LiveDeep: Online Viewport Prediction for Live Virtual Reality Streaming Using Lifelong Deep Learning

Xianglong Feng* Rutgers University
Yao Liu† SUNY Binghamton
Sheng Wei‡ Rutgers University

ABSTRACT

Live virtual reality (VR) streaming has become a popular and trending video application in the consumer market providing users with 360-degree, immersive viewing experiences. To provide premium quality of experience, VR streaming faces unique challenges due to the significantly increased bandwidth consumption. To address the bandwidth challenge, VR video viewport prediction has been proposed as a viable solution, which predicts and streams only the user's viewport of interest with high quality to the VR device. However, most of the existing viewport prediction approaches target only the video-on-demand (VOD) use cases, requiring offline processing of the historical video and/or user data that are not available in the live streaming scenario. In this work, we develop a novel viewport prediction approach for live VR streaming, which only requires video content and user data in the current viewing session. To address the challenges of insufficient training data and real-time processing, we propose a live VR-specific deep learning mechanism, namely *LiveDeep*, to create the online viewport prediction model and conduct real-time inference. *LiveDeep* employs a *hybrid* approach to address the unique challenges in live VR streaming, involving (1) an alternate online data collection, labeling, training, and inference schedule with controlled feedback loop to accommodate for the sparse training data; and (2) a mixture of hybrid neural network models to accommodate for the inaccuracy caused by a single model. We evaluate *LiveDeep* using 48 users and 14 VR videos of various types obtained from a public VR user head movement dataset. The results indicate around 90% prediction accuracy, around 40% bandwidth savings, and premium processing time, which meets the bandwidth and real-time requirements of live VR streaming.

Index Terms: Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality;

1 INTRODUCTION

Recently, virtual reality (VR) video streaming (a.k.a., 360-degree video streaming) has become a popular video application in the consumer video market, with the rapid growth of mobile head mounted display (HMD) devices [19]. Many VR device manufacturers (e.g., HTC [39] and Oculus [28]) have released affordable and practical HMDs to the market. Many video content providers (e.g., YouTube [7], National Geography [35], and CNN [25]) have provided high-definition VR video content for streaming to mobile HMDs over wireless (e.g., WiFi or cellular) networks. More importantly, VR streaming has started its initial deployment for live broadcast events, such as sports games [31] and breaking news [1], providing instant immersive experiences to the end users. Different from traditional video streaming, in VR streaming users have the freedom of selecting the video viewports (i.e., the portion of the video to watch) from a 360-degree sphere using natural head movements, similar to the viewing experience in the physical world.

Given the demands of high resolution and high frame rate in VR streaming to ensure user's quality of experience (QoE), the VR video content is typically huge in size and thus poses significant challenges in the network bandwidth consumption [8, 14]. Even if a single or small number of VR video viewing sessions can be supported by the state-of-the-art high bandwidth networks, the nature of the video streaming services that could involve millions of concurrent viewing sessions would create significant capacity challenges in both the backbone and edge networks. Such challenges would eventually be converted to degraded QoE towards the user end, significantly blocking the wider deployment of premium VR experiences.

The potential solution to the bandwidth challenge of VR streaming leverages the fact that the user can only watch an around 90-degree viewport at any point of time, leaving the rest (more than 80%) of the video content in the 360-degree frame unnecessary to be delivered to the mobile HMD. One example solution is selective streaming [38], which proposes to stream the portion of video that the user is more likely to watch in high resolution, while the rest of the video is delivered in low resolution. However, the implementation of such solution requires accurate prior knowledge about the user's viewport of each individual frame, which is typically hard to obtain since the viewport is completely controlled by the end user. Therefore, the state-of-the-art research focuses on predicting user's viewport using a variety of methods, such as trajectory-based learning and inference based on the individual user's *historical* head movements [3, 4, 17, 23, 26, 27, 29, 30] and heat map analysis of the popular video content/viewport based on multiple users' *historical* viewing behavior [2, 21, 32]. However, notably, both categories of methods require *historical* user and/or video data to build the prediction model. This requirement can be achieved in most of the video-on-demand (VOD) streaming use cases (i.e., pre-recorded video content) but infeasible in live VR streaming (i.e., live-generated video content) where the video content is generated and viewed for the first time. Looking through the video streaming industry, the live streaming service has been a very popular and attractive use case [34]. However, the state-of-the-art viewport prediction techniques have not yet caught up with such demand in the applications.

We develop a novel viewport prediction approach targeting the unaddressed live VR streaming scenario. In our preliminary work-in-progress poster paper [10], we explored the feasibility of using a single convolutional neural network (CNN) model for live viewport prediction and identified the limitations of the simple CNN structure. In this paper, we aim to improve the performance of prediction by employing an alternate and hybrid deep learning approach involving both CNN and long short-term memory (LSTM) models. In particular, we target three specific live VR streaming characteristics, which do not exist in VOD videos and would fail the state-of-the-art VOD-oriented viewport prediction approaches: (1) *Online video generation*: video content is generated on the fly with no trace of previous videos that can be used to train a prediction model; (2) *no historical user data*: we cannot predict the viewport based on this or other users' past head movement trajectories for the current video, as they simply do not exist; and (3) *real-time*: viewport prediction itself

*e-mail: xf56@scarletmail.rutgers.edu
†e-mail: yaoliu@binghamton.edu
‡e-mail: sheng.wei@rutgers.edu

should not incur significant processing delay that would compromise the live streaming latency [40].

Due to the aforementioned live VR streaming characteristics, we cannot adopt the traditional "train-and-then-predict" workflow for viewport prediction. Instead, we explore the possibility of building an online agent to learn the preference of the video content while the user is watching the video and predict the user viewport for the upcoming video segments. Corresponding to the aforementioned characteristics, we develop a novel online viewport prediction mechanism targeting live VR streaming, namely *LiveDeep*, which follows three design principles:

- ***Online***. The prediction model should be generated online to accommodate for the live video content generated on-the-fly and the associated user preference;
- ***Lifelong***. The prediction model should be updated periodically, for the entire video viewing session, to accommodate for the potential changes in either the video content or the user preference; and
- ***Real-time***. The processing delay caused by the viewport prediction algorithm should not compromise the desired frame rate and live streaming latency.

