows the user preference of *Mosaic* compared to VP_Plus. Out of a total of ten videos, 87.7% of them get better ratings using *Mosaic*, 8.5% of them receive the same rating, and 3.8% of them get worse ratings using *Mosaic*. In seven out of ten videos, volunteers prefer the *Mosaic* as much as 92% – 100%. The PC rating shows how much the subject prefers *Mosaic* over VP_Plus. Note that higher is better, with 5 being the best possible. The results in Fig. 12c show that overall the user prefers *Mosaic* by an average value of 2.35. In seven out of ten videos, the subjects rate *Mosaic* is better than VP_Plus as much as 2.5–3.6.

VII. Related Work

Adaptive Video Streaming: Recent works on rate adaptation to improve QoE of streaming videos include Festive [11], MPC [29], Pytheas [47], Pensieve [15], and CS2P [13]. Pensieve and CS2P use neural networks and data-driven rate adaptation scheme respectively, while Festive and Pytheas are based on throughput and network-capacity. There is also another line of work such as BBA [12] based on client's buffer capacity and BOLA [14] as a utility maximization problem using Lyapunov optimization techniques. Unlike *Mosaic* these methods do not handle tiling with rate adaptation.

Streaming 360-degree Videos: Several recent papers consider viewport-based adaptive streaming for 360-degree videos. They either utilize run-time spatial splitting of frames into tiles to deliver a subset [6], [7], [9], [18], [48], or utilize adaptive compression rates based on the region-of-interest and network capacity [22]. A recent work, Rubiks [49] maximizes the video quality as a function of the tile bitrate and the user's current field of view.

While a tile-based approach enables reusing the existing streaming ecosystem, it requires viewport prediction to be really effective. A body of recent work has focused on such prediction. The methods proposed vary from Liner Regression (LR) and variations [6], [8] to advanced machine learning [7], [9], [50]. However, relative to *Mosaic* these approaches have various limitations: i) They bring in only the predicted viewport [6], [7], [23] and unable to handle missing tiles. ii) Use of rate adaptation is limited. Some of the schemes bring in viewport tiles at the highest bitrate regardless of the network capacity [7]. The work in [6] evenly allocates the bandwidth to the predicted tiles. Some works employ a simple binary bitrate adaptation – highest quality for predicted viewport and lowest for outside viewport [17], [17], [23]. In [8] regression is used

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

to predict future viewports and a buffer-based approach is used However, a buffer-based approach has limitation in case of tile-based streaming, as higher buffer levels do not necessarily mean that there is no rebuffering/stalls. A more recent study [25] predicts viewport center using simple machine learning techniques and assign rates to tiles by incrementally enlarging the viewing angles from the predicted viewpoint center. However this rate adaptation is sensitive to the viewport prediction error. It also requires server to decide whether to respond to client HTTP request of tiles, which makes it hard to scale for large number of streaming session. *Mosaic* complies with standard DASH and does not require server side modification and can offer longer duration of segment over 90% accuracy.

Also, many of these works including *Mosaic* play a crucial role in streaming VR and/or AR content. For example, FlashBack [51] and Furion [52] are specifically developed for VR ecosystem. While these methods are orthogonal to *Mosaic*, our tile-based rate adaptation and prediction algorithms can significantly improve these methods.

Data driven Quality of Experience of 360-degree Videos: Pano [53] proposes different quality model using 360-degree video specific features. It then formulates variable size tiles and chooses rates from pre-generate look up tables using such features. However, unlike *Mosaic*, it does not run a prediction system at run-time, and thus risks much worse QoE for atypical users. DRL360 [45] uses machine learning to choose tile qualities to fetch. Unlike our work, DRL360 does not support different bitrates for predicted viewport tiles. Instead it settles for downloading all tiles outside the predicted viewport in the lowest quality, while downloading predicted viewport in one quality matching the available network capacity, i.e., the RL agent learns only one rate. This leads to high quality variance within the viewport, similar to VP_Plus and a poor QoE when the network indeed has sufficient bandwidth and also the viewport prediction is inaccurate. *Mosaic* builds on these quality models to provide a streaming system that runs both viewport prediction, and variable quality streaming at run-time. Other data driven approaches [42], [54] apply ML techniques to manage configurable weight parameters of the ABR algorithms.

VIII. Conclusions

We have developed an end-to-end tiled adaptive video streaming framework for 360-degree videos called *Mosaic*. *Mosaic* uses viewport prediction exploiting latest developments in video analysis and deep learning to predict user viewports in advance. The prediction uses saliency and motion maps of the video and user's head tracking data as input. We have used two different techniques for prediction: one based on a CNN followed by an LSTM-based RNN and the other based on a 3DCNN. They both provide superior prediction performance (about 90% accuracy), especially the latter.

