COLIVE: AN EDGE-ASSISTED ONLINE LEARNING FRAMEWORK FOR VIEWPORT PREDICTION IN 360° LIVE STREAMING

Mu Wang[*], *Shuai Peng*[†], *Xingyan Chen*[‡], *Yu Zhao*[‡], *Mingwei Xu*[*] *and Changqiao Xu*[†]

[*]Tsinghua University, Beijing, China
[‡]Southwestern University of Finance and Economics, Chengdu, China
[†]Beijing University of Posts and Telecommunications, Beijing, China
[*]muwang@mail.tsinghua.edu.cn, xmw@cernet.edu.cn
[‡]{xychen, zhaoyu}@swufe.edu.cn, [†]{pengshuai, cqxu}@bupt.edu.cn

2022 IEEE International Conference on Multimedia and Expo (ICME) | 978-1-6654-8563-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICME52920.2022.9859963

ABSTRACT

The ever-increasing demand for bandwidth resources when delivering premium quality 360° video challenges the current network capacity. To alleviate such bandwidth pressure, it is imperative to predict the viewport via observing the content visual feature and historical viewing behaviors, which thereby allows the system to concentrate the limited resource on viewer's region of interest in 360° content. However, enabling accurate viewport prediction for 360° live streaming is non-trivial given the time-sensitive of live content and shortage of pre-knowledge on the visual features and viewing behaviors. In this paper, we propose ***CoLive***, an edge-assisted online viewport prediction framework. ***CoLive*** incorporates edge computing to offload the prediction model training from viewers and migrates the saliency feature detection to the server side for reducing the processing delay. Viewers can also collaboratively train a central predicting model via sharing their loss gradients. This central model, together with the saliency feature detection, further prompts accuracy prediction and learning acceleration, especially for new incoming viewers. A series of experiments on the public 360° video dataset show how our solution achieves better performance compared with state-of-the-art solutions.

Index Terms— live 360° video, viewport prediction, online learning, edge computing

1. INTRODUCTION

With the provision of immersive view and rich interactivity, 360° video streaming (a.k.a immersive panoramic video) [1] has been considered as a revolutionary technology for the current content market. In particular, the fast development of wireless communication technologies and widespread head-mounted devices (HMDs) offer a solid foundation for the 360° live streaming services [2, 3], which simultaneously records and streams panoramic views of ongoing events. This emerging technology is fostering various applications such as VR streaming, meta-universe, etc, which gain tremendous attention from both academia and industry. The rapid commercialization of 360° live streaming also raises the demand for providing high-quality 360° content to end-users. However, supporting 360° live streaming with such high quality by current network infrastructure is nontrivial. 360° video content can be 24K [1] resolution and requires more than 1.0Gbps bandwidth, which is almost ten times of current worldwide average bandwidth [2]. Based on the fact that users' viewport only covers a small portion of the 360° video content[4], a promising trend to tackle the bandwidth shortage issue is to apply the adaptive streaming that only fetches the high-resolution content within the viewer's viewport region. Due to the viewport location varies with the change of viewer's region of interest, this technology's performance heavily relies on the accuracy of viewport predictions.

Numerous efforts [5, 6, 7] have been made to solve the viewport prediction problems in 360° video and achieve excellent results by both considering the content visual feature and viewer behaviors. For instance, in [8], Hou *et al.* leverages a two-layer LSTM network to predict the viewport movement. By further taking the visual feature into account, Wu *et al.* [7] applies a spherical CNN to predict the future viewport from long-term perspective. In [5, 6], predicting models are fed with both the saliency map and viewport moving traces, by the fact that the visual attention from the saliency maps potentially improves the prediction accuracy. These methods assume that enough content and viewing traces are available for enhancing the learning model performance. However, such assumption is untenable in live streaming given the video is generated on the fly and first time watched by the viewers.

Regression-based methods such as [9, 10] can be applied

This work was supported by China postdoctoral science foundation under Grant 62101301. The corresponding author is Xingyan Chen.