To realize these design principles in the development of *LiveDeep*, we employ CNN for extracting the features in the video content and mapping them to the user preference of viewports. To accommodate for the unique live streaming challenges and the aforementioned principles, we customize the traditional CNN workflow from three aspects. First, following the ***online*** principle, we collect and label the training data at runtime during the video streaming session, as neither the video content nor the user's viewing preference is available before the streaming starts. Second, following the ***lifelong*** principle, we conduct inference (based on the current video segment) and training (based on the previous video segments) in an alternate and iterative manner to continuously update the model during the entire video session. Such *lifelong* learning mechanism provides the opportunity for the deep learning model to accommodate for dramatic changes in either the video scene or the user viewing preference, both of which are very likely to occur due to the highly interactive nature of live VR streaming. Third, following the ***real-time*** principle, we employ a subsampling method to select a small number of representative video frames to run the viewport prediction algorithm, which achieves acceptable processing delay without compromising the prediction accuracy.

In summary, the major contribution of the paper is to develop a novel viewport prediction mechanism for live VR streaming in order to reduce the bandwidth consumption. Given that there is no historical user or video data to build the prediction model, we propose a hybrid deep learning approach following the *online, lifelong, and real-time* principles determined by the unique requirements of live VR streaming. To the best of our knowledge, this is the first deep learning-based viewport prediction mechanism targeting the challenging live VR streaming scenario.

2 Related work

Viewport prediction has become a trending research topic with the increasing popularity of VR streaming applications in the past few years. Many viewport prediction mechanisms have been proposed [2–4,9,17,21,24,26,27,29,30,32,42,43], which can be divided into two categories: (1) prediction approaches for VOD VR streaming that require historical video or user data; and (2) prediction approaches for live VR streaming using user trajectory or video motion in the current viewing session. While the approaches in category (2) can be applied to live VR streaming, the existing approaches rely on strong assumptions about user behavior and video content and thus have their limitations in achieving accurate prediction outcomes.

2.1 Viewport prediction for VOD VR streaming

Most of the existing viewport prediction approaches target the VOD VR streaming scenario, in which the video has already been generated and watched by a large number of users in the past. In this case, sufficient historical user and video data are available to build the viewport prediction model for new users. This line of research originated from saliency detection in images and videos. Borji et al. [5] provided a comprehensive review of the existing saliency detection mechanisms, and a large number of works [6, 13, 15, 18] have been conducted over the years to use either the bottom-up or the top-down method to train the prediction model. For viewport prediction in VOD VR streaming, several works proposed to collect historical user viewport traces for the target video and generate a heat map for predicting the future viewports of new users watching the same video [2, 21, 32]. Several other works [9, 24, 42, 43] employed machine learning to analyze the relationship between the video content and the user viewing behavior.

In summary, this category of work focuses on building a map between the user behavior and the video content using historical user and video data. Although the existing approaches are effective in their specific application scenarios of VOD streaming, they cannot be applied to the live video streaming scenario because the required historical data is unavailable.

2.2 Viewport prediction for live VR streaming

Recently, several viewport prediction approaches that do not rely on historical user or video data have been proposed. For example, user trajectory-based approaches [3, 4, 17, 26, 27, 29, 30] employ real-time head movement data to predict the future viewports, assuming that user's current head moving pattern (i.e., direction and speed) would sustain for a reasonably long period of time (i.e., at least a few seconds). In particular, existing approaches leverage linear regression [29] or LSTM [24] to accomplish the training and prediction tasks. More recently, Feng et al. [12] proposed to detect object motions in live VR video for viewport prediction, given the assumption that users are likely to watch the portions of the VR video that contain moving objects.

Trajectory and motion-based approaches do not require historical user or video data and thus could be applied to live VR streaming. However, both approaches have limitations in terms of prediction accuracy due to the strong assumptions regarding the user's viewports of interest at runtime. First, trajectory-based approaches could be effective only if the user does not dramatically change his/her head movement direction/speed. While this is possible for a short time span (i.e., < 1 second), the head movement direction/speed could hardly hold for a longer period of time (i.e., a few seconds) that is required by the state-of-the-art streaming standard, e.g., the typical segment duration in DASH [37]. Second, motion-detection-based approaches could only work with videos that have clear and easy-to-distinguish foreground and background boundaries due to the limitations of the motion detection algorithm. As a result, it has been shown that the prediction accuracy would dramatically drop under the scenario of moving cameras/background which, unfortunately, is a very common use case in many VR videos especially the live broadcasting targeted by this work.

2.3 Discussion and summary

To summarize, although there have been a large number of viewport prediction approaches developed in the community, they either only work with the VOD scenario where historical user or video data is available, or have limitations in the supported types of videos and users, leaving a significant gap in effectively supporting bandwidth-efficient live VR streaming. In this paper we aim to bridge this gap by developing *LiveDeep*, a novel viewport prediction mechanism targeting the live VR streaming scenario based on online and lifelong

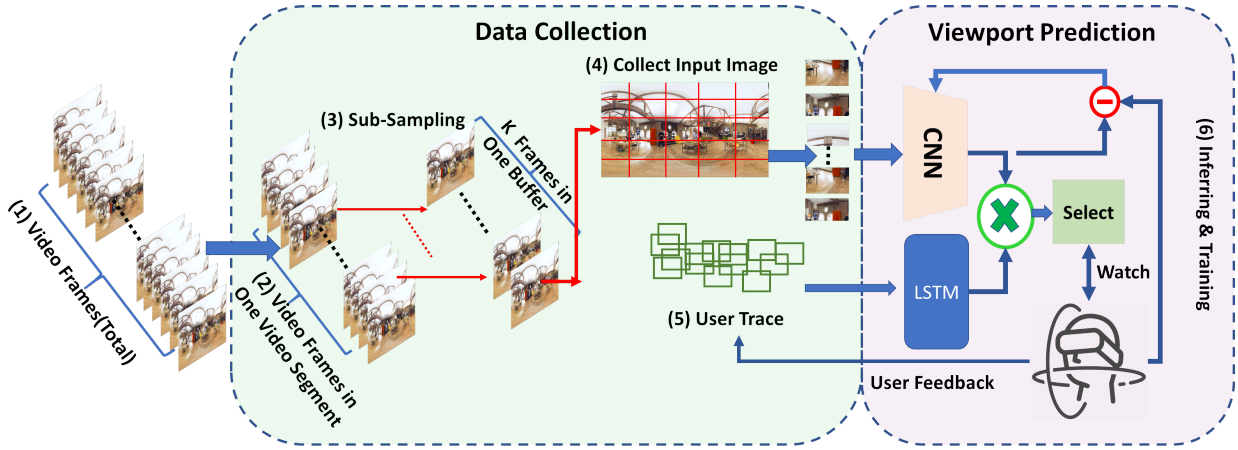

Figure 1: Overall viewport prediction architecture of *LiveDeep*.

deep learning, which supports generic VR videos and users without the limiting assumptions in the state-of-the-art approaches.