The rate adaptation part of *Mosaic* uses a control-theoretic approach and allocates rates for the tiles of the next segment being fetched based on their viewing probabilities. We utilize a representative metric of QoE to achieve higher user perceived video bitrates, less quality variance within viewport and across segments, and less rebuffering.

We implement *Mosaic* using the GPAC MP4Client reference video player. We evaluate *Mosaic* and compare its performance against the current state-of-the-art video streaming solutions – both for regular and 360-degree video streaming. Our evaluations in realistic network settings show that *Mosaic* achieves at least 47–191% better video quality of experience (in terms of the average quality of experience) relative to other state-of-the-art methods. Simulation based evaluation as well as subjective user studies further confirm the superiority of the proposed approach.

Acknowledgment

We thank the anonymous TNSM reviewers and editors for their constructive feedback. This research was partially supported by Intelibs, Inc. The authors acknowledge the time and effort of the volunteer user study participants.

References

[1] "Sandvine the global internet phenomena report," Tech. Rep., 2018.
[2] "Global virtual reality market (hardware and software) and forecast to 2020," http://www.orbisresearch.com/reports/index/global-virtual-reality-market-hardware-and-software-and-forecast-to-2020, February, 2017.
[3] "Samsung 360 round VR camera," https://www.samsung.com/us/business/products/mobile/virtual-reality/360-round-vr-camera/, 2017.
[4] S. Hollister, "Youtube's ready to blow your mind with 360-degree videos," *March 13th*, 2015.
[5] Akamai, "Akamai's: 2018 State of the Internet / Connectivity Report ," 2018.
[6] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *ATC Workshop*, 2016.
[7] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360 video streaming in head-mounted virtual reality," in *NOSSDAV*. ACM, 2017.
[8] L. Xie, Z. Xu, Y. Ban, X. Zhang, and Z. Guo, "360ProbDASH: Improving qoe of 360 video streaming using tile-based HTTP adaptive streaming," in *Multimedia*. ACM, 2017.
[9] Y. Bao, H. Wu, T. Zhang, A. A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *Big Data*. IEEE, 2016.
[10] S. Akhshabi, A. C. Begen, and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http," in *Multimedia systems*, 2011.
[11] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," in *CoNEXT*, 2012.
[12] 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," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, 2015.
[13] 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 *SIGCOMM*. ACM, 2016.
[14] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *INFOCOM*. IEEE, 2016.
[15] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *SIGCOMM*, 2017.
[16] S. K. Park, A. Bhattacharya, M. Dasari, and S. R. Das, "Understanding user perceived video quality using multipath TCP over wireless network," in *IEEE 39th Sarnoff Symposium*. IEEE, 2018.
[17] V. R. Gaddam, M. Riegler, R. Eg, C. Griwodz, and P. Halvorsen, "Tiling in interactive panoramic video: Approaches and evaluation," *IEEE Trans. on Multimedia*, vol. 18, no. 9, 2016.
[18] M. Hosseini and V. Swaminathan, "Adaptive 360 VR video streaming: Divide and conquer," in *IEEE International Symposium on Multimedia (ISM)*, 2016.

[19] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *2017 IEEE international conference on communications (ICC)*.
[20] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Optimal set of 360-degree videos for viewport-adaptive streaming," *ACM Multimedia*, 2017.
[21] S. Petrangeli, F. De Turck, V. Swaminathan, and M. Hosseini, "Improving virtual reality streaming using http/2," in *MMSys*. ACM, 2017.
[22] H. Xie and X. Zhang, "Poi360: Panoramic mobile video telephony over LTE cellular networks," in *CoNext 2017*, 2017.
[23] M. Graf, C. Timmerer, and C. Mueller, "Towards bandwidth efficient adaptive streaming of omnidirectional video over http: Design, implementation, and evaluation," in *MMSys*. ACM, 2017.
[24] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based hevc video for head mounted displays," in *Multimedia (ISM)*. IEEE, 2016.
[25] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proceedings of MobiCom*. ACM, 2018.
[26] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
[27] O. A. Niamut, E. Thomas, L. D'Acunto, C. Concolato, F. Denoual, and S. Y. Lim, "MPEG DASH SRD: spatial relationship description," in *MMSys*. ACM, 2016.
[28] S. Park, A. Bhattacharya, Z. Yang, M. Dasari, S. R. Das, and D. Samaras, "Advancing user quality of experience in 360-degree video streaming," in *2019 IFIP Networking Conference (IFIP Networking)*. IEEE, 2019.
[29] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," *ACM SIGCOMM Computer Communication Review*, 2015.
[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.
[31] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, "Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
[32] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE MultiMedia*, no. 4, 2011.
[33] J. Le Feuvre, C. Concolato, J.-C. Dufourd, R. Bouqueau, and J.-C. Moissinac, "Experimenting with multimedia advances using gpac," in *Multimedia*. ACM, 2011.
[34] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360 video viewing dataset in head-mounted virtual reality," in *MMSys*. ACM, 2017.
[35] "Kvazaar," https://github.com/ultravideo/kvazaar, 2017.
[36] "GPAC," https://github.com/gpac/gpac, 2017.
[37] "PyTorch," http://pytorch.org/, 2018.
[38] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters*, vol. 20, no. 11, 2016.
[39] Q. Liang, P. Shenoy, and D. Irwin, "AI on the Edge: Rethinking AI-based IoT applications using specialized Edge architectures," *arXiv preprint arXiv:2003.12488*, 2020.
[40] "Federal communications commission. 2016. raw data - measuring broadband america. (2016)." https://www.fcc.gov/reports-research/reports/measuring-broadband-america/raw-data-measuring-broadband-america-2016, Tech. Rep.
[41] 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 Multimedia Systems Conference*, 2013.
[42] S. Park, M. Hoai, A. Bhattacharya, and S. R. Das, "Adaptive streaming of 360-degree videos with reinforcement learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2021, pp. 1839–1848.
[43] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011.
[44] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, 2014.
[45] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "DRL360: 360-degree video streaming with deep reinforcement learning," in *Proceedings of IEEE Infocom*, 2019.

