Advancing User Quality of Experience in 360-degree Video Streaming

Sohee Park, Arani Bhattacharya, Zhibo Yang, Mallesham Dasari, Samir R. Das, Dimitris Samaras
Stony Brook University
{soheekim, arbhattachar, zhibyang, mdasari, 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 four other streaming techniques – two for conventional adaptive video streaming and two for 360-degree tile-based video streaming. *Mosaic* outperforms the best of the competition by as much as 50% in terms of median video quality.**

***Keywords*—360-degree video streaming, adaptive video streaming, MPEG-DASH, Convolutional Neural Network (CNN), 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*. A viewport is about $90^\circ - 120^\circ$ 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.

ISBN 978-3-903176-16-4 © 2019 IFIP

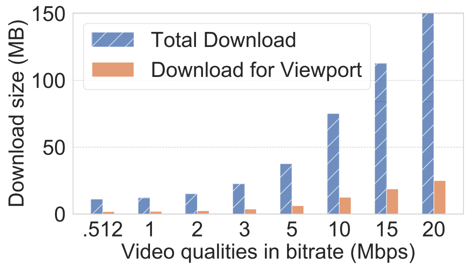


Fig. 1. Total downloaded size vs. download size for only the viewport watched (in MB) for an actual 360-degree video streaming experiment with video encoded in different bit rates. The video is about 1 min long. Further details about the experiments are discussed in Section III.

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, 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]–[15]. With 360-degree video a new dimension is

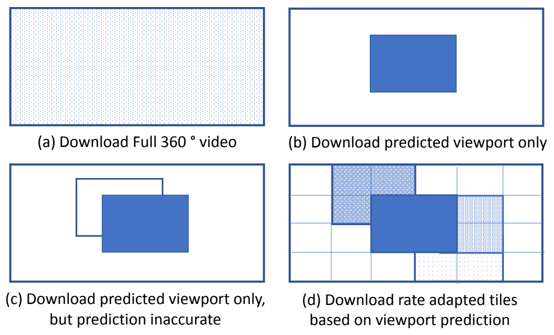

(a) Download Full 360 ° video
(b) Download predicted viewport only
(c) Download predicted viewport only, but prediction inaccurate
(d) Download rate adapted tiles based on viewport prediction

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

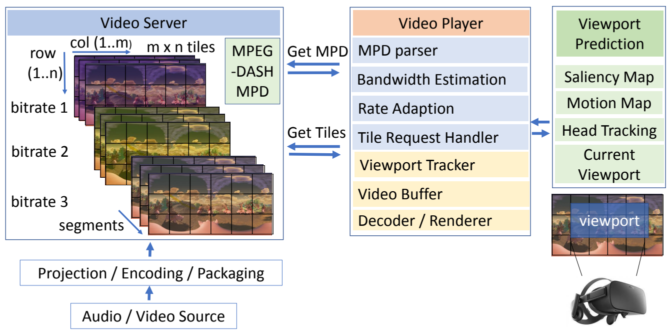


Fig. 3. Overview of *Mosaic* design: a tile-based adaptive 360-degree video Streaming using user viewport prediction.

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 [16]–[23], 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 [24] 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) [25] 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 V.

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) [26]. 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.

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. We discuss related works in Section V and conclude in Section VI.

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 [17], [26]. Across time, each tile is split into multiple chunks of fixed duration, called *segments*. We evaluate tiling/segmentation overhead against 4sec chunk 1x1 tile encoding. We find 4x6 encoding is optimal considering the viewport horizontal and vertical degree and number of HTTP request of tiles. While 1 sec chunk is used in previous studies [6], [24], our evaluation shows that our encoding overhead reduces by 50% using 2sec chunks. 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 [27].

We model the quality of experience metric of an individual segment, Q, as consisting of two distinct components. The first component is the average bitrate perceived by the user. The second component is the quality penalty when tiles are missed. Note that higher bitrate improves the quality, whereas missed tiles hurt the quality. The quality of the segment is modeled as a linear combination of the bitrate perceived and tiles missed.