3 Our Proposed Method: LiveDeep

3.1 The *LiveDeep* framework

The primary goal of *LiveDeep* is to achieve effective viewport prediction for live VR streaming, following the three design principles (i.e., online, lifelong, and real-time) abstracted from the live streaming requirements. Figure 1 shows the overall architecture of *LiveDeep*. The prediction workflow begins with *data collection* and ends with *viewport prediction*, which is repeated for each segment with the user feedback loop to update the model.

The *data collection* step follows the live VR streaming pipeline, in which we subsample the video frames from each segment for processing due to the consideration of the real-time requirement. Each frame is then uniformly segmented into small sub-images as input to the CNN training. Then, in the *viewport prediction* step, we combine the CNN model with an LSTM model to predict the user viewport for an improved perception of user preference, leveraging the CNN model for analyzing video content and the LSTM model for the perception of the user trajectory. The video segment with the predicted viewport will be watched by the user first and, then, the user feedback (i.e., the actual head orientation data) is collected to calculate the loss value and update the CNN and LSTM models accordingly.

3.2 Training image collection

In a live VR streaming system, a live packager would partition the 360-degree video into small segments following the DASH standard [37]. Leveraging the video segments, we refine our video processing pipeline and treat each video segment as the basic processing unit, within which we train the *LiveDeep* model and predict the user viewport. Considering the potential processing delay introduced by the huge amount of video frames in each segment, as well as the similarity of video content between adjacent frames, we uniformly subsample all the frames in each video segment to k frames. Instead of using a fixed subsampling rate, we select a fixed number (i.e., k) of frames, given that different videos may have different frame rates. Also, we can keep the value of k small to ensure a fast processing speed, since adjacent video frames within in the same segment are often similar. For each of the k selected frames, we uniformly divide the entire frame into $x \times y$ tiles, as shown in Figure 1. Therefore, in each video segment, there are $k \times x \times y$ training images. In this paper, we set $x = y = 5$ and $k = 8$. This enables us to collect 200 images for training and inference per video segment. The resolution of test videos we use is 1280×720, and each tile is resized to 32×32.

In traditional supervised learning, the first step is to prepare the training dataset. The existing viewport prediction approaches targeting VOD VR streaming (i.e., discussed in Section 2.1) could obtain such dataset by collecting historical video and user data. However, in the live VR streaming scenario, the collected training images are not labelled due to the lack of user viewing history. As a result, we must wait for the user feedback (i.e., the actual head movement data) that is available only after the user has watched the corresponding video segment. Then, based on the user feedback, we compare the images with the center of the actual user viewport and label them as either "interested" (i.e., if the center of the actual user viewport is covered by the image) or "not interested" (i.e., if the center of the actual user viewport falls outside the image.)

In the meantime, we obtain the *user trace* from the user head movement data, which is defined as the trajectory of the center of the user viewport in the video frames. To be consistent with the subsampling of frames in each video segment, we subsample k sequential user trace points per segment, which serve as the training dataset for the LSTM model in the user trajectory prediction.

3.3 User viewport prediction with *LiveDeep*

Based on the workflow presented in Figure 1, it is obvious that for the first prediction in the first video segment, the CNN and LSTM models have not been trained yet. More importantly, the $k \times x \times y$ training images are unlabeled and cannot be used to train a model. Our solution is to have the CNN model predict the user viewport using random weights. Also, we do not incorporate the LSTM model into the viewport prediction at this point. Once the user has watched the video segment, the predicted results are compared with the actual user viewport, based on which we calculate the loss value. Then, we use the loss value to update the CNN model. In addition, we use the user trace obtained from the actual user view as the input data for training the LSTM model.

For the next video segment, we first use the updated CNN and LSTM models to predict the user viewport independently. For each model, our prediction results are based on the subsampled k frames considering the real-time requirement. To generate the predicted viewport for the entire video segment, we assume that adjacent frames share similar center of viewport, and thus we use the same prediction result for both frames. Finally, based on the prediction results from the CNN and LSTM models, we select the union set of tiles as the final predicted user viewport.

4 Hybrid CNN/LSTM Model for Live VR Viewport Prediction

In this section we discuss the design and implementation of the hybrid CNN/LSTM model in *LiveDeep* for live viewport prediction.

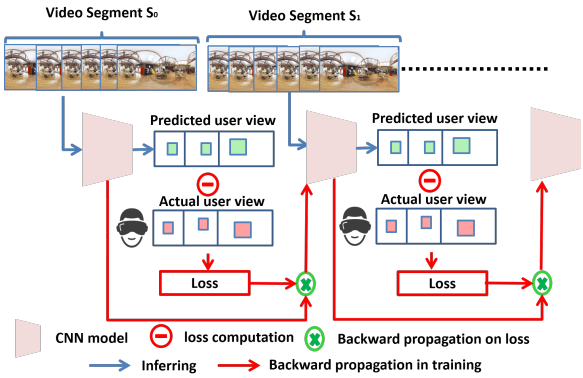

Figure 2: CNN-based viewport prediction workflow.

4.1 CNN-based training and inference

In live VR streaming, the video content is generated on the fly, and the users may have different viewing behavior given the new video content. This may make a pre-trained prediction model fail to achieve accurate prediction results. In *LiveDeep*, targeting the live VR video streaming case, we propose to train new models by employing online and lifelong deep learning. The models are trained at runtime based on new user watching new videos. Figure 2 shows the detailed CNN-based viewport prediction workflow, where the CNN model is not trained until the user starts watching the video. We adopt the classic deep convolutional neural network VGG [36] as the backbone network in *LiveDeep*. Based on the discussion in Section 3.2, we modify the last layer to output only two classes, namely "interested" and "not interested".

4.1.1 Inference and viewport generation

With $k \times x \times y$ unlabeled training images in each video segment, we employ the CNN model to infer the user's viewport of interest. The intuitive idea is to select images that are classified as "interested", and the locations of these selected images in the original frame would indicate the potential viewport of interest. However, we find that almost all of the images are classified as "interested" in this way and, consequently, the whole frame is often chosen as the predicted viewport.

To address the aforementioned problem, we propose a systematic method to determine the viewport of interest by analyzing the output from the neural network models. Instead of checking the classification results from the *softmax* function of the CNN model, we only focus on the results that belong to the class of "interested". We sort the scores from the classification results and select the M position in the sorted list as the threshold. Then, the images with a score that is no less than the threshold will be selected as the predicted viewport of interest. As a result, the total number of tiles that are selected in each frame is not fixed. The locations of those top scored images in the original frame constitute the predicted user viewport. In this work, we set M to 0.5. By setting different "M" values, one can obtain different bandwidth savings with different prediction accuracies.