TABLE III
AVERAGE PREDICTION ACCURACY FOR DIFFERENT NUMBER OF FUTURE FRAMES

LSTM	Accuracy	Precision	Recall	F1
15 frames	92.06	78.13	71.86	0.75
30 frames	90.84	73.73	69.21	0.71
45 frames	89.76	70.14	66.32	0.67
60 frames	88.58	66.30	62.87	0.64
3DCNN				
15 frames	93.37	87.59	80.36	0.84
30 frames	93.06	85.71	80.12	0.83
45 frames	90.47	78.90	73.95	0.76
60 frames	88.44	72.67	69.13	0.71
Linear Regression				
15 frames	82.16	48.53	55.93	0.52
30 frames	81.22	49.02	53.36	0.51
45 frames	80.25	49.31	56.03	0.50
60 frames	79.34	49.52	48.86	0.49

[46] P. ITU-T RECOMMENDATION, "Subjective video quality assessment methods for multimedia applications," *ITU*, 1999.
[47] J. Jiang, S. Sun, V. Sekar, and H. Zhang, "Pytheas: Enabling data-driven quality of experience optimization using group-based exploration-exploitation." in *NSDI*, 2017.
[48] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using MPEG-DASH," in *MMSys*. ACM, 2016.
[49] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," in *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, 2018.
[50] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-prediction-based multicast for 360-degree video transmissions," in *SECON*, 2017.
[51] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Mobisys*, 2016.
[52] Z. Lai, Y. C. Hu, Y. Cui, L. Sun, and N. Dai, "Furion: Engineering high-quality immersive virtual reality on today's mobile devices," in *Mobicom*, 2017.
[53] Y. Guan, C. Zheng, X. Zhang, Z. Guo, and J. Jiang, "Pano: Optimizing 360 video streaming with a better understanding of quality perception," in *SIGCOMM*, 2019.
[54] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang, "Oboe: auto-tuning video abr algorithms to network conditions," in *SIGCOMM*, 2018.

APPENDIX

A. Subsystem Performance Evaluation

In tiled 360-degree video streaming, viewport prediction and rate adaptation algorithm are two key sub-components. In this section, we separately evaluate the performance of each component of the system. In addition to Table II, we provide additional results to evaluate accuracy of viewport prediction under various prediction window sizes in units of number of future frames in Table III.

We also evaluate the ABR algorithms independently of the viewport prediction. To do this, we fix the viewport prediction mechanism and study the performance of different ABR algorithms. Fig. 13 shows the experimental results.

Mosaic has the highest user perceived quality levels, least smoothness variance over viewport and across segments, and highest QoE compared to seven other methods. Note that Flare with LSTM (Flare-L) or Flare with 3DCNN (Flare-3) does not achieve better QoE than Flare. It is because LSTM and 3DCNN give probabilities of tiles likely to be viewed but Flare's ABR algorithm only uses a single predicted viewport center, then decreases the tiles' quality levels further away from the center (up to 4 different levels). As a result, it

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.

does not handle well the case where multiple viewport centers are predicted with similar probabilities. We believe that each method focuses on designing the ABR algorithm to work best with the specific viewport prediction method used. *Mosaic*'s viewport prediction and ABR algorithm both outperform other methods.