[1]https://www.huawei.com/en/technology-insights/industry-insights/outlook/mobile-broadband/xlabs/insights-whitepapers/cloud-vr-ar-white-paper

[2]https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

for 360° live streaming by real-time fitting the regression model according to the fresh data of viewing trajectories. Such solutions well performs when the viewport movement pattern can be relatively static for a while. In contrast, the frequent change of the movement pattern in real world caused by the dynamic of the viewer's interested region may impair the accuracy of the viewport prediction. Several works attempt to apply online learning to capture the saliency feature of video streaming. For example, Xu $et\ al.$[11] uses a deep reinforcement learning method with both online and offline methods to predict the future viewport. LiveDeep [12] uses convolution neural networks (CNN) to recognize the visual feature and long short-term memory (LSTM) predicts the viewport movement based on viewer trajectories and content saliency features. Although these solutions are applicable to 360° live streaming, several issues are yet to be solved:

Firstly, current solutions mainly rely on the viewers to train the saliency feature detection model and the viewport prediction model. An online learning requires extra computation overheads that may overrun the viewers' devices with limited capacities [13]. Besides, the saliency map can be only detected when the corresponding video content is generated, which may introduce an extra delay to the live streaming.

Secondly, 360° live streaming content is revealed to viewers for the first time which limits the prior knowledge on the user behavior. Training the prediction model with insufficient information about user behaviors may affect the prediction accuracy. Besides, live streaming viewers watch the live channel asynchronously. The new viewers need to train their own model from the beginning, which is time-consuming and may degrade their viewing experience.

Thirdly, viewport prediction can be considered as a spatial-temporal sequence forecasting problem, which motivates most current solutions [5, 6, 12] to apply the LSTM to predict the users' viewport locations. However, it is difficult for the original LSTM model to capture the spatial and temporal correlations between the input sequences simultaneously since the network is fully connected.

In this paper, we comprehensively tackle the above challenges by proposing a novel edge-assist online learning framework *CoLive*. The proposed framework avoids the computation burden of viewers' devices and the saliency detection latency discussed in the first challenge by offloading the prediction model to the edge server and embedding the saliency feature detection into media servers. Because of the existence of similar viewing behaviors across the viewers, *CoLive* tackles the second challenge by introducing a collaborative online learning-based prediction framework, in which viewers collaboratively refine their predicting models by uploading the local loss gradients to the central model. For the third challenge, each viewer in *CoLive* is equipped with a hybrid learning model based on convolutional LSTM (ConvLSTM) [14] for recognizing both the spatial and temporal variations of the input sequence of visual feature and viewport trajectories. Specifically, contributions of this paper are multi-folded:

- The edge-assisted learning framework for 360° live streaming viewport prediction that migrates the saliency detection to the server side and trains the prediction models via edge computing.
- A ConvLSTM-based predicting model at edge side that can capture the spatial and temporal relationships between the adjacent frames, in terms of the visual feature and viewport location.
- A throughout evaluation for *CoLive* on the performance of saliency detection, prediction and bandwidth saving, compared with several state of art solutions.

2. THE FRAMEWORK OF *COLIVE*

CoLive aims to timely predict the future viewport locations of viewers watching 360° live video. For this purpose, *CoLive* mainly consists of two components: saliency feature detection and viewport prediction. To accommodate the real-time content generate feature of live streaming, *CoLive* applies the online learning that repeatedly updated the model parameters according to the viewer feedback of each video chunk.

Fig. 1 shows the framework of *CoLive*: 1) The media server applies CNN-based network to output the saliency feature map and deliver this map with corresponding video segment to the edge server; 2) The edge server applies a ConvLSTM-based model for each viewer to predict the viewport locations of future video chunk according to the historical viewport trajectories and sequential saliency maps, which updates model parameters after viewer watch the video chunk; 3) The parametric gradients of viewers' model are integrated to train a central model, which can prompt others prediction performance and help the new incoming viewers to predict their viewport.

3. COLIVE DETAILS

In this section, we propose the detailed design of *CoLive*, including the methodologies of saliency feature detection and viewport prediction in *CoLive*.