Asssume 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_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, and P_{Vi} is probability that viewport V_i is actually viewed. Since the viewport spans over some tiles, this in turn defines P_j is probability that tile T_j is actually viewed. 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 user perceived video bitrate as:

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

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.

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 QoE is modeled as:

$$Q = \gamma_1 P_{Vi} B_i - \gamma_2 S_i, \tag{3}$$

where γ_1 and γ_2 are weights modeling the relative importance of tiles seen vs. unseen. 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 Q is maximized, subject to the estimated network capacity C. So the optimization problem to be solved is:

$$\text{Maximize } Q \text{ subject to } \sum_{j=1}^{N} D_j(1 - L_j) < C. \tag{4}$$

Similar to [15], [27], 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.

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. We model the optimization problem in Equation 4 as a *multi-choice knapsack problem* and solve it using a greedy algorithm.

Algorithm 1 Tiled Adaptive Video Streaming

```
1: Initialize
2: for $k \leftarrow 1$ to $M$ do /* $M$ segments in video*/
3:     Estimate current capacity $C$
4:     Estimate tile probabilities $P_j, \forall j = 1 \ldots N$
5:     Estimate viewport probabilities $P_{Vi}, \forall i = 1 \ldots V$
6:     $[R_1 \ldots R_N]$ = SelectRates ($[P_{V_1}, \ldots, P_{V_V}]$, $[P_1 \ldots, P_N]$, $C$, $[D_1, \ldots, D_N]$)
7:     Download the tiles in segment $k$ with video rates $[R_j]$
```

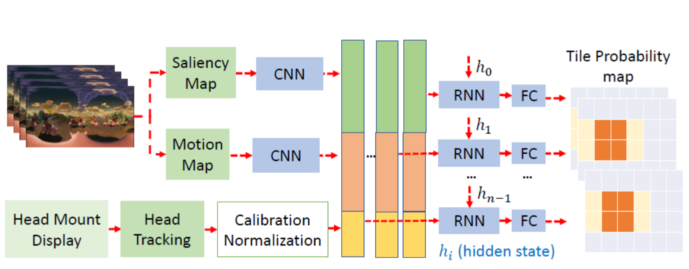
Algorithm 2 *SelectRates*: Viewport Based Rate Adaptation

```
    Input: viewport probabilities $P_{Vi}$, tile probabilities $P_j$, capacity estimated $C$, tile size $D_j$
    Output: $Rates$, bitrates selected for tiles
1: $max \leftarrow -\infty$
2: $Rates \leftarrow \{0, 0, ..0\}$ /*initial condition */
3: $isSelected \leftarrow$ False
4: while $Rates$ is updated do
5:     $R_{tmp} \leftarrow Rates$
6:     for $i \leftarrow 1$ to $V$ do /* viewport index*/
7:         $R \leftarrow R_{tmp}$
8:         for $j \leftarrow 1$ to $N$ do /* tile index*/
9:             if $O_{i,j} > 0$ then
10:                $R_j \leftarrow \min(R_j + 1, R_{max})$
11:        if $\sum_{j=1}^{N} D_j(1 - L_j) > C$ & $isSelected$ then
12:            continue; /* too aggressive.*/
13:        else
14:            $Q \leftarrow \gamma_1 P_{Vi} B_i - \gamma_2 S_i$
15:            if $Q > max$ then
16:                $isSelected \leftarrow$ True
17:                $max \leftarrow Q$
18:                $Rates \leftarrow R$ /*update optimal rates*/
19: return $Rates$
```

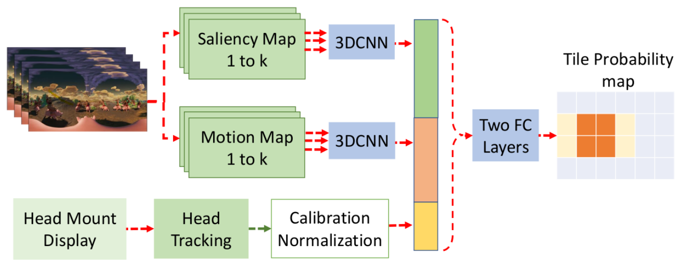
C. Greedy Algorithm

Algorithm 1 selects tiles and their bit rates for each segment to be downloaded. We use the network capacity C observed while downloading the previous segment as a proxy for the current network capacity. We acknowledge that more sophisticated network capacity estimation techniques may possibly provide better results. The viewport and tile probabilities P_{V_i} and 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, P_{V_i}'s, P_i's and D_j's as inputs to determine the selection and rates for each tile T_j.

