Hindawi
Mobile Information Systems
Volume 2021, Article ID 8501990, 12 pages
https://doi.org/10.1155/2021/8501990

Research Article

Lightweight Neural Network-Based Viewport Prediction for Live VR Streaming in Wireless Video Sensor Network

Xiaolei Chen,[1] **Baoning Cao**,[1] **and Ishfaq Ahmad**[2]

[1]*College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou, China*
[2]*Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, TX, USA*

Correspondence should be addressed to Xiaolei Chen; chenxl703@lut.edu.cn

Received 19 August 2021; Revised 7 October 2021; Accepted 16 October 2021; Published 9 November 2021

Academic Editor: Xingsi Xue

Live virtual reality (VR) streaming (a.k.a., 360-degree video streaming) has become increasingly popular because of the rapid growth of head-mounted displays and 5G networking deployment. However, the huge bandwidth and the energy required to deliver live VR frames in the wireless video sensor network (WVSN) become bottlenecks, making it impossible for the application to be deployed more widely. To solve the bandwidth and energy challenges, VR video viewport prediction has been proposed as a feasible solution. However, the existing works mainly focuses on the bandwidth usage and prediction accuracy and ignores the resource consumption of the server. In this study, we propose a lightweight neural network-based viewport prediction method for live VR streaming in WVSN to overcome these problems. In particular, we (1) use a compressed channel lightweight network (C-GhostNet) to reduce the parameters of the whole model and (2) use an improved gate recurrent unit module (GRU-ECA) and C-GhostNet to process the video data and head movement data separately to improve the prediction accuracy. To evaluate the performance of our method, we conducted extensive experiments using an open VR user dataset. The experiments results demonstrate that our method achieves significant server resource saving, real-time performance, and high prediction accuracy, while achieving low bandwidth usage and low energy consumption in WVSN, which meets the requirement of live VR streaming.

1. Introduction

In recent years, with the increasing demand for immersive multimedia experiences, virtual reality (VR) video streaming (a.k.a., 360-degree video streaming) has become increasingly popular. Main industry suppliers have released a variety of head-mounted display (HMD) devices. Many video content providers have launched many high-definition VR video content, including military, education, real estate, retail, entertainment, healthcare and communications, and other fields. In order to further provide users with a more immediate immersive experience, VR streaming media has made preliminary attempts at live events such as sports events and news sites. Unlike traditional video streaming media, VR streaming media can provide users with a full range of scenes, including a 360-degree horizontal and 180-degree physical space view of the viewer's location, so that the viewer can change arbitrarily during playback and get an immersive viewing experience and sense of interaction.

A 360-degree video has high-resolution characteristics. Under current conditions, 4K resolution is the minimum requirement. In the future, resolutions of 6K or even higher may be required. At the same time, because the field of view (FoV) covered by 360-degree video is up to 360°, compared with traditional 2D video with a field of view less than 50°, the data volume of 360-degree video is more than 5 times larger than that of two-dimensional video at the same resolution and video length. Faced with such a huge amount of data, under the existing wireless network bandwidth conditions, even if the latest H.265/HEVC and other standards are used for compression and encoding, it is necessary to transmit a 360-degree video with sufficient resolution and covering a complete 360-degree scene. It will take up too much bandwidth resources, which will cause video buffering

delay, thereby affecting the user experience. In addition, in the VR live scene, live streaming also put forward strict real-time requirements.

The most advanced VR streaming research mainly focuses on viewport prediction [1–16]. The existing solutions [1, 11–13] suggest prefetching all the tiles of each segment, and higher quality of prefetching predicts the tiles in the viewport. Most existing viewport prediction algorithms can be divided into trajectory-based [1–6] and content-based [7–16] methods. However, these methods are either difficult to achieve sufficient prediction accuracy, or they occupy a large amount of server resources in actual deployment and cannot achieve excellent real-time performance due to the complexity of their own algorithms. Therefore, they cannot support live VR streaming well. In addition, there are some studies that try to alleviate the delay problem in VR live from other aspects. Sun et al. [17] used the idea of flocking to improve the performance of both prediction of FoV and caching on the edge servers for live 360-degree video streaming, and Chen et al. [18] proposed an event-driven stitching algorithm to improve the stitching speed of 360-degree videos.

To save server resources and solve the real-time prediction accuracy and bandwidth challenges effectively, we propose a lightweight neural network-based viewport prediction method for live virtual reality streaming in the wireless video sensor network (WVSN). The method uses an alternate and hybrid deep learning method to achieve accurate, real-time, low server resource consumption and low bandwidth usage viewport prediction. Specifically, on the one hand, we use a GhostNet module of compressed channel (C-GhostNet) to analyse users' preferences for video content. The module compresses the feature map channel of the lightweight network GhostNet [19], which can significantly reduce the parameters and prediction time and improve the prediction accuracy. On the other hand, in order to further enhance the perception of user's view trajectory, we propose an improved gate recurrent unit (GRU) module (GRU-ECA). By embedding an efficient attention mechanism method ECANet [20] into GRU [21] network, the module allocates different weights to user's view in different time steps, so as to further improve prediction accuracy without significantly increasing prediction time. Finally, we combine the prediction results of the two modules to generate the final user viewport. In addition, in view of the characteristics of live VR streaming, i.e., online video generation, no historical user data and real-time data, we cannot use the traditional workflow of first training and then prediction for viewport prediction. On the contrary, referring to LiveDeep [15], we use an alternate method, i.e., when the user watches the video, we collect the user's preference for the video content and the user's view trajectory information, train in the current video segment, and infer the user's viewport in the next video segment and constantly update the model during the whole session.

It is worth noting that WVSN is a system containing spatially distributed wireless video sensors (WVSs), and recent works on WVSN include [22–24]. The three main modules of WVS are image sensing, video compression, and wireless transmission. The wireless video sensor network operates under limited energy supply. The available energy will affect the final video quality and the service life of the system. Because bandwidth and energy consumption are proportional to the amount of transmitted video, the proposed method in this work can also save the energy consumption of WVSN.

The main contribution of this work lies in two folds.

We use the lightweight module C-GhostNet, GRU, and ECANet networks with fewer parameters, thereby reducing the prediction time of the entire model and reducing the resource consumption of the server.

We use the GRU-ECA module to predict the user's viewport trajectory and use the C-GhostNet module to analyse the user's preference for video content. The proposed modules can improve the prediction accuracy of viewport prediction, while obtaining lower bandwidth usage and lower energy consumption in WVSN.

2. Background and Related Work

2.1. Live VR Streaming. Figure 1 illustrates the overall workflow of live VR video streaming with viewport prediction. The 360-degree camera captures the panoramic video and delivers the streaming to the server. On the server side, the video is divided into segments by the packer, and then, they are distributed to the client through the content distribution network (CDN) using the optimized viewport-adaptive 360-degree video streaming mechanism [25]. To reduce the bandwidth, the part inside the predicted viewport is coded with high quality, and the part outside the predicted viewport is coded with low quality [26–29]. In this way, users can still watch low-quality videos to avoid interruption of the VR video experience even if the predicted viewport is not accurate.