3.1. 360° Video Saliency Feature Dection in *CoLive*

CoLive relies on the saliency feature of the video that includes the visual attention of viewers to predict the dynamic viewport. Different from the existing works that integrate the saliency detection model with predicting model as in [5, 8], we separate the saliency feature detection from viewport prediction and deploy the saliency detection model at server-side by the following reasons: 1) The viewer side is the destination of live streaming content. Training the saliency detection model at the viewer side may introduce an extra delay to live streaming which is unfriendly to the viewer. By testing the live streaming in the real world, we found there exists a latency between the streamer and viewers. For example, we test the average stream latency between a streamer

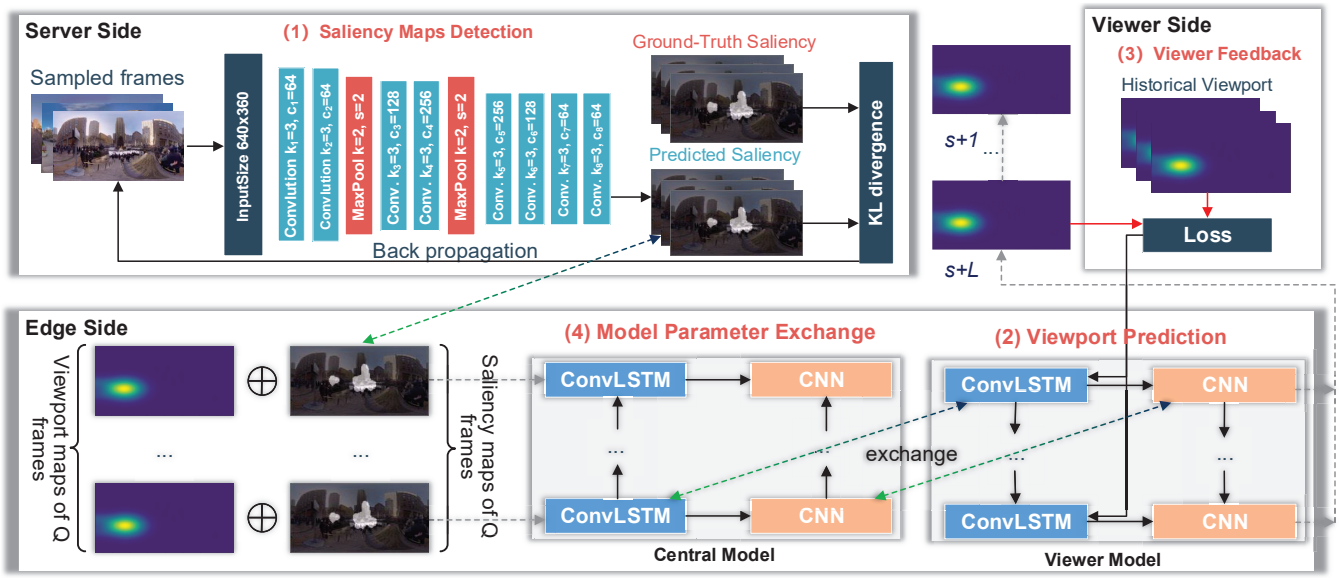


Fig. 1. Work flow in *CoLive*

in *twitch.tv*[3] and its viewers which are from different geographical regions including North Asia (Tokyo), North America (Los Angeles, Waterloo), and Asian-Pacific. We observe that the time of content held by the server before forwarding to viewers can be up to 2 seconds[4]. By leveraging such latency between the server and viewers, we place the saliency feature detection model at server side and deliver the output to edge side, which reduce the processing time of predicting the viewport; 2) Training the model of saliency feature detection at the viewer side is computationally inefficient since the saliency feature reflects the video visual characteristics instead of users' personal viewing behavior.

Considering the live content is generated in real-time and watched by the viewers for the first time, the offline model which is trained with historical trajectories from millions of viewers or transfer a pre-trained model is not suitable for live streaming. Instead, we design an online video saliency feature detection method that outputs the saliency feature of sampled frames for each received video chunk. *CoLive* outputs the saliency feature by using a CNN-based network. The structure of the network consists of 8 convolutional layers and 2 pooling layers. The out channel in each convolutional layer is 64, 64, 128, 256, 256, 128 and 1 respectively. The kernel size of each layer is set to 3×3. The input layer size is 640×360, and the final output layer size is 90×160.