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



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



(b) Architecture of 3DCNN based prediction.

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

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 Q (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.

D. Viewport Prediction

We develop two different viewport prediction mechanisms. The output of the viewport prediction system is the probabilities of different viewports indicating how likely the user is to view the viewport. 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.

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

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) [28] 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 [29] 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,

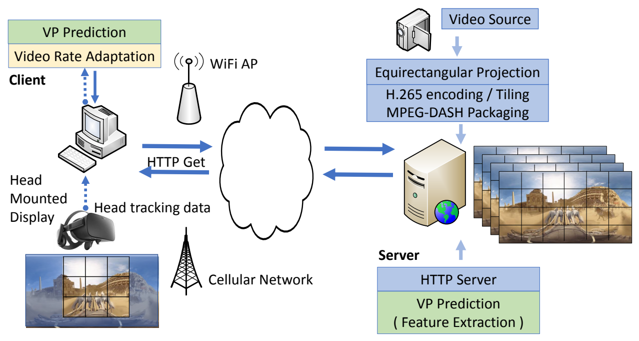


Fig. 5. Evaluation setup.

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) [28] that captures long term dependencies among the frames.

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 [30], 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 [25] 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 offline 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 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 [31] on a Linux platform (Ubuntu 16.04). The player is implemented based on an open-source tile-based adaptive streaming video player, MP4Client [32]. 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) [33]. 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 8 different bit rates – 512 Kbps, 1 Mbps, 2 Mbps, 3 Mbps, 5 Mbps, 10 Mbps, 15 Mbps and 20 Mbps. We use the open source HEVC encoder `kvazaar` [34] for the transcoding. HEVC 'motion constrained'[2] 6×4 tiling is used in the transcoding. Using the GPAC MP4Box [35], we package the HEVC encoded videos in MP4 containers and generate the necessary MPD (Media Presentation Description) including SRD (Spatial Relational Description) [26]. 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 [36] and train the network using the saliency map, motion map, and head tracking dataset [33]. 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 viewport probabilities of next 30 frames (assuming 30 frames/sec framerate).

To implement the client, we extend the MP4Client [35] 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 bandwidth estimation, 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

[1]We have experimented with other segment lengths. But 2 sec segments work well.

[2]This signifies that the decoding and rendering can be done independently on the tiles [22].

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 [37]. A number of studies have evaluated the system over emulated network using public throughput traces [15], [24], [27]. 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:

a) ***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, $q(0) = 0$. To include the contribution of rebuffering periods (stalls) during playback, we assume that these periods are counted as contributing zero rate; thus $q(0) = 0$ during these periods for the entire viewport. Note that we use linear quality function of video bitrate. We plan to evaluate this metric with subjective evaluation. 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, including stalls for rebuffering. B^i, the user perceived quality during the i^{th} unit interval during playback is given by,

$$B^i = \sum_{j=1}^{N} q(R_j) O_{i,j}, \tag{5}$$

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.

b) ***Quality Variation Within Viewport:*** It is possible that the tile quality varies 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. To evaluate this we define V_t, the quality variation among tiles within a viewport as standard deviation of weighted qualities of tiles that overlaps the viewport of the segment during playback.

$$V_t = \frac{1}{M} \sum_{k=1}^{M} \text{StdDev}\left[q(R_j) O_{k,j} \mid j = 1, \ldots, N\right] \tag{6}$$

where $O_{k,j}$ is overlapping ratio of viewport of k^{th} segment during playback and tile T_j and StdDev is standard deviation of weighted qualities of the tiles.