2.2. Viewport Prediction. Viewport prediction can be divided into two categories: trajectory-based and content-based methods. The trajectory-based method uses the user's historical head rotation data to predict the future viewport. Qian et al. [1] proposed three methods: simple average, linear regression, and weighted linear regression to predict the user's future viewport. Battle et al. [2] used the Markov chain model to predict the viewport with the highest probability for the user to watch at the next moment. Jiang et al. [3] applied a model based on long short-term memory (LSTM) to predict future head rotations. Jamali et al. [4] used a LSTM encoder-decoder network to perform sequence-to-sequence prediction. Nasrabadi et al. [5] proposed a clustering-based method to estimate the user's future viewport. Jiang et al. [6] interpreted the original rotation amount as a sine value to reduce the prediction error of the yaw direction. These trajectory-based methods may be used in live because they only require real-time data from the current video session. However, it is difficult to achieve acceptable prediction accuracy using only the user trajectory, especially in a long span of a few seconds (i.e., the

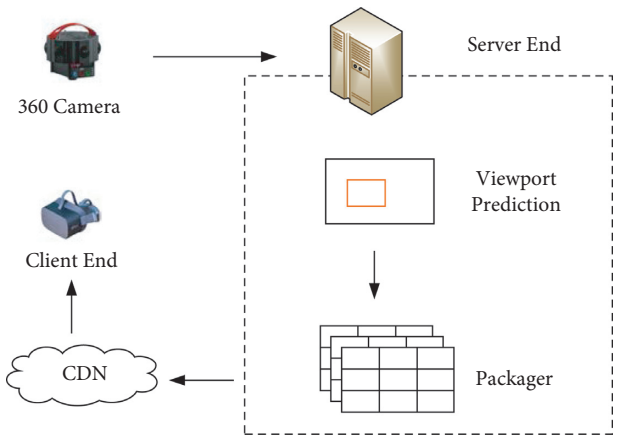


Figure 1: Overall workflow of live VR streaming.

length of the video buffer) because the user is likely to change their head movement.

Content-based methods use rotation information and video content as features. Dasari et al. [7] proposed a PARSEC method that combines viewport prediction with superresolution to improve the quality of user experience. Kan et al. [8] proposed a DRL-based rate adaptation algorithm qualified for learning a policy to strike an optimal trade-off between the video quality, the risk of rebuffering, and the smoothness of video quality by selecting the proper bitrate of tiles. Previous works [9–12] designed a hybrid architecture of CNN and LSTM models. They used convolutional neural networks (CNN) and LSTM to extract video content features from saliency maps, original images, or patterns of motion from historical rotation information. Feng et al. [13] used optical flow and Gaussian mixture model for motion detection and feature tracking and then used a dynamic user interest model to generate the user's future viewport. Feng et al. [14] used a CNN-based model to predict the future viewport in the live streaming by modifying the training/inferring process. On the basis of earlier work [14], Feng et al. [15] proposed a hybrid model to improve prediction accuracy. Feng et al. [16] achieved low bandwidth consumption by performing real-time semantic-based detection and tracking at the object level. All these methods will burden the actual deployment in the actual system because they consume too much server resources.

3. Materials and Methods

In this section, we discuss the design and implementation of the lightweight neural network C-GhostNet/GRU-ECA for live viewport prediction in our method.

3.1. Model Architecture. The main goal of the method in this study was to achieve lightweight live VR streaming viewport prediction in WVSN. The overall architecture of our method is shown in Figure 2. We use the C-GhostNet model to analyse the user's preference for video content and use the GRU-ECA model to perceive the user's viewport trajectory, and then, we merge the results of the two methods into the final predicted viewport. The entire prediction process starts with the input video segment and ends with the output viewport prediction result, and through user feedback, the viewport is trained and predicted for each segment to update the model.

According to the workflow shown in Figure 2, a series of video segments will be obtained after preprocessing the input video. For the prediction of the first video segment, because the training image is not labelled, the model cannot be trained. Our solution is to let the C-GhostNet module use random weights to predict the user's viewport and transmit the tiles in the predicted viewport with high quality and transmit the tiles outside the predicted viewport with low quality. After the user watches the video segment, the predicted viewport is compared with the actual viewport, and the loss value is calculated. Then, use the loss value to update the C-GhostNet module and predict the second video segment. After the user has watched the first video segment, the corresponding viewport trajectory will be generated, which is input into the GRU-ECA model for training and prediction. Combine its prediction result with the prediction result of the C-GhostNet model of the second video segment as a new prediction viewport. And so on, until all video segments are predicted.

3.2. Data Collection. The live viewport prediction method obtains the data that needs training by collecting historical video and user data. However, in the live VR streaming scene, due to the lack of user viewing history, the collected training images are not labelled. Therefore, we must wait for the user to watch the corresponding video segment and then get the corresponding viewport and user viewport sequence as feedback. Then, based on user feedback, we compare all tiles with the actual user viewport. Label the tiles corresponding to the actual viewport as "interested" and label the remaining tiles as "not interested."

For video preprocessing, we use the following steps: first, segment the live VR streaming and input each video segment as a basic processing unit into the C-GhostNet module for training and inference. To deal with the computational resource consumption and processing delay caused by a large number of video frames, we sequentially perform temporal downsampling and spatial downsampling on each video segment. In temporal downsampling, we sample each video segment into fixed k frames. In spatial downsampling, in order to process videos of different resolutions and facilitate the construction of network models, we adjust the size of k frames to 160×160. For each frame, we evenly divide it into $x \times y$ tiles. In the end, $k \times x \times y$ tiles will be output in each VR frequency band. In this article, referring to LiveDeep [15], video segmentation duration is set to 2 s, x and y are both set to 5, and k is set to 8. After preprocessing the VR video with an input of high resolution, each video segment will finally output 200 tiles with a resolution of 32×32 for the training and inference of the C-GhostNet module.

After the user watches a video segment, the corresponding center track of the user's viewport is obtained from the head movement data. Similar to the preprocessing of