Online Training Process. In order to reduce the delay on data processing, we need to sample each video chunk, which also avoids the over-fitting caused by training a large number of similar frames. In our experiment, the sampling rate is $k = 6$ and we normalize the input video frames to between 0 and 1. To determine the difference between a saliency map predicted by the model and the ground truth, we choose the *Kullback-Leibler* (KL) divergence as the loss function to measure the difference since the mean square loss function performs not very well. This can be defined as follows:

$$L_{KL}(P||Q) = \frac{1}{B}\sum_{i=1}^{i=N} Q_i \ln\left(\epsilon + \frac{Q_i}{\epsilon + P_i}\right) \quad (1)$$

where B is the batch size ($B = 8$), P is the predicted saliency maps, Q is the ground true (fixation maps) and ϵ is a regularization constant which equals to 10^{-7}. We apply the Adaptive Moment Estimation (Adam) gradient descent method to train the model. The result will be validated every 100 iterations and the learning rate is set to 1×10^{-6}.

The training process can be described as follow: the online training starts with detecting the saliency feature of the first chunk from live streaming by the CNN-based model with above configurations. The output saliency maps of sampled frames are then delivered to the edge server. Viewers who watched the chunk feed the server with viewport trajectories. The server updates the saliency detection model according to the feedback from viewers. The rest of chunk generated from the live channel follows the same training process.

3.2. Viewport Predictions in CoLive

In this subsection, we discuss how to leverage the limited knowledge in terms of the viewport trajectories and video saliency maps to predict the future viewport.

Viewport Prediction Problem. The purpose of the viewport prediction is to use the previous observed saliency maps and viewport motion trajectories to output a series of future viewport location maps. Let the value of pixels in the maps as the measurement, each saliency map can be represented by a tensor $X^s \in \mathbf{R}^{p \times a \times b}$ and viewport location map by a tensor $X^v \in \mathbf{R}^{a \times b}$, where p is the number of channels in the saliency map and p is equal to 1 since the saliency map is a black and white map with channel 1. Suppose saliency maps (viewport location maps,

[3]https://www.twitch.tv/

[4]The detail results of such observations can be referred to the supplementary materials

respectively) of received content is Q sequence of tensor $X^s_{t-Q+1}, X^s_{t-Q+2}, \ldots, X^s_t$ ($X^v_{t-Q+1}, X^v_{t-Q+2}, \ldots, X^v_t$, respectively). Accordingly, the viewport prediction problem is to output a L-length viewport location map sequence that indicates the parts of content that users are most likely to view in the future given the previous known Q-length sequence of saliency and viewport location maps with $\theta = \{s, v\}$.

$$\begin{aligned} &\hat{X}^v_{t+1}, \hat{X}^v_{t+2}, \ldots, \hat{X}^v_{t+L} \\ &= \underset{X^v_{t+1},\ldots,X^v_{t+L}}{\arg\max}\ p(X_{t+1}, \ldots, X_{t+L} | X^\theta_{t-Q+1}, \ldots, X^\theta_t) \end{aligned} \quad (2)$$

Model Structure. Most current studies use the fully connected LSTM (FC-LSTM) network to predict the viewport location, which neglects the spatial feature of video content sequences. Instead, we propose a ConvLSTM-based network for the viewport prediction. ConvLSTM extends the FC-LSTM by using a convolutional operator to replace the Hadamard product in LSTM cell, i.e.,[14]:

$$\begin{aligned} \mathbf{I}_t &= \sigma(W_{xi} * X^\theta_t + W_{hi} * H_{t-1} + W_{ci} * \mathcal{C}_{t-1} + B_i) \\ \mathbf{A}_t &= \sigma(W_{xa} * X^\theta_t + W_{ha} * H_{t-1} + W_{ca} * \mathcal{C}_{t-1} + B_a) \\ \mathbf{O}_t &= \sigma(W_{xo} * X^\theta_t + W_{ho} * H_{t-1} + W_{co} * \mathcal{C}_{t-1} + B_o) \\ \mathbf{C}_t &= \mathbf{A}_t \circ \mathcal{C}_{t-1} + \mathbf{I}_t \circ \tanh(W_{xc} * X_t + W_{hc} * H_{t-1} + B_c) \\ \mathbf{H}_t &= \mathbf{O}_t \circ \tanh \mathbf{C}_t \end{aligned}$$

where σ and $\tanh$ are the sigmoid and hyperbolic tangent, respectively. $*$ is the convolutional operator. The W_{xj}, W_{hj} and $W_{cj}, j = \{i, a, o, c\}$ are the kernal parameters, $\{B_i, B_a, B_o, B_c\}$ are the bias of the convolutional layers in ConvLSTM. $\mathbf{I}_t$, $\mathbf{A}_t$, $\mathbf{O}_t$ denote the input, forget and output gates at time t, respectively. $\mathbf{C}_t$ and $\mathbf{H}_t$ are the memory cell and hidden state at time t.