As shown in Figure 2, our processing pipeline starts the inference with a non-pre-trained model on the first video segment S_0 for a new user. The collected images from S_0 are provided to the CNN model and used to infer if the user likes the images or not. Once the labeling is finished after the user has watched the video segment, we compare the inference result with the labeling to calculate the loss value, based on which we train the CNN model. Then, we leverage the model trained based on S_0 (i.e., the video content and the user viewing behavior) to infer the user's interest in the images from the next segment S_1. Finally, we update the model based on the user feedback (i.e., the labeling and training) from segment S_0.

Table 1: Adjustment of learning rate (LR) based on loss and coverage.

Loss Value	LR	Coverage Rate (%)	LR
< 0.2	0	0∼25	0.01
0.4∼0.2	0.001	25∼50	0.008
0.4∼0.6	0.004	50∼75	0.006
>0.6	0.006	75∼100	0.004

4.1.2 Training procedure

Once the predicted user viewport is generated based on the top M scored images, the user would watch the corresponding segment, and the actual user viewport can be collected for labeling the training data. In traditional supervised learning, the training time depends on the acceptable minimum loss value and the epoch. In *LiveDeep*, we use a high minimum loss value and a low maximum epoch value to meet the real-time requirement.

- ***High acceptable loss value***. During the experiment, we find that the prediction method could achieve a high accuracy when the loss is lower than 0.2, which is different from the common scenario of classification applications. The reason is that we do not directly use the classification output as the prediction result. Instead, we only select the most likely ones from the output and thus the high loss value as the threshold for the termination criteria of training. We further observe that the CNN model could learn so deep that, when the user switches to the new content, the model would spend more time updating the "prediction skills". As a result, by setting the threshold as $loss = 0.2$, we could terminate training when the model is well trained for the current state and make it easier for the CNN to learn new user viewing behavior based on new user preference and video content.

- ***The number of epochs***. Even if the loss value is high for several epochs during the training, we cannot keep reducing it by updating the model with a large number of epochs because of the real-time requirement brought by live VR streaming. Instead, we set the maximum number of epochs as 10, which is much smaller than that in traditional CNN applications. Consequently, the training overhead is bounded to a fixed range.

- ***The batch size***. Due to limited training images, in each video segment, we set the batch size as the number of training images (i.e., $k \times x \times y$). Therefore, the total number of iterations equals to the number of epochs, which further reduces the training overhead.

- ***Dynamic learning rate***. At different stages, the deep learning method could adopt different learning rates. In *LiveDeep*, the learning rate is dynamically adjusted based on both the loss value in the previous video segment and the coverage rate in the previous frame. (1) *Loss-based:* we define four levels of loss values and set the learning rate accordingly. Table 1 shows how the learning rate is adjusted to different levels based on loss values. If the loss value is smaller than 0.2, we stop updating the model, as described earlier. (2) *Coverage-rate-based:* similar to IoU loss [33], we calculate how much the user's actual viewport is covered by the predicted user viewport (a.k.a, the coverage rate). The coverage rate indicates how well the model works. For a low coverage rate, we would adopt a higher learning rate. For a high coverage rate, we could even skip the training process to reduce the timing overhead. The learning rate adjustment is at the beginning of the training for each video segment. The adjusted learning rates based on different levels of coverage rate are also listed in Table 1.

Table 2: Test videos from a public dataset [41].

No.	Video Name	Category	Content	Cameras	Background	Outdoor Scene	Frame rate
0	Cooking	Performance	Cooking show	1	Static	None	25
1	RioVR	Performance	Interview talk show	1	Static	None	25
2	FemaleBasketball	Sport	Basketball game	1	Static	None	30
3	Fighting	Sport	Showtime boxing	2	Dynamic	None	30
4	Anitta	Performance	Anitta dancing	1	Static	None	30
5	Conan Gore Fly	Performance	Talkshow	2	Dynamic	None	30
6	TahitiSurf	Sport	Surfing	1	Dynamic	Included	30
7	Reloaded	Performance	Concert	1	Static	None	30
8	VoiceToy	Performance	Concert	1	Static	None	25
9	Front	Sport	Skiting	1	Dynamic	Included	30
10	Falluja	Documentary	The fight for Falluja	1	Dynamic	Included	30
11	Football	Sport	Football game	1	Dynamic	None	25
12	Rhinos	Documentary	The last of the rhinos	2	Dynamic	Included	30
13	Korean	Performance	Weekly idol dancing	2	Dynamic	None	30

4.2 LSTM-based user trajectory prediction

We note that the single CNN model would have a strong "memory" of certain video content, and it would take a long time for the model to accept new user preference on new video content. Consequently, the CNN model could not respond quickly to the fast switching of user preferences. To improve the perception of the model for frequent user preference switches, we develop a user trace-based viewport prediction method by leveraging the LSTM [16] model to predict the user's trajectory in a short time.

In particular, we use the same supervised learning process as in the original LSTM model, which is to train the model with the collected training data first and then infer the new data in the future. We set the size of hidden layer as 64 and use 2 hidden layers. Then, we collect the user trace in the previous video segment, which consists of k sample points from the k subsampled frames. The k sample points are collected as the training data, and we assign the index of each frame in each sample point as the timestamp. Finally, we train the LSTM model with the collected training data and infer the user viewport in the next video segment. The output of the LSTM model is the indices of tiles that the LSTM predicts.

4.3 The hybrid CNN/LSTM model

We adopt the prediction results from both the CNN model and the LSTM model to generate the final predicted viewport for each video segment. During this process, we first set the prediction results obtained by the CNN model as the main results. Then, we combine the results from both methods as the final predicted viewport.

5 Experimental Results

In this section, we describe our experimental results based on a prototype system implementation of *LiveDeep*.

5.1 Prototype system implementation and deployment

We implement *LiveDeep* on a Dell workstation with two Intel Xeon E5-2623 CPUs and one GPU of Titan X with 32G RAM. We adopt PyTorch 1.0 to implement the CNN and LSTM modules and use CUDA to accelerate the deep learning processing. We evaluate *LiveDeep* using videos and head movement traces from a public dataset [41], in which each video was watched by 48 users.

Figure 3 demonstrates the viewport prediction results using 4 test videos from the dataset [41]. Figure 3 (a) shows a video captured by a moving camera, in which the surroundings are moving along with the camera, creating a challenging case for the viewport prediction algorithm. Figure 3 (b) shows a video containing a single "attractive region" captured by a single camera. In this case, the target area seems fixed in the center and the background is static, which makes it a relatively simple case for viewport prediction. Figure 3 (c) demonstrates a video that contains multiple scenes, in which the surroundings are static but the scene changes very often. Figure 3 (d) demonstrates a video with multiple "attractive regions", which is relatively difficult for viewport prediction due to the possible view switches. The blue rectangles indicate the viewports predicted by *LiveDeep*, and the green rectangles indicate the actual user viewports (i.e., the ground truth). The results demonstrate that *LiveDeep* could cover most regions watched by the user. Note that the final predicted viewports are not regular shapes, but they are compliant with existing tile-based selective streaming designs [38] for direct deployment.