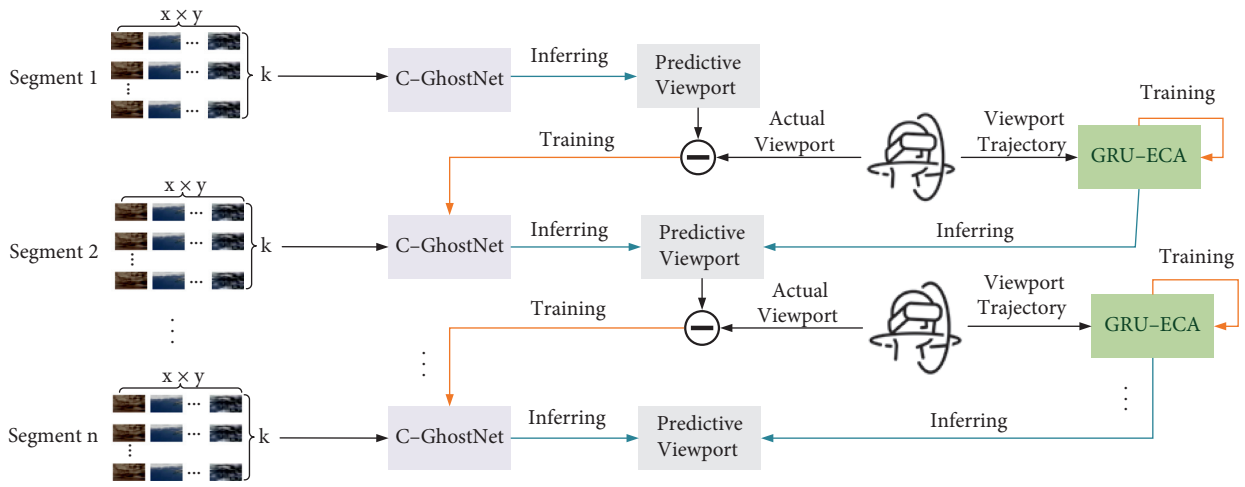


Figure 2: Overall viewport prediction framework.

image data, the downsampling of the parameter k is performed on the center trajectory of the user's viewport of each video segment. Finally, the sampling result is input into the GRU-ECA module for the prediction of the user's viewport trajectory.

3.3. C-GhostNet Module. In view of the calculation delay and server resource consumption problems caused by the traditional CNN used in the previous viewport prediction work, we choose to use C-GhostNet for the VR video content feature extraction backbone network. GhostNet is a new type of lightweight convolutional neural network that uses Ghost module to generate more feature maps from fewer parameters, which can reduce the computational cost of the convolutional layer while maintaining high recognition performance.

3.3.1. Ghost Module. After visualizing the convolutional feature map, it is found that the output feature map of the original convolutional layer usually contains a lot of redundancy, and some of them may be similar to each other. Therefore, the output feature map can be obtained from a few original feature maps through some cheap conversion "ghost," as shown in Figure 3. Specifically, m original feature maps $Y' \in \mathbb{R}^{h' \times w' \times m}$ are generated using one convolution:

$$Y' = X * f', \tag{1}$$

where $*$ is the convolution operation and $f' \in \mathbb{R}^{c \times k \times k \times m}$ is the convolution kernel used and further performs a series of cheap linear operations on each original feature in Y' to generate s ghost feature maps:

$$y_{ij} = \Phi_{i,j}(y'_i), \quad \forall i = 1, \ldots, m, \; j = 1, \ldots, s, \tag{2}$$

where y'_i is the i-th original feature map in Y' and $\Phi_{i,j}$ in the above function is the j-th linear operation, which is used to generate the j-th ghost feature map y_{ij}. The final $\Phi_{i,s}$ is the identity mapping used to preserve the original feature map as shown in Figure 3. Using cheap operations, we can obtain $n = m \cdot s$ feature maps $Y = [y_{11}, y_{12}, \ldots, y_{ms}]$ as the output data of the Ghost module. The linear operation $\Phi_{i,j}$ runs on each channel, so the amount of calculation is much less than that of ordinary convolution.

3.3.2. Ghost Bottlenecks. Taking advantage of the Ghost module, Ghost bottleneck (G-bneck) is specially designed for small CNN in GhostNet. Ghost bottleneck integrates multiple convolutional layers and shortcuts. Ghost bottleneck is mainly composed of two-stacked Ghost modules. The first Ghost module is used as an expansion layer to increase the number of channels, and the second Ghost module reduces the number of channels to match the shortcut path. Then, use shortcut to connect the input and output of the two Ghost modules.

3.3.3. C-GhostNet. Figure 4 shows the GhostNet network architecture diagram. GhostNet is mainly composed of a bunch of Ghost bottlenecks, which are based on the Ghost modules. The first layer is a standard convolutional layer with 16 convolution kernels, and then, a series of Ghost bottleneck with gradually increasing channels. Finally, the global average pooling layer and convolution layer are used to convert the feature map into a 1280-dimensional feature vector for final classification. The SE module is also used in the residual layer in some Ghost bottleneck. According to the actual demand predicted by the viewport, only two categories of "interested" and "not interested" are output. However, GhostNet has too many channels in each layer, resulting in a waste of computer resources, which is not in line with our actual needs. Therefore, we adjust the GhostNet network structure to suit our task. We remove the original global average pooling layer and reduce the number of channels per layer. The information of each level of C-GhostNet is shown in Table 1. It can be seen that the network parameters of the original GhostNet after channel compression are reduced to 1,307,970. If channel compression is not used, the parameters of the original GhostNet are 3,904,070.

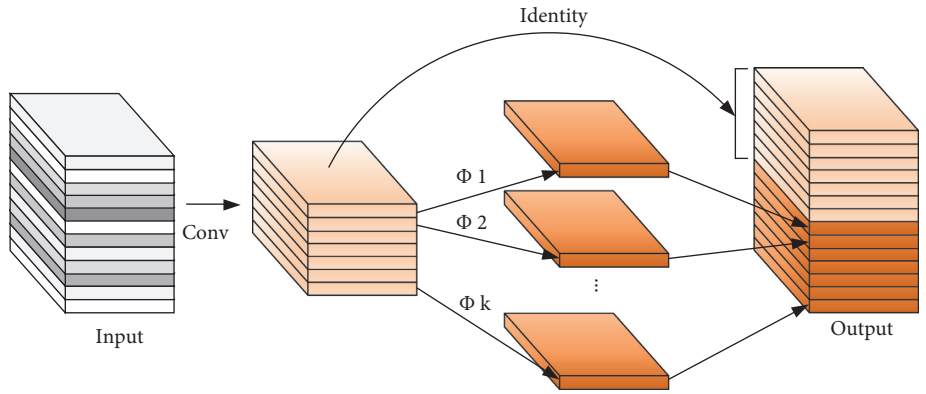


Figure 3: Ghost module for outputting the same number of feature maps [19].

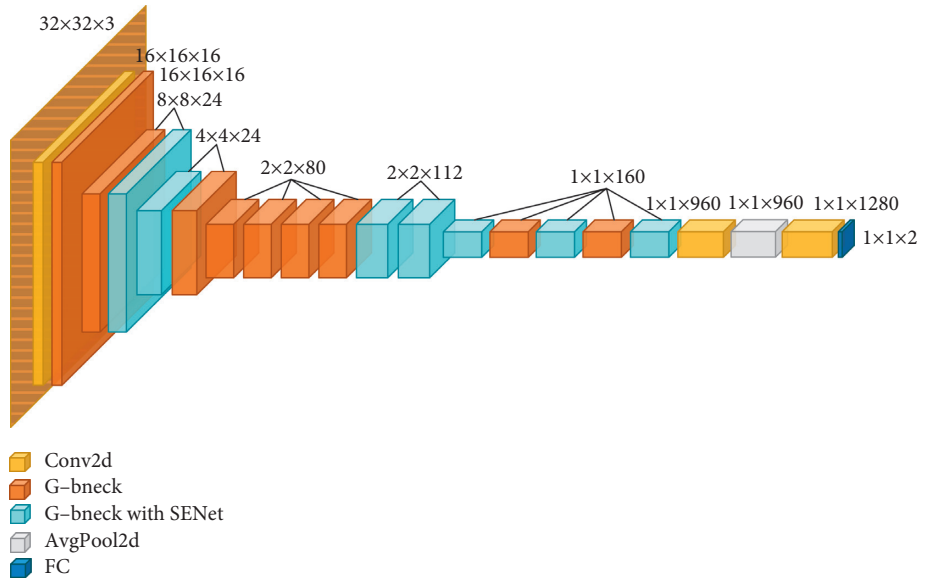


Figure 4: GhostNet network architecture diagram.