Based on ConvLSTM, we design a hybrid model whose structure is shown as Fig. 2 and deploy this model at edge servers. The hybrid model mainly consists of ConvLSTM processing input with a time length of Q and CNN processing output. The input of each ConvLSTM cell is a concatenation of the saliency map and viewport location map of a video frame. The output H_t of ConvLSTM is then fed to a CNN network that consists of two convolutional layers and two deconvolutional layers. The first convolutional layer is followed by a pooling layer and the second convolutional layer is followed by an upsampling layer. The output channel of convolutional and deconvolutional are 128, 128, 64, 1, respectively. The kernel size of each layer are 5,2,5,2,5,5, respectively.

Local Training Process. For online predicting the viewport, the edge server builds the above ConvLSTM hybrid model for each viewer and timely updates the prediction model when the viewer feedback the viewport data of sampled frames to the server. We define the loss function as mean square error with L2 regularization:

$$L_{ConvLSTM}(\hat{X}^v_t, X^v_t) = \frac{1}{Q}\sum_{t=1}^{Q} f(\hat{X}^v_t, X^v_t) - \lambda\|\omega'\|^2$$

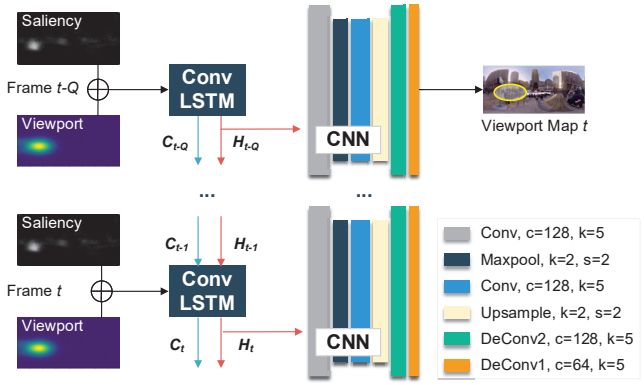


Fig. 2. ConvLSTM hybrid model in *CoLive*

where $\hat{X}^v_t$ and X^v_t are the predicted and real viewport location maps for each video chunk, respectively. To timely output the results, we set the iterations to 50 and learning rate is 1×10^{-3}.

Collaborative Learning Process. We test the similarity of viewing behaviors among in the public 360° video dataset [15]. As shown in Fig. 3, most of the viewers' ROIs are highly overlapped during the playback. For example, point A in the figure indicates that in the 801^{st} frame, more than 15 tiles are watched by 30 viewers. Thus, *CoLive* also collaboratively trains an integrated prediction model by allowing the model of each viewer to share their loss gradients to the central model. To achieve this design purpose, the central model has the same structure as that of the prediction model for each viewer. Viewer's model first calculates the gradient of his loss functions in hybrid model $\nabla L^s_{ConvLSTM}(\hat{X}^v_t, X^v_t)$ and upload the accumulated gradients every τ iteration. We define the accumulated gradients as $\alpha_l \sum_{d=1}^{d=\tau} \nabla L^s_{ConvLSTM}(\hat{X}^v_t, X^v_t, d)$, where $\nabla L^s_{ConvLSTM}(\hat{X}^v_t, X^v_t, d)$ indicates the gradients of $\nabla L^s_{ConvLSTM}$ at iteration d and α_l denotes the learning rate. Once the server receives the accumulated gradients from s, the model parameters will be updated as follows:

$$W_{CL}(e) = W_{CL}(e-1) - \frac{1}{N_k}\nabla L^s_{ConvLSTM}(\hat{X}^v_t, X^v_t)$$

