Exploring CNN-based Viewport Prediction for Live Virtual Reality Streaming

Xianglong Feng
Electrical and Computer Engineering
Rutgers University
Piscataway, USA
xf56@scarletmail.rutgers.edu

Zeyang Bao
Computer Science
Rutgers University
Piscataway, USA
zb95@scarletmail.rutgers.edu

Sheng Wei
Electrical and Computer Engineering
Rutgers University
Piscataway, USA
sheng.wei@rutgers.edu

***Abstract*—Live virtual reality streaming (a.k.a., 360-degree video streaming) is gaining popularity recently with its rapid growth in the consumer market. However, the huge bandwidth required by delivering the 360-degree frames becomes the bottleneck, keeping this application from a wider range of deployment. Research efforts have been carried out to solve the bandwidth problem by predicting the user's viewport of interest and selectively streaming a part of the whole frame. However, currently most of the viewport prediction approaches cannot address the unique challenges in the live streaming scenario, where there is no historical user or video traces to build the prediction model. In this paper, we explore the opportunity of leveraging convolutional neural network (CNN) to predict the user's viewport in live streaming by modifying the workflow of the CNN application and the training/testing process. The evaluation results reveal that the CNN-based method could achieve a high prediction accuracy with low bandwidth usage and low timing overhead.**

I. Introduction

With the rapid growth of Head-Mounted Displays (HMDs) in the consumer market, virtual reality (VR) video streaming (a.k.a., 360-degree video streaming) is gaining popularities recently [1]. However, the huge bandwidth consumption required by the immersive experience becomes the bottleneck of the VR streaming system. Different from traditional 2D videos, the VR video frames provide a sphere of the 360-degree video scene centering at the HMD device. Within the sphere, the user could switch to any portion of the video via head movement. In this case, a typical 720p (1280×720) 360-degree video would be approximately equivalent to a traditional 2D video with 3840×1920 resolution, resulting in huge bandwidth usage.

To solve this problem, several research efforts have focused on viewport prediction-based selective streaming [2]. These approaches independently encode a selected portion of the 360-degree frame with high resolution and the rest with low resolution to reduce the video size. The basis of these approaches is that there exists a perfect viewport prediction algorithm to figure out the user's viewport, i.e., the field of view (FOV). However, most of the existing viewport prediction methods [3]–[5] are designed for the video-on-demand scenario, as opposed to live VR streaming, the more attractive use case for VR-based consumer applications. In particular, the existing techniques are not able to handle two unique features in live VR streaming. **Feature #1** is that live streaming has a strict requirement on the video processing time, as the delay caused by the viewport prediction algorithm would be accumulated into the live streaming latency [6]. **Feature #2** is that in live VR streaming the video content is created on the fly, which means that no past video or user traces would be available to build the prediction model.

In this paper, we explore the opportunity of modifying the traditional CNN application architecture to accommodate for the two new features in live VR streaming. For **Feature #1**, we avoid using complicated deep learning structure and reduce the number of training iterations. For **Feature #2**, we collect the training data and train the model online at the runtime. Furthermore, we modify the output of the CNN by setting a dynamic threshold to finally predict the user viewport.

II. Background and Related Work

Figure 1 illustrates the overall workflow of live VR video streaming with viewport prediction. Typically, the 360-degree camera captures the panoramic video and delivers the stream to the media server. On the media server, the packager partitions the video into small segments (i.e., typically a few seconds each) and distributes them via the content distribution network (CDN) using standard video streaming mechanisms, such as DASH [7]. To reduce the bandwidth, selective streaming methods [2] can be applied on the server end, in which the predicted viewport of interest is encoded in high quality while the rest of the frame is in low quality.