Table 1: C-GhostNet each layer information.

Operator	Output	SE	Stride	Parameters
Conv2d 3×3	$16^2 \times 8$	—	2	232
G-bneck	$16^2 \times 8$	—	1	168
G-bneck	$8^2 \times 12$	—	2	1,012
G-bneck	$8^2 \times 12$	—	1	744
G-bneck	$4^2 \times 20$	1	2	3,428
G-bneck	$4^2 \times 20$	1	1	3,386
G-bneck	$2^2 \times 40$	—	2	6,840
G-bneck	$2^2 \times 40$	—	1	4,910
G-bneck	$2^2 \times 40$	—	1	4,538
G-bneck	$2^2 \times 40$	—	1	4,538
G-bneck	$2^2 \times 56$	1	1	45,336
G-bneck	$2^2 \times 56$	1	1	78,232
G-bneck	$1^2 \times 80$	1	2	97,644
G-bneck	$1^2 \times 80$	-	1	42,040
G-bneck	$1^2 \times 80$	1	1	157,440
G-bneck	$1^2 \times 80$	-	1	42,040
G-bneck	$1^2 \times 80$	1	1	157,840
Conv2d 1×1	$1^2 \times 480$	—	1	39,360
Conv2d 1×1	$1^2 \times 640$	—	1	615,680
FC	$1^2 \times 2$	—	—	2,562
Total	—	—	—	1,307,970

3.4. GRU-ECA Module. When users watch 360-degree videos, their future viewports are related to many factors, such as video content and scene details. On the one hand, users are easily attracted by some features in the video. On the other hand, due to different movement patterns, the historical head movement is also a key factor in predicting the user's future viewport. In previous work, the LSTM model is often used to predict the user's head movement trajectory to quickly respond to the rapid switching of user preferences, but this method is limited to the correct prediction of the user's viewport. Therefore, we proposed the GRU-ECA module. It hardly increases the delay overhead while improving the prediction accuracy of the user's viewport.

3.4.1. GRU. Both GRU and LSTM, as variants of recurrent neural network (RNN), can alleviate the gradient explosion and gradient disappearance in long-term memory and back propagation. On the other hand, GRU further simplifies the structure of LSTM. Compared with LSTM, GRU has one less gating unit, so it has fewer parameters, which can speed up the whole training and prediction process, reduce the consumption of computing resources and still achieve good results.

3.4.2. ECANet. The channel attention mechanism has shown great potential in improving the performance of neural networks. However, most of the existing methods are dedicated to the development of more complex attention modules to obtain better performance, which inevitably increases the complexity of the model. In order to overcome the contradiction between performance and complexity, ECANet proposes an efficient channel attenuation (ECA) module; this module only requires a small number of parameters to obtain significant performance gains. In ECANet, the channel dimension is not reduced to facilitate the learning of effective channels. Considering only the interaction between each channel and its k neighbors, the weight of the output y_i can be calculated as

$$w_i = \sigma\left(\sum_{j=1}^{k} \alpha_i^j y_i^j\right), \alpha_i^j \in \Omega_i^k, \tag{3}$$

where Ω_i^k represents the set of k adjacent channels of y_i. Obviously, equation (3) captures local cross-channel interaction, and this locality constraint avoids interaction across all channels, thereby allowing higher model efficiency. In order to further improve network performance, ECANet uses a one-dimensional convolution method of sharing weights, that is, the weights of each group are exactly the same, which greatly reduces the amount of parameters, from the original $k \times C$ (C is the number of channels) to k.

3.4.3. GRU-ECA. The head movement prediction problem is a nonlinear regression problem, in which the historical head movement is independent variables, and the future head movement is the dependent variable. We propose the GRU-ECA module to solve this prediction problem, and the proposed module structure is shown in Figure 5. First, take the user's viewport trajectory as input and use GRU to extract the features in the user's viewport sequence. Then put the output features of GRU into ECANet to obtain the features with different time steps, so that the network can better learn the implicit relationship between the sequences. Finally, the predicted viewport trajectory is output through a fully connected layer. In order to make ECANet adapt to the input of one-dimensional data, the AdaptiveAvgPool2d is modified to AdaptiveAvgPool1d.

4. Results and Discussion

A lot of experiments and analyses have been done for the proposed method in this section. First, we introduce an open VR user head movement dataset. Secondly, we present our experimental environment and related settings. Finally, we carry out the epoch impact analysis, ablation study, and comparison of different advanced methods.

4.1. Dataset. The open dataset [30] is a VR head movement dataset composed of 18 videos in 5 categories watched by 48 users. It not only provides records of user head movement data but also includes the videos used in the experiment. In order to evaluate the performance of our method, we select 8 representative test videos from the dataset for experimental testing, as shown in Table 2. These 8 videos include sports, performances, talk shows, and documentary videos. Videos 3 and 8 were captured by multiple cameras. The backgrounds in videos 1, 4, and 5 are static, and the dynamic backgrounds in the remaining videos are caused by camera movement or switching between multiple cameras.

4.1.1. Experimental Environment. Our experiments are conducted in graphics processing unit (GPU) mode and use CUDA to accelerate deep learning processing. The detailed configuration of the computer used in the experiment is as follows: RAM 32G, an Intel(R) Core(TM) i3-9100F CPU, an Nvidia GTX1080Ti GPU, and the operating system used is 64 bit Ubuntu 16.04. The scripting language selected for the experiment is *Python*, and the OpenCV image processing library and the Pytorch deep learning framework are installed.

4.1.2. Implementation Details. In the C-GhostNet module, we set the image batch for training and inference to 200. Different from the general CNN epoch setting, we set the epoch to 10 to meet the real-time demand brought by VR live. In addition, we use CrossEntropyLoss as the loss function.

In the GRU-ECA module, we set the hidden_size (i.e. number of hidden layer nodes) of the GRU to 64 and the num_layers (i.e. number of GRU layers) to 2.