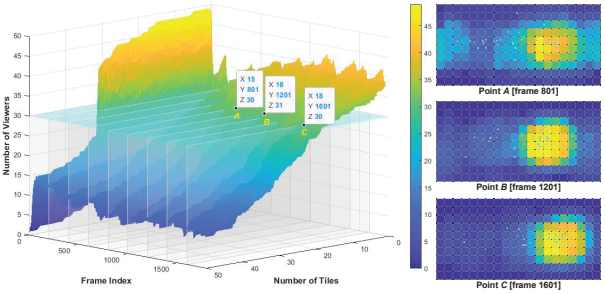


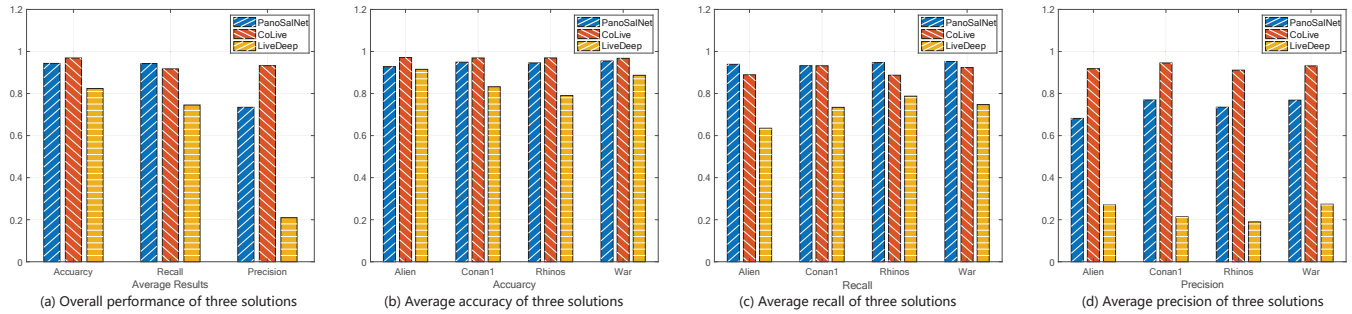
Fig. 3. Viewers similarity in viewport locations

4. PERFORMANCE EVALUATION

4.1. Experimental Setup

We have built a 360° streaming prototype system based on the open-source framework **srs**[5]. This prototype system in-

[5]https://github.com/ossrs/srs



(a) Overall performance of three solutions (b) Average accuracy of three solutions (c) Average recall of three solutions (d) Average precision of three solutions

Fig. 4. The performance of three methods (*CoLive*, *LiveDeep*, *PanoSalNet*): (a) Overall performance, (b) average accuarcy, (c) average recall, and (d) average precision.

cludes one central server (CS), one edge server (ES), two tethered terminals (TTs) and two mobile terminals (MTs). We provide the diagram of the system prototype in the supplementary materials. We deployed two servers (Dell R740 with Intel Xeon Gold 5222, 3.8Ghz/32G) as CS and ES, respectively. We implement the models of these solutions over four network nodes including two Dell workstations that for each has an i7-10700 CPUs and an RTX3070 with 8G RAM, and two personal notebooks (Intel i7 10750H CPUs and RTX2070 with 8G RAM and AMD R5 CPUs and RTX2060 with 6G RAM). We use **H**TTP **L**ive **S**treaming (**HLS**) to stream the live panoramic video from the severs to users. We deploy saliency feature detection model over the server side, and four learning agents (TTs and MTs) with the viewer prediction model at the edge server. We adopt the PyTorch 1.8.1 with Python 3.7 to implement our model[6].

We compare *CoLive* with *PanoSalNet* [5] and *LiveDeep* [12] based on the public panoramic video **user behavior dataset** [15] and **saliency dataset** [16]. A brief introduction of baseline methods and datasets are given in our supplementary materials.