Existing methods [3]–[5] leverage the historical user data to map the user interest to the video content by drawing a heatmap or using machine learning techniques. However, these methods would fail to address **Feature #2**. Other works [8], [9] estimate the user's head movement for a short period of time based on the user's head movement trajectory, in which the prediction accuracy would drop dramatically for long video buffer (i.e., longer than 1 second). The existing prediction methods that can be applied for live VR streaming is the motion-based method [10], which assumes that the user would watch moving objects in the video and thus detect the motions to achieve viewport prediction. However, the effectiveness of the motion-based method is limited by the motion detection method and would not perform well for a video with dynamic background.

978-1-7281-5604-0/19/$31.00 ©2019 IEEE
DOI 10.1109/AIVR46125.2019.00038

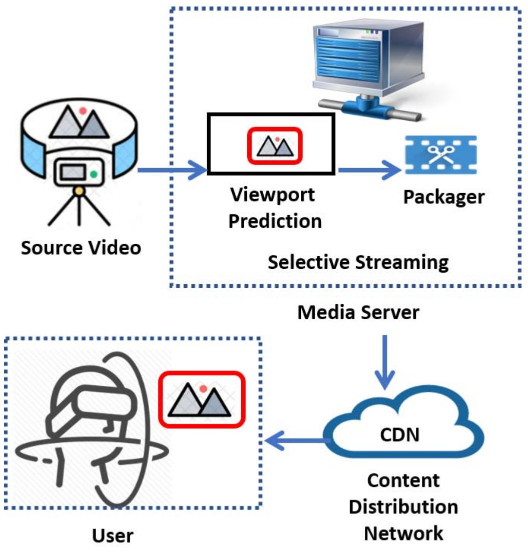


Fig. 1. Overall workflow of a typical live VR streaming system. The viewport prediction module is deployed at the server end before the packager encodes and generates the video segments.

III. CNN BASED LIVE VIEWPORT PREDICTION

In this work, we propose several modifications to the traditional CNN application as well as the video processing pipeline, which make the CNN-based prediction method possible to learn what viewports the user is interested in during the live streaming.

A. *Overall Framework*

Figure 2 illustrates our viewport prediction architecture. The training and prediction are carried out in every video segment (i.e., the video segment defined in the DASH standard [7]). Considering the fact that the continuous frames within the same video segment are very similar and that the huge amount of training images would increase the processing delay, we adopt a sub-sampling mechanism to reduce the number of frames for processing to 8 per segment. In each sampled frame, we divide the frame into 5×5 tiles, as indicated by the red net in Figure 2. Then, we collect a total of $8 \times 5 \times 5$ images as the input images for the CNN network. Based on the output results, we select the corresponding tiles as the predicted user viewport. In the end, after the user has watched the video segment, the user feedback is collected and, comparing with the CNN output, the loss is calculated to update the CNN model.

B. *AlexNet for Live Viewport Prediction*

We adopt AlexNet [11] as the CNN backbone network with five convolutional layers and three fully-connected layers. Between each pair of layers, we use the Relu function for modifying the input data. Three max-pooling layers are added in the convolutional layers, and both the convolutional layers and the fully-connected layers have a dropout layer. The overall structure and parameters are shown in Figure 3. In our work, there are two classes in the output, which represent "like" and "dislike" for the user.

1) Batch size: In traditional CNN applications, where all training data is available, the training data is divided into small batches. In live streaming, one video segment contains 200 images. We found that the processing time would increase when we divide the 200 images into small batches. As a result, we set the batch size to 200, which means that all the images from one video segment would be processed together.

2) Epoch: It is a common practice to leverage more iterations to reduce the loss during training. However, given the small training dataset in one video segment, we found that the loss value would decrease quickly, and it usually takes only 3 iterations till there is no obvious improvement. Therefore, considering the limited processing time, we set the epoch value to 3 for the training.

3) Training and testing: Different from the traditional CNN application, which trains the model first and tests the model later, our approach integrates the training and testing together. Given the images from one video segment, the network would conduct the forward propagation first to infer the user's preference. Based on the output from the forward propagation, we predict the user's viewport. Then, once the user has watched the video segment, his/her feedback will be collected to calculate the loss value and continue with the backward propagation to update the model.