4.1.3. Evaluation Metrics. To fully evaluate the performance of our method, we use 4 evaluation metrics, namely, prediction accuracy, bandwidth usage, prediction time, and

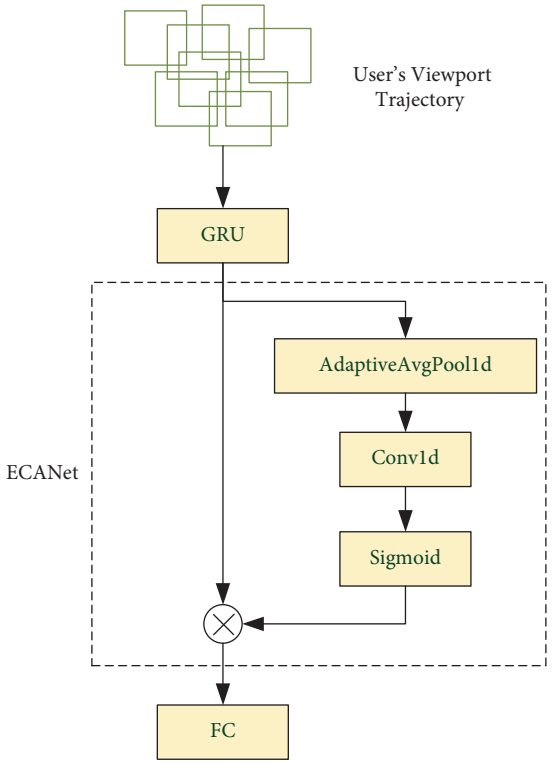


Figure 5: GRU-ECA module structure.

Table 2: Test videos.

No.	Video name	Category	Background	Cameras
1	Cooking	Performance	Static	1
2	Football	Sport	Dynamic	1
3	Fighting	Sport	Dynamic	2
4	Anitta	Performance	Static	1
5	RioVR	Talk show	Static	1
6	Front	Sport	Dynamic	1
7	Falluja	Documentary	Dynamic	1
8	Rhinos	Documentary	Dynamic	2

model parameters. First of all, prediction accuracy is an important factor that affects the user's experience of watching VR videos. For each frame in the video, as shown in Figure 6, the blue rectangle represents the actual viewport, and the red grids represents the predicted viewport. We compare the predicted viewport with the actual viewport, and when the actual viewport is completely covered by the predicted viewport, the prediction of the frame is considered correct. If the prediction is not correct, the user will only be able to watch low-quality video. Then, the prediction accuracy is obtained by calculating the ratio of the number of correct frames in the video to the number of all frames, which is different from the intersection over union (IoU) metric used to evaluate accuracy in other works. Secondly, reducing the bandwidth usage in the VR streaming is the basic goal of viewport prediction, and it is also an important metric reflecting the energy consumption in WVSN. To facilitate the calculation, we ignore the bandwidth consumption of the tiles outside the viewport and define the proportion of all tiles corresponding to the predicted viewport as the bandwidth usage. Thirdly, the prediction time evaluates the unique real-time requirements brought about by live VR. We will use the data collection to the viewport prediction as a complete prediction. According to the data segmentation setting in data collection, a smooth real-time streaming experience can be obtained only when the prediction time of each segment is less than 2 s (ignore video stitching time [31]). Finally, we use model parameters to measure the consumption of server resources by the backbone network.

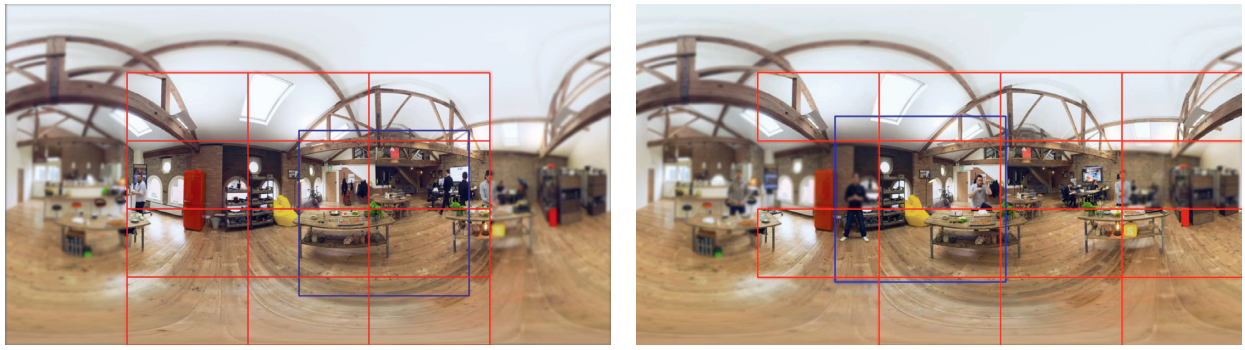
4.1.4. Ablation Study. To illustrate the effectiveness of each module in our method, we conducted an ablation study on our method, i.e., to evaluate the prediction accuracy, prediction time of each video segment, and parameters of VGG13, GhostNet, C-GhostNet, GRU, GRU-ECA, and C-GhostNet + GRU-ECA, respectively.

Table 3 shows the results of the ablation experiment. The results show that compared with VGG13, C-GhostNet further improves the prediction accuracy and significantly reduces parameters and prediction time, ranging from 64% to 98%. Compared with GhostNet, C-GhostNet reduces the parameters by 66.5% without decreasing the prediction accuracy. The prediction accuracy rate of GRU + ECA is between 35% and 56%, which is an increase of 2.1% compared with GRU, whereas the prediction time and parameters have not increased significantly. The prediction accuracy of C-GhostNet + GRU-ECA is between 86% and 99%, achieving consistent high prediction accuracy in different test videos. Figure 7 shows the prediction accuracy of each segment of video 1 and video 8. We observe that the prediction accuracy of C-GhostNet will decline sharply with the rapid movement of user's view, especially in the video with dynamic background, and the addition of GRU-ECA module can effectively resist the interference caused by this situation. Therefore, compared with a single module, the C-GhostNet + GRU-ECA combined module can significantly improve the prediction accuracy.

Figure 8 shows the cumulative distribution of the average prediction time of each segment of 8 test videos using our method. We can observe that the processing time of most video segments is between 280 ms and 700 ms, and the prediction time of all video segments are less than 800 ms and much less than 2 s, which means that the viewport prediction can be completed during the video playback of the current segment without delay to the next segment, and a smooth VR streaming experience is supported.

4.1.5. The Impact of Epoch. We further evaluate the performance of our method under different epochs, which is one of the important parameters that affect the prediction accuracy and prediction time of the entire network model.