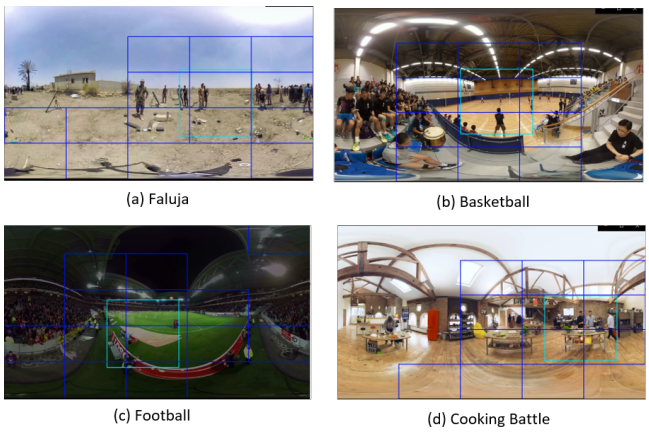

Figure 3: Snapshots of viewport prediction results achieved by *LiveDeep* using 4 test videos from [41]. The blue rectangles indicate the viewports predicted by *LiveDeep*, and the green rectangles indicate the actual user viewports (i.e., the ground truth).

5.2 Experimental setup

5.2.1 Evaluation metrics

To achieve a comprehensive evaluation on the performance of *LiveDeep*, we adopt three evaluation metrics, namely prediction accuracy, bandwidth usage, and processing time. First, the prediction accuracy is an important performance factor that directly impacts the user experience. If the prediction is incorrect, the user would watch low quality videos according to the state-of-the-art tile-based VR streaming designs that *LiveDeep* would be integrated to [38].

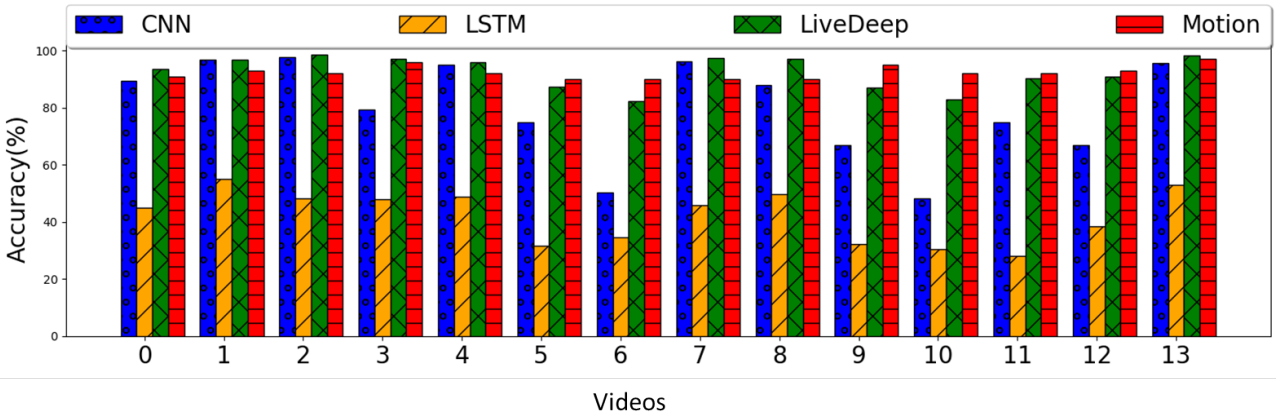

Figure 4: Overall prediction accuracy (Epoch=10,VGG13) with the comparison among the four methods.

Different from the popular IoU metric for accuracy evaluation [33], we evaluate the prediction accuracy by using a correct/incorrect labeling method. For each frame, we compare the predicted viewport with the actual user view and treat the prediction as "correct" only if the actual user view is completely covered by the predicted viewport. The prediction accuracy is then calculated by the number of "correct" frames over the total number of frames in the video. Second, the bandwidth reduction in VR streaming is the original and fundamental goal of this work, but it also presents a trade-off between bandwidth savings and prediction accuracy. Therefore, the key issue is how well *LiveDeep* would balance the trade-off, which must be evaluated using the bandwidth usage metric. Third, the processing time evaluates the unique real-time requirement brought by the live VR streaming system, which must be fulfilled for a smooth live streaming experience.

5.2.2 Test Dataset

We adopt 14 test videos from the empirical user head movement dataset for VR streaming [41], as shown in Table 2. We observe that the videos are different in their content (i.e., sports, performance, and documentary), number of capturing cameras (i.e., 1 or 2), background motion (i.e., static or dynamic), and whether outdoor scene is involved. This diverse set of video features would impact the prediction accuracy and thus the overall performance of *LiveDeep*, which we evaluate in our experiments. More importantly, all the 14 test videos are associated with the actual head movement traces collected from 48 real-world users, which are included in the dataset [41]. We adopt these head movement traces as the ground truth to evaluate the prediction accuracy of *LiveDeep*.

5.2.3 Test Cases

We adopt two test cases in our evaluation to cover both the basic *single-video* viewing experience and the *lifelong* scenario concatenating multiple videos of different types.

- **The *single-video* test case.** We first evaluate *LiveDeep* in the scenario of 14 single videos watched by single users obtained from the dataset [41], as summarized in Table 2. For each single video, we follow the workflow presented in Figure 1 to iteratively train and predict the user viewports for the continuous segments of the video content.

- **The *lifelong* test case.** We further create a *lifelong* test case, where the user continuously watches different videos (i.e., concatenating all the 14 test videos from the dataset [41]). This is to reflect the real-world scenario where a user may watch different videos either from the same channel or by switching channels. The *lifelong* scenario may create unique challenges, e.g., whether the prediction algorithm could learn new video patterns while still keeping the "memory" of the old patterns.

5.3 Prediction accuracy

5.3.1 *Single-video* test case

Figure 4 shows the prediction accuracy results in the single-video test case. We compare the performance of 4 methods, namely CNN (i.e., only the CNN model is adopted for the viewport prediction), LSTM (i.e., only the LSTM model is adopted), *LiveDeep* (i.e., the proposed approach), and Motion (i.e., the state-of-the-art content-based live viewport prediction approach [12]).