4) Dataset collection at runtime: The 200 training images in one video segment are not labeled. After the user watched the segment, we collect the user feedback and label the images by comparing the user's viewport with the location of tiles in each frame. We define the image as "like" if the center of the user viewport lies in the corresponding tile in the frame.

5) Output results: We observe that the output results are not stable due to the limited training dataset and the dynamic video content based on the user's interest. In particular, the output after the softmax function contains huge errors. Therefore, besides the softmax function, we further analyze the value for "like" and set a dynamic threshold for selecting "like" based on the statistic information of the output. Specifically, we select the top one-fifth of all the tiles as the predicted viewport by ranking the output scores.

IV. EXPERIMENTAL RESULTS

A. *Experimental Setup*

In our experiments we adopt the 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 use CUDA to accelerate the processing in deep learning. We evaluate the proposed viewport prediction method using videos and head movement traces from the public dataset [12], in which each video was watched by 48 users.

B. *Prediction Accuracy*

Figure 4 shows the snapshots of the viewport prediction results with the three test videos "Falluja", "Cooking Battle" and "Female Basketball" from the public dataset [12]. The

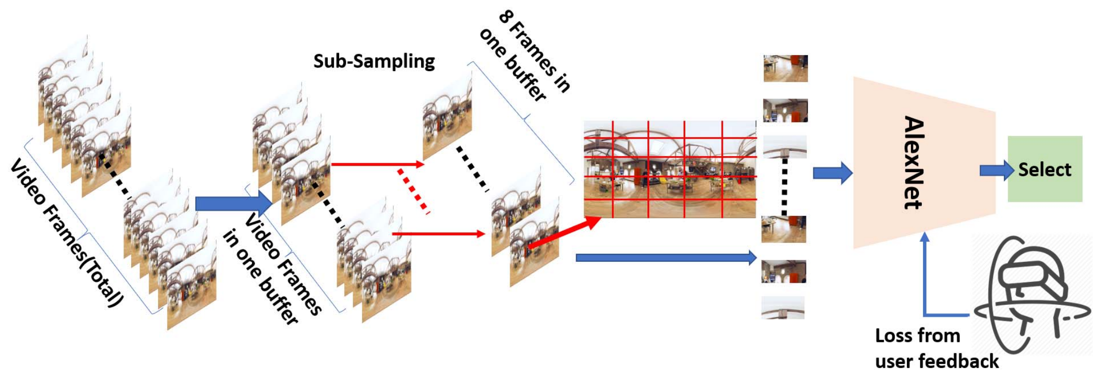
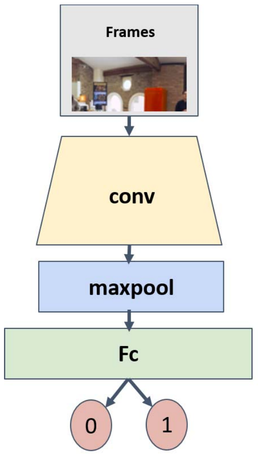


Fig. 2. Proposed viewport prediction framework. The workflow includes segmenting original video, sub-sampling from one video segment, cutting one frame to small tiles, and training/testing with AlexNet.



Raw image	32x32x3
Conv1	16x16x64
Maxpool1	8x8x64
Conv1	8x8x192
Maxpool1	4x4x192
Conv1	4x4x384
Conv1	4x4x256
Conv1	4x4x256
Maxpool3	2x2x256
FC1	1024x1
FC2	4096x1
FC3	4096x1

Fig. 3. Detailed design of the AlexNet architecture. The left figure is the general illustration of the CNN architecture and the table on the right shows the detailed parameters for each layer.