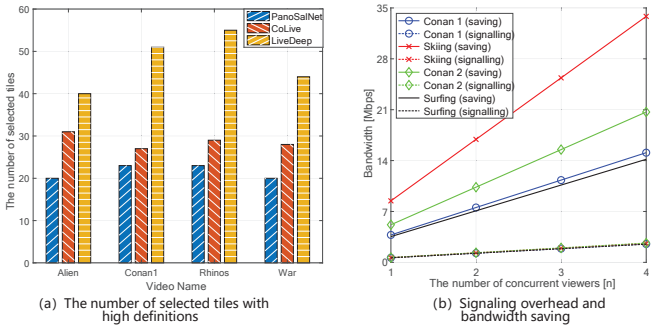
4.2. Evaluation on a System Prototype

Performance of Saliency Detection. Table 1 shows the saliency detection results of saliency dataset, *PanoSalNet* and *CoLive*. In this table, the saliency related metrics include CC, AUC, sAUC and NSS introduced by [17], which measure the errors between the detected saliency maps and ground truth eye fixation maps. These metrics suggest that our lightweight model can learn similar saliency feature compared with the model in *PanoSalNet*. Moreover, *PanoSalNet* is an offline model which cannot be applied for live streaming.

Results of Viewport Prediction. Fig. 4 compares the three methods in overall performance and average accuracy, recall and precision in several panoramic videos. Fig. 4 (a) illustrates the overall performance of three solutions. *CoLive* and *PanoSalNet* achieved no less than 90% accuracy in the average of all tested videos. The accuracy and recall of *LiveDeep* are around 80% yet the precision is only above 20%. The reason for this poor precision is because the number of selected tiles (40∼50 tiles during the tests) is far more than the size of the real viewport. *PanoSalNet* and *CoLive* jointly input the visual feature and individual's viewing behaviors to predict the viewport locations, which improves the accuracy. *CoLive* further integrates models with similar viewing behavior, thus improving the accuracy of new viewers and thus the average performance during testing. Fig. 4 (b)-(d) plot the average precision, recall and accuracy, respectively for different videos, which reveal a similar trend as 4 (a). More comprehensive results are given in our supplementary materials.

Table 1. Saliency feature detection performance

Approaches	CC	AUC	sAUC	KL
DatasetSaliency	0.2885	0.9941	0.7966	2.4531
PanoSalNet	0.2521	0.9512	0.7112	3.7425
CoLive	0.2047	0.9814	0.7847	3.6678



(a) The number of selected tiles with high definitions (b) Signaling overhead and bandwidth saving

Fig. 5. (a) The number of selected tiles with high definitions, (b) CoLive singal overhead and bandwidth savings.

Fig. 5 shows the bandwidth savings and signal overhead. As in Fig. 5 (a), *CoLive* performs well in terms of bandwidth savings since the selected tiles maintain around 20 when predicting viewport and this number is only half of *LiveDeep*. This is because *CoLive* has good performance in both accuracy and precision and viewers don't need to spend extra bandwidth resources to fetch unviewed tiles. Fig. 5 (b) shows the variation of control signaling overhead and bandwidth saving with the number of concurrent viewers. Because updating model parameters in *CoLive* requires the interactions between the video server and edge server, the extra bandwidth consumed by updating parameters cannot be neglected. However, compared with the bandwidth savings on prediction, such signal overhead is acceptable to the system.

[6]The source code is available at https://github.com/EricPengShuai/CoLive.

Processing Time. To support smooth video streaming, we also test the processing time of saliency feature detection and viewport prediction modules in *CoLive*. Table 2 shows the processing time of our model trained in different epochs for different videos. The saliency feature detection time increases with the epoch of training, but it is smaller than the average streaming latency tested in our supplementary materials. This indicates that the server can output the saliency map before the viewer requests the corresponding content, which avoids the latency introduced by detecting saliency feature. The processing time of viewport prediction also reveals a linear growing trend when the epoch increases. But even for the case of epoch equals 15, it is still less than the video chunk length (2s) in our system. Therefore, *CoLive* can support the smooth video streaming.

Table 2. Processing time (s) of the saliency detection (termed as *Sal*) and viewport prediction (termed as *View*) in *CoLive*

Video	Epoch=5		Epoch=10		Epoch=15	
	Sal	*View*	*Sal*	*View*	*Sal*	*View*
Conan1	0.190	0.542	0.379	1.083	0.569	1.611
Skiing	0.195	0.540	0.386	1.075	0.587	1.610
Alien	0.193	0.542	0.386	1.084	0.576	1.633
Conan2	0.193	0.542	0.387	1.085	0.579	1.637

5. CONCLUSION

In this paper, we propose *CoLive*, a novel viewport prediction method tailored for 360° live streaming service. To address the existing challenges of viewport prediction in 360° live streaming, *CoLive* migrates the video saliency detection module to the server side to avoid the processing delay. The edge computing server applies a hybrid model to perform the online viewers' future viewport prediction. Moreover, a collaborative online learning scheme is then proposed, which allows viewers' models to share their parameters to a central model. The experiment results show that *CoLive* outperforms several state-of-the-art solutions in terms of prediction accuracy, bandwidth savings and has reasonable signal overheads and processing time.