We observe that both *LiveDeep* and *Motion* achieved high prediction accuracy (i.e., >80%) over the 14 test videos, and *LiveDeep* outperforms *Motion* in 8 out of the 14 videos. The CNN model, although performs well for a subset of videos and even better than motion based method in 4 of them, would result in significantly low prediction accuracy in Videos 6, 9, 10, and 12. According to Table 2, these videos contain outdoor scenes and thus the users are more likely to frequently switch viewport and look around, which creates a challenge for the single-model approaches. The LSTM method achieves the lowest prediction accuracy among the 4 methods, which is around 40% for all the test videos.

5.3.2 *Lifelong* test case

Figure 5 shows the prediction accuracy of CNN and *LiveDeep* over the 14 videos in the *lifelong* test case. The results indicate that the prediction accuracy of CNN varies significantly, ranging from around 50% to 90%, while *LiveDeep* achieves consistently high prediction accuracy over most of the test videos, ranging from 80% to 90%. We further compare the prediction accuracy of *LiveDeep* in the *single-video* and *lifelong* test cases, as shown in Figure 6. We observe that *LiveDeep* performs equally well in the two test cases, indicating its capability of supporting the empirical *lifelong* viewing scenario with premium prediction accuracy.

5.3.3 The impact of epoch

We evaluate the performance of *LiveDeep* under various numbers of epochs, which is one of the important parameters that would impact the prediction accuracy and timing overhead of deep neural network models. The intuition about epoch is that more epochs would help achieve higher prediction accuracy. However, more epochs also means longer processing time in the prediction. To quantitatively analyze this potential tradeoff, we set 3 maximum epoch values for the training process, namely 4, 10, and 14. The results are shown in Table 3, including prediction accuracy (ACC) and maximum processing time (MT) for both the CNN and the

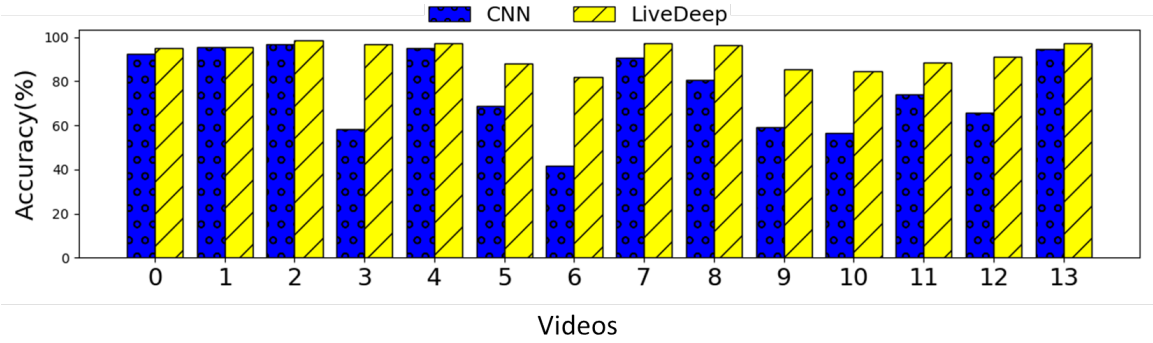

Figure 5: Prediction accuracy in the *lifelong* test case comparing CNN and *LiveDeep* (Epoch=10, VGG13).

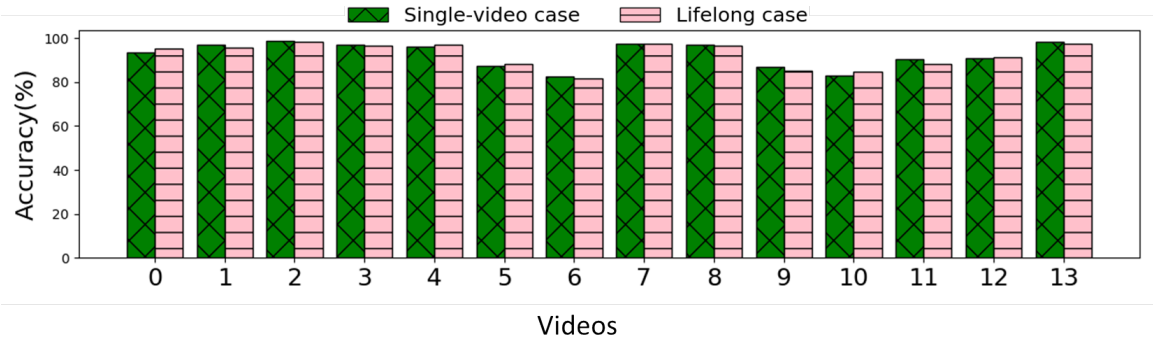

Figure 6: Prediction accuracy of *LiveDeep* in the *single-video* and *lifelong* test cases (Epoch=10, VGG13).

LiveDeep methods. The results indicate that for most of the test videos, the prediction accuracy of CNN would improve with more epochs (i.e., from epoch=6 to epoch=14). On the other hand, the accuracy of *LiveDeep* is consistently high (i.e., $> 80\%$) under all the epoch values, which implies that with *LiveDeep* we can achieve low processing time by choosing smaller epoch values (e.g., 6) without significantly impacting the prediction accuracy.

5.3.4 The impact of CNN structure

We further evaluate the impact of the CNN structure by comparing the prediction accuracy based on AlexNet [20], VGG11, VGG13, VGG16 and VGG19 [36], listed in ascending order of network complexity. Table 4 shows the average prediction accuracy and the maximum processing time over all the 14 test videos. The results reveal that more complex networks, such as VGG, achieve higher prediction accuracy than simpler networks, such as AlexNet. The results also indicate that the processing time is impacted by the complexity of the network structure and, in all the test cases, the inference and training can be finished within 2 seconds (i.e., the length of one video segment).

5.4 Bandwidth savings

The size of the predicted region is an important factor that determines the bandwidth savings achieved by the viewport prediction mechanism. Figure 7 shows the boxplot of bandwidth usage, evaluated by the size of the predicted region over the original 360-degree frame for all the 14 videos and 48 users. The results show that CNN and *LiveDeep* achieve similar bandwidth efficiency (around 60% of the original bandwidth) with narrow range of distributions, indicating stable and consistent results over different types of videos. Overall, *LiveDeep* outperforms CNN with better prediction accuracy (discussed in Section 5.3). The *Motion* method, on the other hand, results in a huge variance in the bandwidth usage over different videos and users. This is because the GMM method adopted by the *Motion* method [12] introduces errors when the background of the video is dynamic. Therefore, even though the *Motion* method demonstrated premium prediction accuracy (i.e., shown in Figure 4), our proposed *LiveDeep* method outperforms *Motion* by achieving better overall performance for both accuracy and bandwidth.