c) ***Quality Variation Across Segments:*** Quality variation across segments during playback also impacts QoE. Similar to a metric used for regular adaptive video streaming [27], we use V_s to quantify quality change at each segment transition, averaged over all such transitions. If there are M segments

$$V_s = \frac{1}{M-1} \sum_{i=2}^{M} |B^i - B^{i-1}| \tag{7}$$

The above formulation tacitly assumes that there are no rebuffering stalls. In case of any stall, any quality change between the segments before and after the stall is not counted.

d) ***Miss Ratio:*** Miss ratio evaluates the fraction of the viewport that user sees as missing during playback (i.e., relevant tiles are not downloaded at the time of playing). The miss ratio M^i of the i^{th} segment is defined as

$$M^i = \sum_{j=1}^{N} O_{i,j} L_j. \tag{8}$$

$L_j = 1$, if tile T_j is missing during the playback interval.

e) ***Rebuffering:*** While we run rate adaptation within the estimated 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. The rebuffering duration is an important QoE metric.

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.

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 two variations of tiled adaptive video streaming algorithms [7], [22]. In our knowledge these are two most recent works closest to *Mosaic*. The four 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

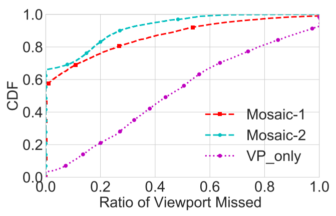


Fig. 6. Miss ratios over WiFi.

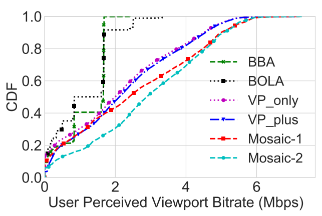


Fig. 7. Video bitrate over WiFi.

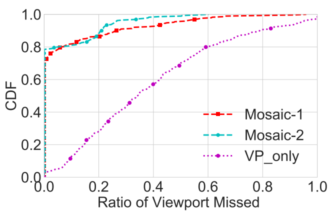


Fig. 8. Miss ratios over LTE traces.

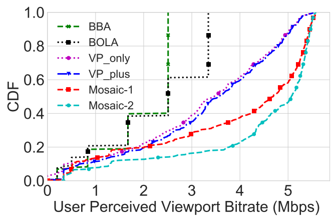


Fig. 9. Video bitrate over LTE traces.

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

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 available 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 and all other tiles in the lowest quality [22].[3]

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 in two categories given the available ground truth head tracking data [33] 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.

TABLE II shows the prediction accuracy, precision, recall and F1-score of the LSTM and 3DCNN-based predictions. Note that as expected the 3DCNN-based method performs somewhat better than LSTM in general. 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.

[3]The evaluation presented in [22] 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.

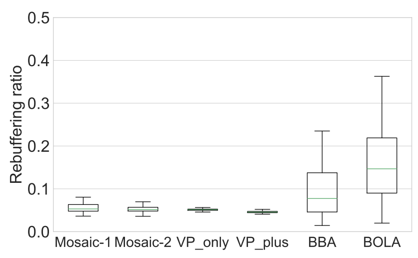
While TABLE II uses only the video and head tracking data set, Fig. 6 shows the the CDF of the miss ratio with actual streaming video in action over WiFi. Fig. 8 shows results using the 4G/LTE dataset. Note that the miss ratio for VP_Plus, BBA and BOLA is always zero, as these techniques bring in the entire frame always. They are thus not shown in the plots. The miss ratios of the two *Mosaic* techniques are substantially better than VP_Only. In WiFi experiment, with *Mosaic* about 20% of the viewport is missed at most 25% of the times and about half of the viewport is missed at most 10% of the times. Average miss ratio is 8%. Using the 4G/LTE dataset, average miss ratio is 4.5%. The actual miss ratio depends on the prediction technique used and the network condition.

