A Content-based Viewport Prediction Framework for 360° Video Using Personalized Federated Learning and Fusion Techniques

Mehdi Setayesh and Vincent W.S. Wong
Department of Electrical and Computer Engineering
The University of British Columbia, Vancouver, Canada
{setayeshm, vincentw}@ece.ubc.ca

***Abstract*—Viewport prediction is a key enabler for 360° video streaming over wireless networks. To improve the prediction accuracy, a common approach is to use a content-based viewport prediction model. Saliency detection based on traditional convolutional neural networks (CNNs) suffers from distortion due to equirectangular projection. Also, the viewers may have their own viewing behavior and are not willing to share their historical head movement with others. To address the aforementioned issues, in this paper, we first develop a saliency detection model using a spherical CNN (SPCNN). Then, we train the viewers' head movement prediction model using personalized federated learning (PFL). Finally, we propose a content-based viewport prediction framework by integrating the video saliency map and the head orientation map of each viewer using fusion techniques. The experimental results show that our proposed framework provides higher average accuracy and precision when compared with three state-of-the-art algorithms from the literature.**

***Index Terms*—Personalized federated learning, saliency detection, 360° video, viewport prediction.**

I. Introduction

A 360° video provides immersive viewing experience with the sense of total presence for the viewers. Delivering the entire 360° video to the viewers require much higher bandwidth compared to the conventional two-dimensional (2D) video [1]. However, at any time, each viewer watching a 360° video perceives it only from one direction. The region of the video that the viewer is watching at any given time is called a viewport. The center of the viewport is known as viewpoint. To efficiently utilize the wireless network bandwidth, each video frame is divided into tiles. Then, the tiles covering the viewport are delivered to the viewers with maximum possible quality [2]. Most of the 360° video streaming techniques require the prediction of the viewport of each viewer [3].

Recently, deep neural networks (DNNs) are utilized in viewport prediction models to improve the prediction accuracy. Viewport prediction can be categorized into content-independent and content-based approaches [4]. In content-independent approach, the viewers' historical head movement is used for viewport prediction. Liu *et al.* in [5] proposed long short-term memory (LSTM) and gated recurrent unit (GRU) architectures to predict the viewers' viewports. Chao *et al.* in [6] proposed a transformer-based architecture to predict the future viewports. Since DNNs require a large amount of data for training, each viewer cannot obtain a prediction model with high accuracy only by using its local historical data. Moreover, due to privacy concerns, the viewers may not be willing to share their historical data with others.

Recently, federated learning (FL) has been employed to enable different viewers to train a single learning model collaboratively under the coordination of a central server. The viewers do not need to share their local raw data with others. FedAvg [7], which is a popular FL algorithm, has been used in [8] for viewport prediction. However, viewers may have different viewing patterns and this can lead to data heterogeneity among the viewers' historical data. The data heterogeneity issue can be addressed by using personalized FL (PFL) algorithms. In PFL, a personalized model is obtained for each viewer using techniques such as meta-learning [9] and model decomposition [10]. A PFL algorithm based on meta-learning has been used in [11] for viewport prediction. However, both [8] and [11] use a content-independent viewport prediction approach.

In content-based viewport prediction approach, both the video content and the viewers' historical head movement are used to predict the future viewports. In particular, the content of the video frames is used for saliency detection. The performance of viewport prediction can further be improved by taking into account the historical head movement of the viewers [12]. Nguyen *et al.* in [13] proposed a deep convolutional neural network (CNN), trained by transfer learning, to obtain the saliency map of each 360° video frame. Then, an LSTM architecture is used to predict the viewport. Wang *et al.* in [2] proposed a convolutional LSTM architecture as the learning model for content-based viewport prediction. The proposed algorithms in [2] and [13] are centralized viewport prediction algorithms. In addition, both [2] and [13] use traditional CNNs in their saliency detection model.

To facilitate storing and processing of 360° videos, the videos are usually projected onto the 2D plane. Equirectangular projection (ERP) is a commonly used projection format. However, planar projection of 360° video frames can lead to space-varying distortions [12]. This makes translational weight sharing in traditional CNNs to be ineffective for 360°

This work was supported in part by Rogers Communications Canada Inc., the Institute for Computing, Information and Cognitive Systems (ICICS) at the University of British Columbia, and the Digital Research Alliance of Canada.

videos. Spherical convolution for an ERP projected video frame has been introduced in [14], which can impose both the translational and rotational invariance of the weights in the CNNs. Recently, employing spherical convolution in the U-Net architecture has shown superior performance in saliency detection for 360° videos [12], [14]. However, the work in [14] leverages the saliency map of the previous frame as the model input. Thus, it is prone to error propagation. The algorithm in [12] requires feedback from the viewers about their viewport.