"Falluja" video has both static and dynamic backgrounds, the "Cooking Battle" video contains three objects that are likely to attract the user's attention, and the "Female Basketball" video represents sports videos that is one of the most popular types of VR streaming applications. In Figure 4 the red circle in each picture indicates the user's actual viewport, and the blue rectangles indicate the predicted viewport, i.e., the tiles of the user's interest as inferred by the CNN model. It is shown that in the "Falluja" video, the big vehicles are labeled as the predicted viewport. In the "Cooking Battle" video, the players and the cooking tables are labeled as the predicted viewport. In the "Female Basketball" video, the basketball field is labeled as the predicted area. Overall, in all the three videos, the predicted viewports are related to the theme of the video.

Figure 5 plots the prediction accuracy results for the 3 test videos over the 48 users. The average prediction accuracy rates for videos "Falluja", "Cooking Battle" and "Female Basketball" are around 70%, 84% and 88%, respectively. We observe that video "Falluja" has more complicated and frequently changing scenes and, as a result, the prediction accuracy is lower and with more outliers. The video "Female Basketball" achieves the highest accuracy rate with low variance, because the model correctly sets the playground as the viewport of interest.

C. Bandwidth Usage

In the discussion in Section III-B5, we set a dynamic threshold for selecting the top one fifth of the total tiles based on the ranking of the output values. In this case, the theoretical tile usage (i.e., the estimated bandwidth usage) should be around 20%. In practice, there are several outputs sharing the same values and, as a result, more tiles would be selected than the theoretical estimation. In our experiment, we observe that our method only selects either 32% or 44% of the original frame, which means either 8 tiles or 11 tiles are selected out of 25 tiles in each prediction. We further analyze the frequency of each case and find that for most of the time (86%-87%), the prediction achieves 44% of the original frames size. These results combined indicate around 57% of bandwidth savings using our viewport prediction approach.

D. Timing Overhead

We further analyze the timing overhead of the viewport prediction process. Since we focus on live video streaming, the basic requirement is to finish processing each video segment within its duration, less other sources of delays such as network transmission and video decoding/rendering. We set a timer to collect the time consumption for each segment in the three test videos. Figure 6 shows the distribution of the viewport prediction processing time per segment. We observe that all the video segments are processed within 475 ms, which indicates that the CNN-based viewport prediction meets the timing requirement of live streaming.

V. Limitations

Our results also show that the prediction method introduces noises (i.e., predicting the area that the user is unlikely to watch), which would increase the bandwidth usage. Besides the noises, there are prediction errors (i.e., failed to detect the area that the user is interested in) even after 20 seconds of the

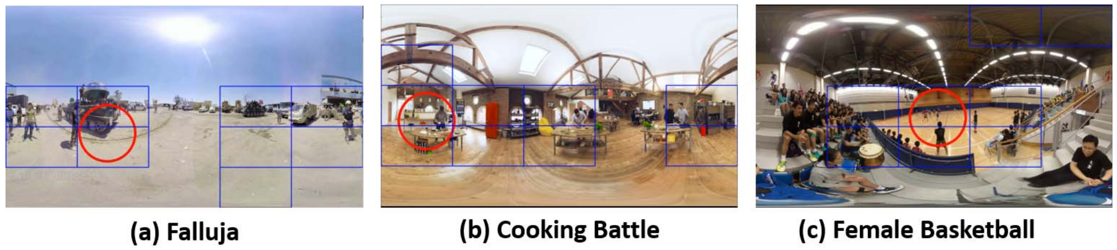


Fig. 4. Snapshot of the prediction results. The blue grids represent the predicted viewport. The red circle indicates the actual user viewport.

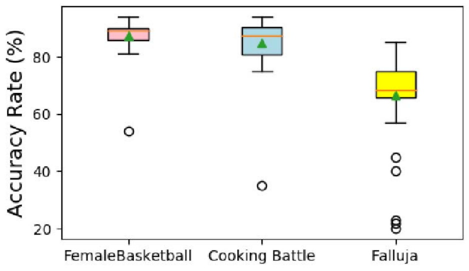


Fig. 5. The boxplot of the prediction accuracy for the three test videos over the 48 users. The three colored boxes represent the three videos.