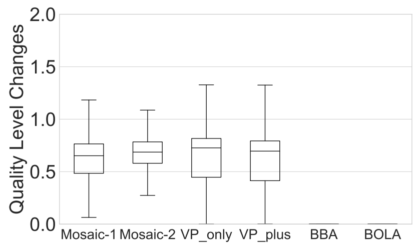
The prediction computation is efficient. Excluding training of the neural networks which can be done offline on the server, the prediction takes about 0.6ms for 3DCNN, 12.7ms for LSTM on a CPU (Intel Xeon E5-1650 v3 Haswell-EP 3.5GHz). The speed improves for LSTM when a GPU is used. On Nvidia Titan X GPU, it takes 0.7ms for 3DCNN, 2.8ms for LSTM. This is fast enough for a seamless video experience. We have not yet evaluated the prediction performance on commodity mobile systems. While similar performance may be hard to achieve in a straightforward fashion on mobile systems, we expect to gain from ongoing innovations in deep learning on mobile devices [38].

C. Results for Streaming Performance

Fig. 7 and Fig. 9 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 median user perceived bitrates of 2.4 and 2.9 Mbps over WiFi, and 4.7 and 5.1 Mbps using the 4G/LTE dataset respectively. This is in comparison to VP_Only and VP_Plus which offer in the median about 2 and 2.1 Mbps over WiFi and 3.4 and 3.5 Mbps over the 4G/LTE emulated network, respectively. Thus, *Mosaic* with 3DCNN offers almost 50% improvement in quality at the median versus the state-of-the-art. As expected, BBA and BOLA both perform somewhat poorly as they download the entire panoramic frame.

Fig. 10 and 11 plot several other metrics – rebuffering ratio (fraction of playback time spent in stalls) for 1-minute video playback, quality level variation within a viewport and across the segments as explained at Section III-C. Note that


(a) Rebuffering.


(b) Quality smoothness within viewport.

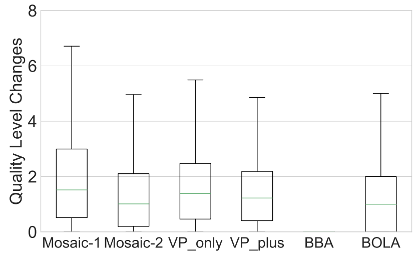
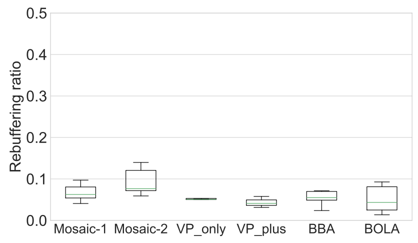
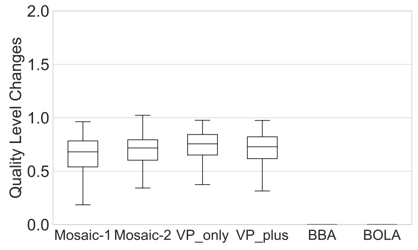

(c) Quality smoothness across segments.

Fig. 10. Rebuffering and quality variations with viewport and across segments for different methods over WiFi network with bandwidth shaping of maximum 10 Mbps using Linux *tc* utility.


(a) Rebuffering.


(b) Quality smoothness within viewport.

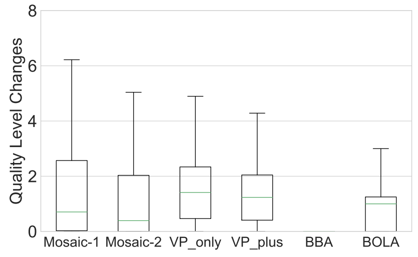

(c) Quality smoothness across segments.

Fig. 11. Rebuffering and quality variations within viewport and across segments for different streaming methods over emulated network with LTE traces.

rebuffering performance of *Mosaic* is among the best with either LSTM and 3DCNN-based predictions over WiFi. Over the 4G/LTE dataset, it is similar with other techniques. For quality variation within viewport, the spatial quality changes, again *Mosaic* with LSTM and 3DCNN are similar to VP_Only and VP_Plus. Note, however, BBA and BOLA do not have any variation as they do not use tiling but bring in the entire frame always. For variation across segments we have similar observations. Here, however, BBA performs better (almost negligible variation) due to a conservative rate control.

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.

Overall, *Mosaic* (specifically with 3DCNN) provides a significantly better video quality during playback with about similar amount of rebuffering stalls relative to other state-of-the-art 360-degree video streaming.