In this work, we propose a content-based viewport prediction framework, which to the best of our knowledge is the first one using PFL. We decouple the viewport prediction model into two parts. The first part is responsible for saliency detection. The second part is responsible for head movement prediction. The contributions of this paper are as follows:

- We develop a DNN architecture based on spherical CNN (SPCNN) to enable efficient saliency detection for 360° video frames. The model is trained by the server without the need for previous saliency maps or any feedback from the viewers to be used as its input.
- To tackle the viewers' privacy concerns and data heterogeneity issue, we propose to train the head movement prediction model using a PFL algorithm.
- We use fusion techniques to enable content-based viewport prediction. In particular, we integrate the outputs of the saliency detection and head movement prediction models using fusion techniques. Then, the viewport is predicted based on the obtained fused feature map.
- The results on a public 360° video dataset [14] show that for 6×8 tiling pattern, our proposed framework can achieve an accuracy which is on average 4.35%, 5.88%, and 7.46% higher than that of the viewport prediction algorithms proposed in [11], [8], and [4], respectively.

This paper is organized as follows. The saliency detection model is described in Section II. Our proposed PFL-based head movement prediction algorithm is presented in Section III. In Section IV, we present our proposed viewport prediction framework. In Section V, we evaluate the performance of the proposed framework. Section VI concludes the paper.

II. SPCNN-based Saliency Detection Model

In this section, we first describe the spherical convolution for 360° video frames that are projected onto a 2D plane. We then present our proposed SPCNN-based architecture for saliency detection.

A. Spherical Convolution

A spherical manifold $\mathcal{S}^2$ is defined as a set of points located on the surface of a unit sphere in $\mathbb{R}^3$. We parameterize $\mathcal{S}^2$ by the latitude and longitude angles θ and ϕ, respectively. The latitude angle θ, where $0 \leq \theta \leq \pi$, is the angle measured from the z-axis. The longitude angle ϕ, where $0 \leq \phi \leq 2\pi$, is the angle measured from the x-axis after projection onto the $x-y$ plane. The vertical and horizontal coordinates of an ERP projected video frame are the latitude and longitude angles on the sphere, respectively. Since the areas near the poles are required to be stretched horizontally, ERP results in greater distortion in the polar regions of the sphere. Each video frame of a 360° video can be represented as a continuous function λ on a spherical manifold $\mathcal{S}^2$. We have $\lambda : \mathcal{S}^2 \rightarrow \mathbb{R}^K$, where K is the number of channels (e.g., $K = 3$ for RGB color system). Let k denote a kernel function on $\mathcal{S}^2$. Similar to [14], we consider the kernel function k as a sphere cap. For an ERP projected video frame, the spherical convolution can be expressed as follows:

$$(\lambda * k)(\theta, \phi) = \int_0^{2\pi} \int_0^{\pi} \lambda(\theta', \phi') k(\theta' - \theta, \phi' - \phi) \sin\theta' \, d\theta' \, d\phi', \quad (1)$$

where $\sin\theta'$ is used to compensate the ERP distortion. Given (1), sampling along the vertical and horizontal coordinates enables discrete spherical convolution on ERP projected video frames. We denote the sets of sampling points along the vertical and horizontal coordinates by Θ and Φ, respectively.

B. Saliency Detection Model

A saliency detection model determines the saliency map for each video frame. Consider a server containing a training dataset that consists of ERP projected 360° video frames and their corresponding saliency maps. Let $\mathcal{M}^{\text{srv}} = \{1, \ldots, M^{\text{srv}}\}$ denote the set of 360° videos in the server's training dataset. Since the duration of each video is different, we denote the set of indices corresponding to frame timestamps of 360° video $m \in \mathcal{M}^{\text{srv}}$ by $\mathcal{T}^m = \{1, \ldots, T^m\}$. For 360° video $m \in \mathcal{M}^{\text{srv}}$, let v_t^m and s_t^m denote the video frame and its corresponding saliency map at timestamp $t \in \mathcal{T}^m$, respectively. We develop a saliency detection model which consists of a contraction path and an expansion path. Different from U-Net, which is based on traditional CNNs, our proposed model uses a SPCNN-based architecture. Moreover, we employ a convolutional block attention module (CBAM) [15] in our proposed saliency detection model to refine the intermediate output features using the channel attention module (CAM) and spatial attention module (SAM). The CAM and SAM extract the inter-channel and inter-spatial relationship of the features, respectively.

As shown in Fig. 1, the saliency detection model takes the current video frame v_t^m as input. The output is the predicted saliency map $\hat{s}_t^m$. The input passes through a contraction path, a spherical convolutional layer, a CBAM, and an expansion path. The contraction path consists of three spherical convolutional layers, each of which is followed by a 2×2 max pooling layer. In the expansion path, there are three spherical convolutional layers, each after an unpooling layer.