Generally speaking, more epoch helps to achieve higher prediction accuracy but also with a longer prediction time. In order to analyse the impact of epoch on the performance of the entire network model, we set three epochs for the training process, namely, 6, 10, and 15. The results are shown

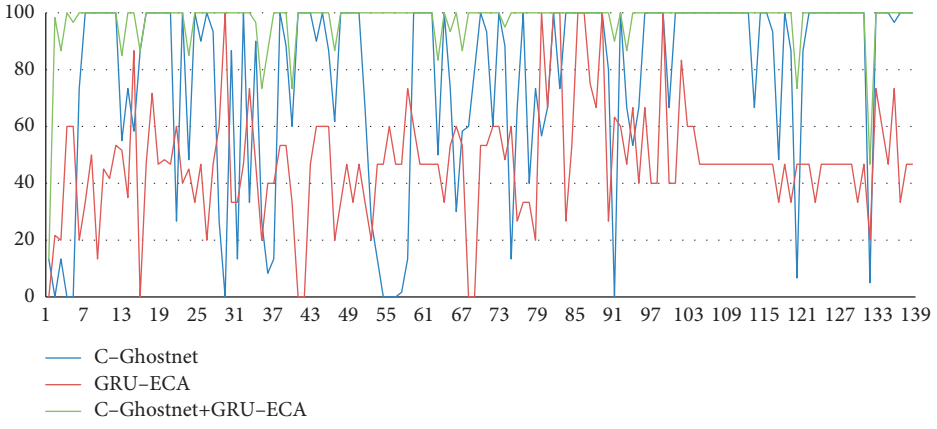


(a) (b)

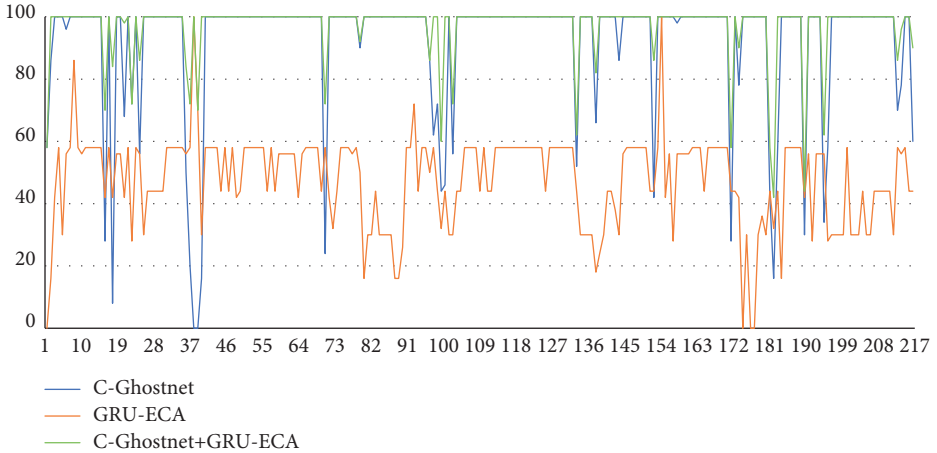
Figure 6: Snapshot of the prediction results. (a) Correct prediction. (b) Wrong prediction.

Table 3: Performance for ablation study.

Network	Accuracy (%)	Average time (s)	Parameters
VGG13	[43.0, 96.9]	0.41	128,959,042
GhostNet	[64.1, 97.8]	0.29	3,904,070
C-GhostNet	[64.9, 97.5]	0.28	1,307,970
GRU	[34.7, 55.1]	0.16	36,160
GRU-ECA	[35.2, 55.7]	0.17	36,165
C-GhostNet + GRU-ECA	[86.2, 98.3]	0.46	1,344,135



(a)



(b)

Figure 7: The prediction accuracy of each video segment of C-GhostNet + GRU-ECA. The X-axis shows the segment number in the video, and the Y-axis shows the prediction accuracy (%). (a) Video 1. (b) Video 8.

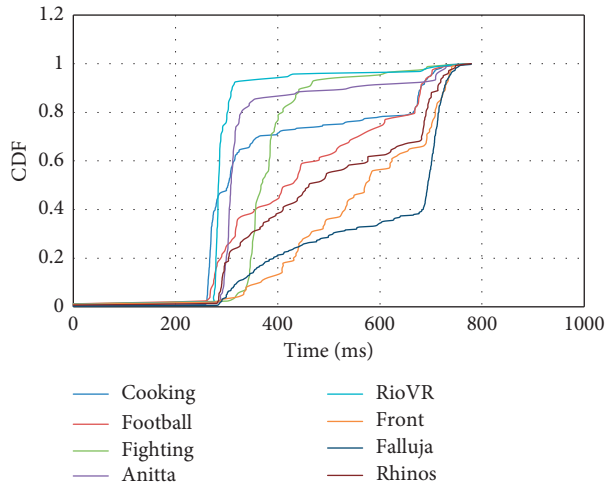


Figure 8: Demonstration of average prediction time overhead for 8 test videos. *X*-axis indicates the processing time (ms) for one video segment and *Y*-axis indicates the cumulative distribution value.

in Table 4, including the prediction accuracy, the average prediction time of each segment, and each the maximum prediction time of the segment. The results show that, compared with epoch = 10, when epoch = 6, the prediction accuracy of the entire network model is significantly reduced, and the prediction time is correspondingly reduced. When epoch = 15, the prediction time increases, but the improvement of prediction accuracy is very limited. This is because in our method, the complexity of the entire model is limited, and an epoch that is too large will not bring about a significant improvement in the prediction accuracy but will increase the extraprediction time. Therefore, we recommend setting epoch to 10, which can achieve higher prediction accuracy while meeting the delay overhead.

Table 4: Performance for different epochs.

Epoch	Accuracy (%)	Average time (s)	Maximum time (s)
6	88.5	0.39	0.61
10	94.4	0.47	0.78
15	94.9	0.66	0.92

4.2. Comparison of Different Advanced Methods

4.2.1. Prediction Accuracy. Table 5 shows the average prediction result of all video segment in the test video. The results show that both our method and Motion [13] have achieved a high prediction accuracy rate, which is maintained above 86% overall. The overall prediction accuracy of LiveObj [16] is low, ranging from 75% to 89%. The performance of the other methods is not stable, and the prediction accuracy in different videos varies greatly, and the maximum difference can reach 49.6%. Our method obtained the highest prediction accuracy in videos 1, 3, 4, 5, 6, and 8. By further analyzing the video content, we can find that videos 2, 6, and 7 contain a large number of dynamic background segments, and the number of scene switching is more than that of other videos; the prediction accuracy of our method in these videos is lower than that of other videos.

4.2.2. Prediction Time. Figure 9 shows the average prediction time per frame of randomly selected videos, where the backgrounds of video 1 and video 5 are static, and the backgrounds of video 2 and video 7 are dynamic. The results show that in different video types, the overall prediction time of LiveDeep [15] and LiveObj [16] is higher than that of our method, and the prediction time of LiveObj [16] for different frames is quite different. For video 1 and video 5, the prediction time in our method is concentrated around 40 ms. However, in video 2 and video 7, the prediction time in our method is between 35 ms and 97 ms and between 32 ms and 96 ms, respectively. These show that for dynamic background videos, our method requires more time to predict the viewport due to the user's interest in content and the rapid changes in the viewport, and the complexity of the video content will cause large differences in the prediction time of different frames. In general, our method obtains a lower prediction time, which is generally kept within 100 ms.

4.2.3. Parameters. Figure 10 shows the parameters of the different methods. The results show that traditional CNN is used in CNN [14] and LiveDeep [15], and the network structure is optimized. Therefore, compared with the original Alexnet and VGG13, parameters are reduced to a certain extent, but the overall number is above 10 M. The target detection network Yolov2 is used in LiveObj [16], and its parameter quantity is still at a high level. In our method, due to the use of the lightweight network C-GhostNet, its parameters are reduced by an order of magnitude compared with other methods, so the consumption of server resource is reduced to a greater extent in the prediction process.