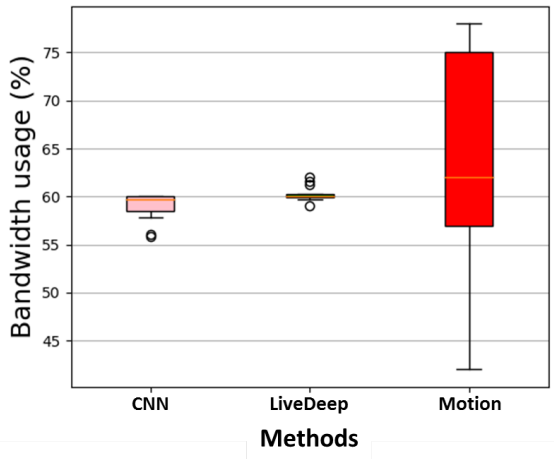

Figure 7: Average bandwidth consumption with viewport prediction evaluated by the percentage over the original bandwidth.

Note that the reported bandwidth savings are calculated based on the scenario of correct viewport prediction. When the viewport prediction is incorrect, the bandwidth savings would be worse and would depend on the recovery strategy that is orthogonal to viewport prediction. There are typically three recovery strategies, in the descending order of bandwidth consumption: (1) Selective streaming [38], in which the tiles that are outside the predicted viewport

Table 3: Performance for different epoch (VGG13). "ACC" stands for the accuracy (%), and "MT" stands for maximum processing time (s).

Video	Epoch=6				Epoch=10				Epoch=14			
	CNN		LiveDeep		CNN		LiveDeep		CNN		LiveDeep	
	ACC	MT	ACC	MT	ACC	MT	ACC	MT	ACC	MT	ACC	MT
0 Cooking Battle	70.7	0.7	93.7	0.8	91.4	1.0	93.1	1.1	92.5	1.5	99.2	1.6
1 RioVR	98.4	0.6	98.6	0.7	98.0	1.0	97.7	1.1	96.8	1.4	98.3	1.6
2 FemaleBasketball	97.3	0.7	98.3	0.8	98.1	1.0	98.7	1.1	98.8	1.5	99.7	1.6
3 Fighting	76.9	0.7	97.3	0.8	89.6	1.0	96.3	1.1	83.5	1.5	97.3	1.6
4 Anitta	87.5	0.7	95.8	0.8	67.4	1.0	95.3	1.1	94.8	1.5	97.6	1.7
5 Conan Gore Fly	41.7	0.8	81.5	0.9	59.3	1.0	87.1	1.1	59.5	1.5	97.4	1.6
6 TahitiSurf	33.0	0.7	84.9	0.8	43.0	1.1	84.9	1.3	59.2	1.5	97.3	1.6
7 Reloaded	90.3	0.7	97.4	0.8	92.7	1.0	96.4	1.1	96.8	1.5	98.4	1.6
8 VoiceToy	90.6	0.7	98.3	0.8	91.5	1.0	97.9	1.1	93.2	1.4	98.4	1.5
9 Front	48.9	0.7	83.6	0.8	48.3	1.0	85.7	1.2	71.2	1.5	98.1	1.6
10 Falluja	32.8	0.8	83.5	0.9	45.9	1.1	81.9	1.3	58.5	1.5	99.3	1.6
11 Football	56.7	0.7	82.6	0.9	71.4	1.0	85.5	1.2	76.6	1.4	97.5	1.5
12 Rhinos	44.5	0.7	86.8	0.9	56.4	1.0	90.1	1.2	69.3	1.5	97.8	1.7
13 Korean	89.2	0.7	97.6	0.8	95.8	1.0	97.4	1.1	95.7	1.5	98.1	1.6

would still be streamed from the server to the client but with lower resolution; (2) re-transmission of the video segment with correct viewport [22]; and (3) no recovery, in which the user would watch blank view under a wrong viewport prediction. Since the viewport prediction accuracy of *LiveDeep* is very high (around 90%), the overall bandwidth savings even considering the case of incorrect predictions will not significantly vary from the bandwidth savings in the case of correct prediction.

5.5 Timing overhead

To ensure a smooth viewing experience in live VR streaming, the processing time for each video segment must be smaller than the segment duration (i.e., 2 seconds in our experiments). Otherwise, the processing delay would be accumulated into the end-to-end streaming latency, and the user may experience re-buffering or frozen frames. Therefore, we evaluate the processing time required by *LiveDeep* while executing the proposed viewport prediction tasks, the results of which are shown in Table 3 and Table 4. Table 3 indicates that the processing time would increase with the epoch value but, even in the case of epoch=14, *LiveDeep* can still meet the real-time requirement in live streaming (i.e., < 2 seconds of maximum processing time) while achieving very high prediction accuracy (97%-99%). Table 4 shows that increasing the complexity of the neural network structure (e.g., from VGG11 to VGG 19) may not necessarily improve the prediction accuracy but would incur higher processing overhead. However, even with VGG19 *LiveDeep* still achieves less than 2 seconds of maximum processing time and thus satisfies the real-time requirement.

Table 4: Performance comparison of different deep neural network structures (epoch=10).

Network	Accuracy	Average Time	Maximum Time
AlexNet	88.0%	0.5s	0.8s
VGG11	92.0%	0.6s	1.0s
VGG13	92.7%	0.7s	1.2s
VGG16	92.2%	0.8s	1.6s
VGG19	91.0%	1.1s	1.6s

6 Conclusion and Future Work

In this paper, we developed a novel viewport prediction approach, namely *LiveDeep*, for live VR streaming. *LiveDeep* employs a hybrid method to address the live streaming challenges, including an alternate online/lifelong training and learning process, as well as a mixture of hybrid neural network models. We evaluated *LiveDeep* using 48 users and 14 VR videos of various types obtained from a public VR user head movement dataset. The experimental results show that *LiveDeep* could achieve high prediction accuracy with premium bandwidth usage and short processing time. Through evaluation with *single-video* and *lifelong* use cases, we demonstrate that *LiveDeep* is able to maintain high prediction accuracy when the user continuously watches different videos. The source code and other materials of this work can be found in our *LiveDeep* project repository [11].

By analyzing the prediction accuracy results and the predicted user viewports in the test videos, we note that there are two challenging problems that are still difficult to address, which we plan for continuing explorations in the future work. The first challenge is the prediction for videos with huge scenes in an open field, such as in Videos 6, 9 and 10. The scenes are also dynamically changing, which attracts more attentions from the user and results in high probability of switching viewports. Consequently, the new video content and the diverse user traces may compromise the prediction accuracy of the hybrid model. The second challenge is the scenario of frequent user switches between multiple attractive objects, such as in Video 0 and Video 4. The user switches are hard to detect using the LSTM model, and the attractive objects share similar patterns but with different labels when the viewport only covers one object, which increases the difficulty in training the CNN model.