6. REFERENCES

[1] A. Yaqoob, T. Bi and G. -M. Muntean, "A Survey on Adaptive 360° Video Streaming: Solutions, Challenges and Opportunities," *in IEEE Commun. Surveys & Tutorials*, vol. 22, no. 4, pp. 2801-2838, Fourthquarter 2020.

[2] P. Maniotis and N. Thomos, "Tile-Based Edge Caching for 360° Live Video Streaming," *in IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4938-4950, Dec. 2021.

[3] O. Eltobgy, O. Arafa and M. Hefeeda, "Mobile Streaming of Live 360-Degree Videos," *in IEEE Transactions on Multimedia*, vol. 22, no. 12, pp. 3139-3152, Dec. 2020.

[4] L. Zhong et al., "A Multi-user Cost-efficient Crowd-assisted VR Content Delivery Solution in 5G-and-beyond Heterogeneous Networks," *in IEEE Transactions on Mobile Computing*, vol.99, no.99, pp. 1-1, Jan. 2022.

[5] A. Nguyen, Z. Yan, and K. Nahrstedt. "Your Attention is Unique: Detecting 360-Degree Video Saliency in Head-Mounted Display for Head Movement Prediction," *In Proc. of the 26th ACM international conference on Multimedia (MM '18)*, NY, USA, pp. 1190–1198.

[6] Y. Xu, et. al., "Gaze Prediction in Dynamic 360° Immersive Videos," *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5333-5342.

[7] C. Wu, R. Zhang, Z. Wang, et. al., "A spherical convolution approach for learning long term viewport prediction in 360 immersive video," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 14003-14040, 2020.

[8] X. Hou, et al., "Predictive Adaptive Streaming to Enable Mobile 360-Degree and VR Experiences," *in IEEE Transactions on Multimedia*, vol. 23, pp. 716-731, 2021.

[9] L. Xie, et al., "360ProbDASH: Improving QoE of 360 Video Streaming Using Tile-based HTTP Adaptive Streaming," *In Proceedings of the 25th ACM international conference on Multimedia (MM '17)*, NY, USA, pp. 315–323, 2017.

[10] A. T. Nasrabadi, et al., "Adaptive 360-Degree Video Streaming using Scalable Video Coding," *In Proceedings of the 25th ACM international conference on Multimedia (MM '17)*, NY, USA, pp. 1689–1697, 2017.

[11] M. Xu, et al., "Predicting Head Movement in Panoramic Video: A Deep Reinforcement Learning Approach," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2693-2708, Nov. 2019.

[12] X. Feng, et. al., "LiveDeep: Online Viewport Prediction for Live Virtual Reality Streaming Using Lifelong Deep Learning," 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), pp. 800-808, 2020.

[13] M. Wang, C. Xu, X. Chen, H. Hao, et al., "Differential Privacy Oriented Distributed Online Learning for Mobile Social Video Prefetching," *in IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 636-651, March 2019.

[14] X. Shi, Z. Chen, H. Wang, et. al., "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," *MIT Press*, 2015.

[15] C. Wu, Z. Tan, Z. Wang, et. al., " A Dataset for Exploring User Behaviors in VR Spherical Video Streaming", *In Proceedings of the 8th ACM on Multimedia Systems Conference (MMSys'17)*. NY, USA, pp. 193–198, 2017.

[16] A. Nguyen, et al., " A saliency dataset for 360-degree videos," *In Proc. of the 10th ACM Multimedia Systems Conference (MMSys '19)*, NY, USA, pp. 279–284, 2017.

[17] Z. Bylinskii, et al., "What Do Different Evaluation Metrics Tell Us About Saliency Models?," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 3, pp. 740-757, 1 March 2019.