4.2.4. Bandwidth Usage. Figure 11 shows the box plot of the bandwidth usage of different methods, which are evaluated based on the size of the prediction area of all videos and users

Table 5: Comparison of prediction accuracy (%) of CNN, LiveDeep, LiveObj, Motion, and our method (ours).

Method	Video 1	Video 2	Video 3	Video 4	Video 5	Video 6	Video 7	Video 8
LiveObj	88.9	75.6	86.2	80.0	80.5	81.4	87.2	87.1
LiveDeep	93.2	85.4	96.1	95.3	97.5	85.9	81.7	90.1
CNN	91.5	71.7	89.4	67.6	98.0	47.8	48.4	70.9
Motion	95.3	**94.2**	93.4	93.8	91.1	90.3	**91.1**	91.5
Ours	**96.5**	91.3	**97.6**	**97.7**	**98.3**	**91.8**	86.2	**95.8**

The bold values represent the highest prediction accuracy among different methods.

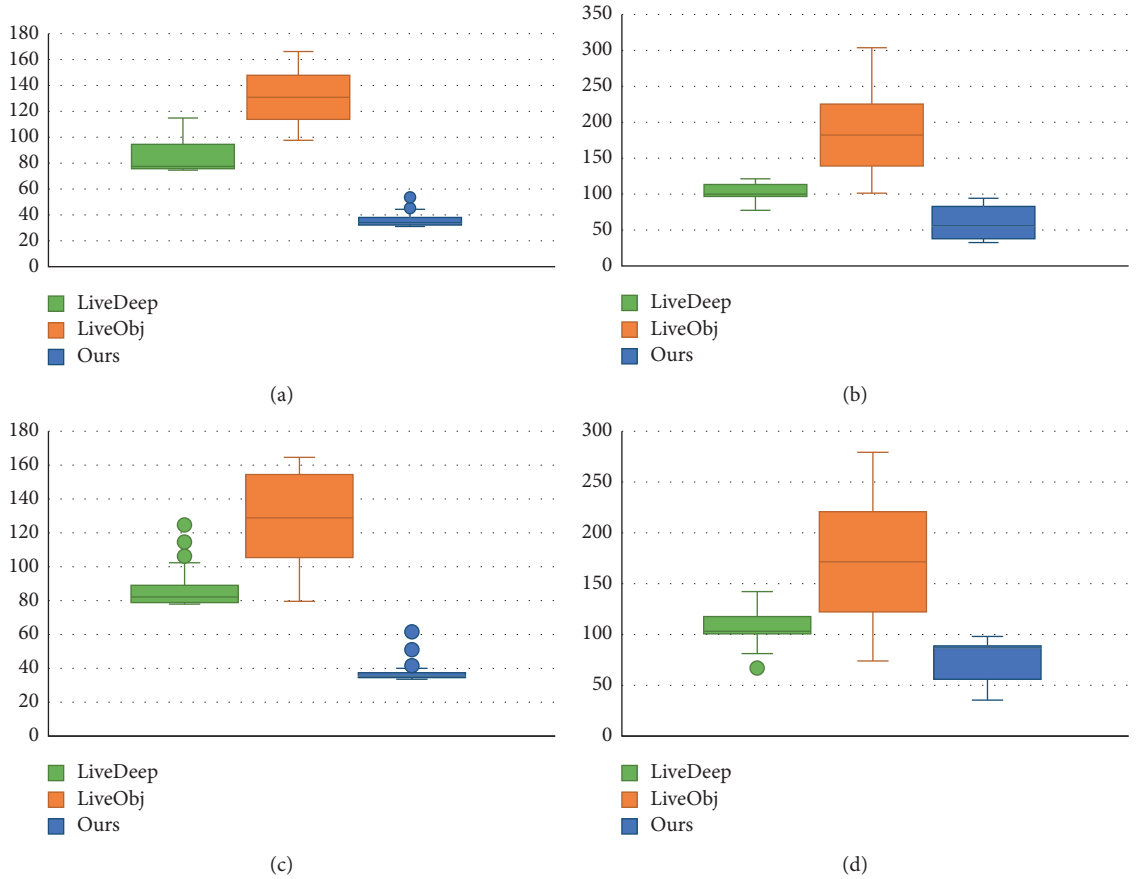


Figure 9: Comparison of prediction time of each frame of LiveDeep, LiveObj, and our method (Ours). The *Y*-axis shows the prediction time of each frame (ms). (a) Video 1. (b) Video 2. (c) Video 5. (d) Video 7.

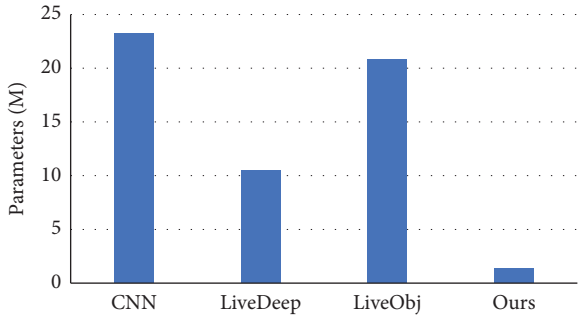


Figure 10: Comparison of parameters of CNN, LiveDeep, LiveObj, and our method (Ours).

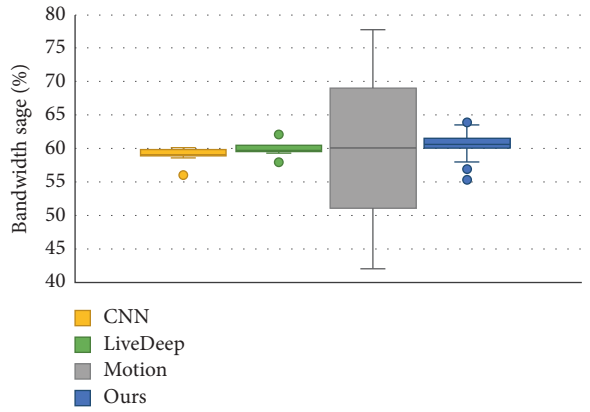


Figure 11: Comparison of bandwidth usage of CNN, LiveDeep, Motion, and our method (Ours).

on each frame. The results show that Motion can cause huge differences in bandwidth usage between different videos and users. The results of CNN [14], LiveDeep [15] and our method are relatively stable on most videos. Among them, the bandwidth usage of our method has increased slightly, but overall, it is still within an acceptable range.

In general, our method achieves higher prediction accuracy with less prediction time and parameters under the premise of lower bandwidth usage and energy consumption, so its overall performance is better than other methods.