V. Related Work

Adaptive Video Streaming: Recent works on rate adaptation to improve QoE of streaming videos include Festive [11], MPC [27], Pytheas [39], 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], [17], [40], or utilize adaptive compression rates based on the region-of-interest and network capacity [21]. A recent work, Rubiks [41] maximizes the video quality as a function of the tile bitrate and the user's current field of view. However, Rubiks does not adapt to changes in the viewport since it does not utilize any viewport prediction.

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], [42]. However, relative to *Mosaic* these approaches have various limitations: i) They bring in only the predicted viewport [6], [7], [22] 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 [16], [16], [22]. In [8] regression is used 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 [24] with end to end system of viewport prediction and rate adaptation uses server ticks to decide whether to respond to client HTTP request. *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 [43] and Furion [44] 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.

VI. 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 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 50% better video quality of experience (in terms of the median of per-frame video bitrate) relative to other state-of-the-art methods.

As future work, we plan to extend our work to 6-DOF VR (videos with six degrees of freedom), where the problem of prediction is more challenging. We also plan to further validate our work using actual user studies and larger datasets.

Acknowledgment

This research was partially supported by Intelibs, Inc. and by the MSIT (Ministry of Science and ICT), Korea, under ICTCCP Program (IITP-2019-2011-1-00783).

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's: Q1 2017 State of the Internet / Connectivity Report ."

[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, pp. 315–323.

[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] 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.

[17] M. Hosseini and V. Swaminathan, "Adaptive 360 vr video streaming: Divide and conquer," in *Multimedia (ISM)*. IEEE, 2016.

[18] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *ICC*. IEEE, 2017.

[19] X. Corbillon, A. Devlic, G. Simon, and J. Chakareski, "Optimal set of 360-degree videos for viewport-adaptive streaming," *ACM Multimedia*, 2017.

[20] S. Petrangeli, F. De Turck, V. Swaminathan, and M. Hosseini, "Improving virtual reality streaming using http/2," in *MMSys*. ACM, 2017.

[21] H. Xie and X. Zhang, "Poi360: Panoramic mobile video telephony over LTE cellular networks," in *CoNext 2017*, 2017.

[22] 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.

[23] R. Skupin, Y. Sanchez, C. Hellge, and T. Schierl, "Tile based hevc video for head mounted displays," in *Multimedia (ISM)*. IEEE, 2016.

[24] 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.

[25] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*. IEEE, 2017.

[26] 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, p. 5.

[27] 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*, vol. 45, no. 4, 2015.

[28] C. Olah, "Understanding lstm networks," 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2016.

[30] M. Zolfaghari, G. L. Oliveira, N. Sedaghat, and T. Brox, "Chained multi-stream networks exploiting pose, motion, and appearance for action classification and detection." IEEE, 2017.

[31] I. Sodagar, "The mpeg-dash standard for multimedia streaming over the internet," *IEEE MultiMedia*, no. 4, 2011.

[32] J. Le Feuvre, C. Concolato, J.-C. Dufourd, R. Bouqueau, and J.-C. Moissinac, "Experimenting with multimedia advances using gpac," in *Multimedia*. ACM, 2011.

[33] 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.

[34] "Kvazaar," https://github.com/ultravideo/kvazaar, 2017.

[35] "GPAC," https://github.com/gpac/gpac, 2017.

[36] "PyTorch," http://pytorch.org/, 2018.

[37] 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, pp. 2177–2180, 2016.

[38] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "Deepx: A software accelerator for low-power deep learning inference on mobile devices," in *Proc. 15th IPSN*, 2016.

[39] 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.

[40] J. Le Feuvre and C. Concolato, "Tiled-based adaptive streaming using mpeg-dash," in *MMSys*. ACM, 2016, p. 41.

[41] J. He, M. A. Qureshi, L. Qiu, J. Li, F. Li, and L. Han, "Rubiks: Practical 360-degree streaming for smartphones," ser. MobiSys '18.

[42] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-prediction-based multicast for 360-degree video transmissions," in *SECON*, 2017.

[43] K. Boos, D. Chu, and E. Cuervo, "Flashback: Immersive virtual reality on mobile devices via rendering memoization," in *Mobisys*, 2016.

[44] 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.