B. Segment Duration and Efficiency

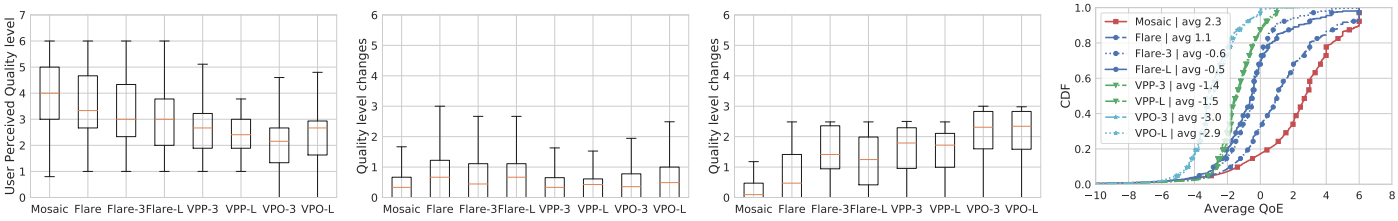
In adaptive video streaming, encoding efficiency reduces when the segment duration becomes short because the number of I-Frame increases. (An I-Frame does not require other video frames in the segment to decode; it is the first frame of each segment.) On the other hand, larger segment duration is not always the best because it is less adaptive to the network variations.

In tiled 360-degree video streaming viewport prediction, the accuracy of prediction drops with a large segment duration. On the other hand, having a shorter segment duration does not necessarily guarantee a higher QoE. According to our study, short segment duration (1 sec) has a higher ratio of HTTP request overhead than 2 second. As a result, the player selects lower quality to avoid frequent rebuffering. In turn it degrades the QoE. Fig. **??** shows the average QoE of different methods when streaming videos of segment duration (1,2,3 and 4 seconds). Using *Mosaic*, in spite of the better viewport prediction accuracy under 1 sec segment, the QoE of video with 1 sec segment is worse than the QoE of 2 sec segment under the same network condition. And QoE is worse at 3 second and 4 second segment duration than that of 2 second segment. *Mosaic* outperforms other methods in all segment duration. *Mosaic* in 2 seconds outperforms the other 15 comparing cases.

C. Evaluation with another Quality Function

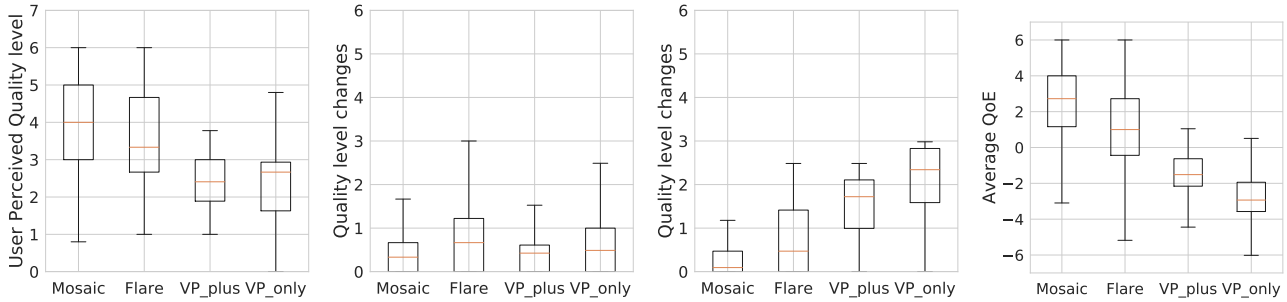
We evaluate the system when using the index quality function ($q(R)$ = quality level of the encoded video (Section II-B). Flare also uses the index quality function [25]. Note that we encode the video in six different quality levels. 1 is the lowest quality level, and 6 is the highest quality level. Fig. 14d shows *Mosaic* achieves the highest QoE. *Mosaic* also outperforms the other methods in terms of user perceived quality level (Fig. 14a), smoothness with viewport (Fig. 14c) and across segments (Fig. 14b).

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.



(a) User perceived quality levels. (b) Variance across segment. (c) Variance within viewport. (d) User perceived QoE.

Fig. 13. Experimental results of ABR Algorithms with different Viewport Prediction methods. Mosaic (with 3DCNN), Flare (with LR), Flare-3 (Flare with 3DCNN), Flare-L (Flare with LSTM), VPP-3 (VP_plus with 3DCNN), VPP-L (VP_plus with LSTM), VPO-3 (VP_only with 3DCNN), and VPO-L (VP_only with LSTM)



(a) User perceived quality levels. (b) Smoothness variance across segment. (c) Smoothness variance within VP. (d) User perceived QoE.

Fig. 14. Index Quality Function. We encode videos in 6 different quality levels. 1 indicates the lowest and 6 indicates the highest video quality level. Video Quality Level is used at the index quality function, $q(R_j)$= index of the quality levels.

This paper has been accepted for publication at IEEE Transactions on Network Service and Management. The authentic version of this paper is available on IEEEXplore at https://doi.org/10.1109/TNSM.2021.3053183.