5. Conclusions

In this article, we propose a lightweight neural network-based viewport prediction method for live virtual reality streaming in WVSN, which includes a C-GhostNet module and a GRU-ECA module for user content preference analysis and user's viewport trajectory perception. The C-GhostNet module can significantly reduce parameters and prediction time and can improve the prediction accuracy. The GRU-ECA module can further improve prediction accuracy through the attention mechanism, and at the same time, it has almost no effect on the prediction time and parameters of the overall model. Due to the contributions of these two modules, our method has achieved excellent performance on the dataset [30], especially in WVSN.

By analyzing the prediction accuracy and bandwidth usage in the test video, we notice that there are still two difficult problems to solve. The first problem is that in videos with dynamic background, especially with a large number of scenes switching, such as videos 2, 6, and 7, the prediction accuracy of this method is low due to the complexity of video content and the limitation of model. The second problem is that compared with other methods, the bandwidth utilization of our method still has room for further decline. We plan to further optimize these problems in future work combined with action recognition [32].

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China(nos. 61967012, 61866022, and 61861027).

References

[1] F. Qian, L. Ji, B. Han, and V. Gopalakrishnan, "Optimizing 360 video delivery over cellular networks," in *Proceedings of the 5th Workshop Things Cellular Oper, Appl. Challenges (ATC)*, pp. 1–6, London, UK, August 2016.

[2] L. Battle, R. Chang, and M. Stonebraker, "Dynamic prefetching of data tiles for interactive visualization," in *Proceedings of the 2016 International Conference on Management of Data*, pp. 1363–1375, San Francisco, CA, USA, July 2016.

[3] X. Jiang, Y.-H. Chiang, Y. Zhao, and Y. Ji, "Plato: learning-based adaptive streaming of 360-degree videos," in *Proceedings of the IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 393–400, Chicago, IL, USA, October 2018.

[4] M. Jamali, S. Coulombe, and A. Vakili, "LSTM-based viewpoint prediction for multi-quality tiled video coding in virtual reality streaming," in *Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, Sevilla, Spain, April 2020.

[5] A. T. Nasrabadi, A. Samiei, and R. Prakash, "Viewport prediction for 360 videos: a clustering approach," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, pp. 34–39, Istanbul Turkey, June 2020.

[6] X. Jiang, S. A. Naas, Y.-H. Chiang, S. Sigg, and Y. Ji, "SVP: sinusoidal viewport prediction for 360-degree video streaming," *IEEE Access*, vol. 8, pp. 164471–164481, 2020.

[7] M. Dasari, A. Bhattacharya, and S. Vargas, "Streaming 360-degree videos using super-resolution," in *Proceedings of the IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 1977–1986, Toronto, Canada, July 2020.

[8] N. Kan, J. Zou, and C. Li, "RAPT360: reinforcement learning-based rate adaptation for 360-degree video streaming with adaptive prediction and tiling," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–17, 2021.

[9] Y. Xu, Y. Dong, J. Wu et al., "Gaze prediction in dynamic 360° immersive videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5333–5342, Salt Lake City, UT, USA, June 2018.

[10] Q. Yang, J. Zou, K. Tang, C. Li, and H. Xiong, "Single and sequential viewports prediction for 360-degree video streaming," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, Sapporo, Japan, May 2019.

[11] X. Li, S. Wang, C. Zhu, L. Song, R. Xie, and W. Zhang, "Viewport prediction for panoramic video with multi-CNN," in *Proceedings of the IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pp. 1–6, Chengdu, China, June 2019.

[12] X. Chen, A. T. Z. Kasgari, and W. Saad, "Deep learning for content-based personalized viewport prediction of 360-degree VR videos," *IEEE Networking Letters*, vol. 2, no. 2, pp. 81–84, 2020.

[13] 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*, vol. 3, no. 2, pp. 1–22, 2019.

[14] X. Feng, Z. Bao, and S. Wei, "Exploring CNN-based viewport prediction for live virtual reality streaming," in *Proceedings of the IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 183–186, San Diego, CA, USA, December 2019.

[15] X. Feng, Y. Liu, and S. Wei, "LiveDeep: online viewport prediction for live virtual reality streaming using lifelong deep learning," in *Proceedings of the IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, pp. 800–808, Atlanta, GA, USA, March 2020.

[16] X. Feng, Z. Bao, and S. Wei, "LiveObj: object semantics-based viewport prediction for live mobile virtual reality streaming," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 5, pp. 2736–2745, 2021.

[17] L. Sun, Y. Mao, T. Zong, Y. Liu, and Y. Wang, "Flocking-based live streaming of 360-degree video," in *Proceedings of the 11th ACM Multimedia Systems Conference*, pp. 26–37, Istanbul, Turkey, May 2020.

[18] B. Chen, Z. Yan, H. Jin, and K. Nahrstedt, "Event-driven stitching for tile-based live 360 video streaming," in *Proceedings of the 10th ACM Multimedia Systems Conference*, pp. 1–12, Amherst, MA, USA, June 2019.

[19] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "GhostNet: more features from cheap operations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1580–1589, Seattle, Was, USA, June 2020.

[20] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "ECA-net: efficient Channel attention for deep convolutional neural networks," 2019, http://arxiv.org/abs/1910.03151.

[21] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (GRU) neural networks," in *Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 1597–1600, Boston, MA, USA, August 2017.

[22] X. Xue and J. Zhang, "Matching large-scale biomedical ontologies with central concept based partitioning algorithm and adaptive compact evolutionary algorithm," *Applied Soft Computing*, vol. 106, pp. 1–11, 2021.

[23] X. Xue, C. Yang, C. Jiang, P.-W. Tsai, G. Mao, and H. Zhu, "Optimizing ontology alignment through linkage learning on entity correspondences," *Complexity*, vol. 2021, Article ID 5574732, 12 pages, 2021.

[24] X. Xue, X. Wu, C. Jiang, G. Mao, and H. Zhu, "Integrating sensor ontologies with global and local alignment extractions," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6625184, 10 pages, 2021.

[25] X. Chen, D. Wu, and I. Ahmad, "Optimized viewport-adaptive 360-degree video streaming," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 3, pp. 347–359, 2021.

[26] Y. Sun, I. Ahmad, D. Li, and Y. Q. Zhang, "Region-based rate control and bit allocation for wireless video transmission," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 1–10, 2006.

[27] Y. Sun and I. Ahmad, "A robust and adaptive rate control algorithm for objects-based video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 14, pp. 1167–1182, 2004.

[28] J. Zhao, B. Li, C. W. Kok, and I. Ahmad, "MPEG-4 video transmission over wireless networks: A link level performance study," *Wireless Networks*, vol. 10, no. 2, pp. 133–146, 2004.

[29] I. Ahmad and J. Luo, "On using game theory to optimize the rate control in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 209–219, 2006.

[30] 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, New Yor, NY, USA, June 2017.

[31] J. Yi, M. R. Islam, S. Aggarwal, D. Koutsonikolas, Y. C. Hu, and Z. Yan, "An analysis of delay in live 360° video streaming systems," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 982–990, Seattle, MA, USA, October 2020.

[32] Y. Xing and Z. Jia, "Deep learning-based action recognition with 3D skeleton: a survey," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 80–92, 2021.