Acknowledgments

We appreciate the constructive comments provided by anonymous reviewers. This work was partially supported by the National Science Foundation under awards CNS-1910085 and CNS-1618931.

REFERENCES

[1] ABC News, VR Virtual reality news has opened the door to boundless possibilities allowing users to be anywhere we are at any time. 2019. https://abcnews.go.com/VR.

[2] Y. Ban, L. Xie, Z. Xu, X. Zhang, Z. Guo, and Y. Wang. Cub360: Exploiting cross-users behaviors for viewport prediction in 360 video adaptive streaming. In *IEEE International Conference on Multimedia and Expo*, pp. 1–6, 2018.

[3] Y. Bao, H. Wu, A. A. Ramli, B. Wang, and X. Liu. Viewing 360 degree videos: Motion prediction and bandwidth optimization. In *International Conference on Network Protocols*, pp. 1–2, 2016.

[4] 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 *IEEE International Conference on Big Data*, pp. 1161–1170, 2016.

[5] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li. Salient object detection: A survey. *Computational Visual Media*, pp. 1–34, 2014.

[6] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand. What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*, 41(3):740–757, 2018.

[7] CNET. Watch any YouTube video in VR mode. 2016. https://www.cnet.com/how-to/watch-any-youtube-video-in-vr-mode/.

[8] C. Dawson. VR and AR will push the limits of connectivity, 2019. https://immersed.io/pushing-the-limits-of-connectivity/.

[9] 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 *ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 67–72, 2017.

[10] X. Feng, Z. Bao, and S. Wei. Exploring CNN-based viewport prediction for live virtual reality streaming. In *IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 183–1833, 2019.

[11] X. Feng, Y. Liu, and S. Wei. Repository for Project *LiveDeep*, 2020. https://github.com/hwsel/LiveDeep.

[12] X. Feng, V. Swaminathan, and S. Wei. Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(2):43, 2019.

[13] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(10):1915–1926, 2011.

[14] GSMA. Cloud AR/VR Whitepaper. 2019. https://www.gsma.com/futurenetworks/wiki/cloud-ar-vr-whitepaper/.

[15] H. Hadizadeh and I. V. Bajić. Saliency-aware video compression. *IEEE Transactions on Image Processing*, 23(1):19–33, 2013.

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] M. Hosseini and V. Swaminathan. Adaptive 360 VR video streaming: Divide and conquer. In *IEEE International Symposium on Multimedia*, pp. 107–110, 2016.

[18] X. Huang, C. Shen, X. Boix, and Q. Zhao. Salicon: Reducing the semantic gap in saliency prediction by adapting deep neural networks. In *IEEE International Conference on Computer Vision*, pp. 262–270, 2015.

[19] IDC. AR/VR headsets return to growth in the first quarter as new models and use cases restore momentum to the market, according to IDC. 2019. https://www.idc.com/getdoc.jsp?containerId=prUS45326719.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[21] E. Kuzyakov, S. Chen, and R. Peng. Enhancing high-resolution 360 streaming with view prediction. Facebook Inc., 2017.

[22] C. S. Lakshmi, G. S. Kumar, and V. Venkatachalam. Survey on caching and replication algorithm for content distribution in peer to peer networks. *International Journal of Computer Science and Network Security (IJCSNS)*, 15(10):78, 2015.

[23] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the Oculus Rift. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 187–194. IEEE, 2014.

[24] C. Li, W. Zhang, Y. Liu, and Y. Wang. Very long term field of view prediction for 360-degree video streaming. *arXiv preprint arXiv:1902.01439*, 2019.

[25] L. Matney. CNN launches dedicated virtual reality journalism unit, 2017. https://techcrunch.com/2017/03/07/cnn-launches-dedicated-virtual-reality-journalism-unit/.

[26] Mavlankar. *Video Streaming with Interactive Pan/Tilt/Zoom*, pp. 431–455. Springer Berlin Heidelberg, 2010.

[27] A. Mavlankar and B. Girod. Pre-fetching based on video analysis for interactive region-of-interest streaming of soccer sequences. In *International Conference on Image Processing*, pp. 3025–3028, 2009.

[28] Oculus. VR Headsets and Equipment. 2019. https://www.oculus.com/.

[29] S. Petrangeli, G. Simon, and V. Swaminathan. Trajectory-based viewport prediction for 360-degree virtual reality videos. In *IEEE International Conference on Artificial Intelligence and Virtual Reality*, pp. 157–160, 2018.

[30] S. Petrangeli, V. Swaminathan, M. Hosseini, and F. De Turck. Improving virtual reality streaming using HTTP/2. In *ACM Multimedia Systems Conference (MMSys)*, pp. 225–228, 2017.

[31] M. Przepiorkowski. VR is leading us into the next generation of sports media., 2017. https://venturebeat.com/2018/11/16/vr-is-leading-us-into-the-next-generation-of-sports-media/.

[32] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan. Optimizing 360 video delivery over cellular networks. In *ACM Workshop on All Things Cellular: Operations, Applications and Challenges*, pp. 1–6, 2016.

[33] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 658–666, 2019.

[34] A. Robertson. Facebook now lets you live stream from inside VR. 2017. https://www.theverge.com/2017/7/12/15944120/facebook-live-streaming-spaces-vr-social-oculus-rift.

[35] J. Shieber. National Geographic is working with YouTube and DayDream on its latest VR series., 2018. https://techcrunch.com/2018/12/11/national-geographic-is-working-with-youtube-and-daydream-on-its-latest-vr-series/.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] I. Sodagar. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE multimedia*, 18(4):62–67, 2011.

[38] J. Son, D. Jang, and E.-S. Ryu. Implementing 360 video tiled streaming system. In *ACM Multimedia Systems Conference*, pp. 521–524, 2018.

[39] TechRadar. HTC VIVE review. 2019. https://www.techradar.com/reviews/wearables/htc-vive-1286775/review.

[40] S. Wei and V. Swaminathan. Low latency live video streaming over HTTP 2.0. In *ACM Workshop on Network and Operating System Support on Digital Audio and Video*, pp. 37:37–37:42, 2014.

[41] C. Wu, Z. Tan, Z. Wang, and S. Yang. A dataset for exploring user behaviors in vr spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*, pp. 193–198. ACM, 2017.

[42] M. Xu, Y. Song, J. Wang, M. Qiao, L. Huo, and Z. Wang. Predicting head movement in panoramic video: A deep reinforcement learning approach. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 41(11):2693–2708, 2018.

[43] Y. Xu, Y. Dong, J. Wu, Z. Sun, Z. Shi, J. Yu, and S. Gao. Gaze prediction in dynamic 360 immersive videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5333–5342, 2018.