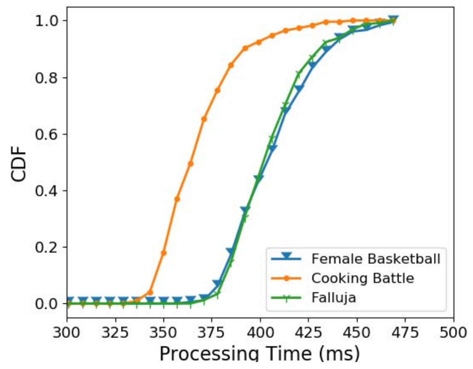


Fig. 6. Demonstration of average timing overhead for 3 test videos. X-axis indicates the processing time (ms) for one video segment and Y-axis indicates the cumulative distribution value.

video playback. This may be related to the user switching to a new viewport. Once the user switches to focus on different video content, the loss value would increase significantly and drop below 0.01 after two or more segments. Another issue is that improving the prediction accuracy in the first 20 seconds of video is still challenging because of the limited training data, even though the loss value has dropped to a low value. We also conducted further experiments and found that our approach would fail in complicated videos where different portions of the frame share similar patterns. In these scenarios, a single CNN model or less deep models (e.g., AlexNet) may be subject to prediction errors.

VI. Conclusion and Future Work

In this work, we proposed a CNN-based viewport prediction framework for live virtual reality streaming. To meet the unique requirements in live streaming (i.e., real-time and lack of historical data), we modify the processing pipeline, the structure, and the traditional training process of CNN, making the prediction model updated at runtime with user feedback. Our results show that the proposed approach could predict the user viewport accurately within the required time duration, which reduces the bandwidth usage to a bounded range. The current results also reveal limitations of this preliminary work, such as the noises in the prediction results of complicated video scenes, which we plan to improve in our future work.

Acknowledgement

We appreciate the constructive reviews provided by the anonymous reviewers. This work was supported in part by the National Science Foundation under Award CNS-1910085.

References

[1] REWO. (2019) Virtual reality market size in 2018 with forecast for 2019. [Online]. Available: https://www.viar360.com/virtual-reality-market-size-2018/

[2] J. Son, D. Jang, and E.-S. Ryu, "Implementing 360 video tiled streaming system," in *MMSys 2018*, pp. 521–524.

[3] E. Kuzyakov, S. Chen, and R. Peng, "Enhancing high-resolution 360 streaming with view prediction." Facebook Inc., 2017, https://code.facebook.com/posts/118926451990297/enhancing-high-resolution-360-streaming-with-view-prediction/.

[4] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *IEEE AIVR 2018*, pp. 157–160.

[5] 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 ICME 2018*, pp. 1–6.

[6] THEO. (2019) The importance of low latency in video streaming. [Online]. Available: https://www.theoplayer.com/blog/the-importance-of-low-latency-in-video-streaming

[7] I. Sodagar, "The MPEG-DASH standard for multimedia streaming over the internet," *IEEE multimedia*, vol. 18, no. 4, pp. 62–67, 2011.

[8] Yanan Bao, Huasen Wu, A. A. Ramli, Bradley Wang, and Xin Liu, "Viewing 360 degree videos: Motion prediction and bandwidth optimization," in *IEEE ICNP 2016*, pp. 1–2.

[9] A. Mavlankar and B. Girod, "Pre-fetching based on video analysis for interactive region-of-interest streaming of soccer sequences," in *IEEE ICIP 2009*, pp. 3025–3028.

[10] X. Feng, V. Swaminathan, and S. Wei, "Viewport prediction for live 360-degree mobile video streaming using user-content hybrid motion tracking," *ACM IMWUT*, vol. 3, no. 2, pp. 43:1–43:22, 2019.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[12] C. Wu, Z. Tan, Z. Wang, and S. Yang, "A dataset for exploring user behaviors in vr spherical video streaming," in *MMSys 2017*, pp. 193–198.