Let $\boldsymbol{\omega}^{\text{srv}}$ denote the learning parameters of the saliency detection model. We have $\hat{s}_t^m = g^{\text{srv}}(v_t^m; \boldsymbol{\omega}^{\text{srv}})$, where $g^{\text{srv}}(\cdot)$ denotes the prediction function based on the considered SPCNN-based architecture. The server aims to determine $\boldsymbol{\omega}^{\text{srv}}$ by minimizing the following loss function:

$$L^{\text{srv}}(\boldsymbol{\omega}^{\text{srv}}) = \frac{1}{M^{\text{srv}}} \sum_{m \in \mathcal{M}^{\text{srv}}} \frac{1}{T^m} \sum_{t \in \mathcal{T}^m} \sum_{\theta \in \Theta} \sum_{\phi \in \Phi} \beta(\theta, \phi) \times \left(\hat{s}_t^m(\theta, \phi) - s_t^m(\theta, \phi)\right)^2, \quad (2)$$

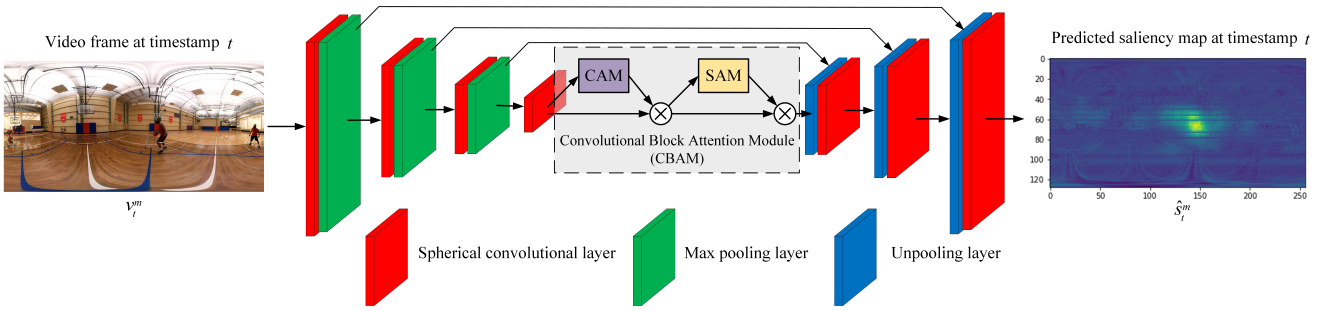

Fig. 1: SPCNN-based architecture for saliency detection.

where $\beta(\theta, \phi)$ is a weighting coefficient and is equal to the solid angle of the sampled area normalized by the solid angle of a unit sphere (i.e., 4π). $L^{\text{srv}}(\boldsymbol{\omega}^{\text{srv}})$ is minimized using stochastic gradient descent (SGD) and a fixed learning rate over training epochs.

III. PFL-based Head Movement Prediction

In this section, we first describe a GRU-based DNN architecture for predicting the viewers' head movements. Then, we propose a PFL algorithm which enables the viewers to train their head movement prediction models without sharing their historical data with others including the server.

A. Head Movement Prediction Model

The viewers can watch 360° videos with head mounted displays (HMDs). An HMD is able to measure the head movement of each viewer. The measured head movement can be represented by the latitude and longitude angles. Let $\mathcal{U} = \{1, \ldots, U\}$ denote the set of viewers. Let $\mathcal{M}^u = \{1, \ldots, M^u\}$ denote the set of videos for which viewer $u \in \mathcal{U}$ has its head movement measurements in its training dataset. Let $\mathcal{T}^{m,u} = \{1, \ldots, T^{m,u}\}$ denote the set of indices corresponding to frame timestamps of 360° video $m \in \mathcal{M}^u$ watched by viewer $u \in \mathcal{U}$. At timestamp $t \in \mathcal{T}^{m,u}$, we denote the head movement of viewer u by $\boldsymbol{o}_t^{m,u} = (\theta_t^{m,u}, \phi_t^{m,u})$. We use a GRU-based DNN architecture for head movement prediction.

As shown in Fig. 2, the head movement prediction model takes a sequence with length Q of the current and previous head movements, i.e., $\mathcal{O}_t^{m,u} = \{\boldsymbol{o}_{t-Q+1}^{m,u}, \ldots, \boldsymbol{o}_t^{m,u}\}$ as input, and returns the predicted head movement at the $(t+P)$-th timestamp, i.e., $\hat{\boldsymbol{o}}_{t+P}^{m,u}$, where P denotes the prediction window. The head movement prediction model consists of two layers. Each layer has Q GRU cells. Let d denote the number of features in the hidden state of each GRU cell. $\boldsymbol{h}_0^1$ and $\boldsymbol{h}_0^2 \in \mathbb{R}^d$ are the initial hidden states of the first and second layers, respectively. $\boldsymbol{h}_0^1$ and $\boldsymbol{h}_0^2$ are initialized to zero for the first input sequence of the head movements corresponding to each video $m \in \mathcal{M}^u$ watched by viewer $u \in \mathcal{U}$. The output of the last GRU cell in the second layer passes through a fully connected (FC) layer. The output of the FC layer is the predicted head movement at the $(t+P)$-th timestamp.

For each viewer $u \in \mathcal{U}$, let $\boldsymbol{\omega}^u$ denote the learning parameters of the head movement prediction model. We have $\hat{\boldsymbol{o}}_{t+P}^{m,u} = g^u(\mathcal{O}_t^{m,u}; \boldsymbol{\omega}^u)$, where $g^u(\cdot)$ denotes the prediction function based on the considered GRU-based DNN architecture. Each viewer u aims to determine $\boldsymbol{\omega}^u$ by minimizing the following loss function based on its local historical head movement:

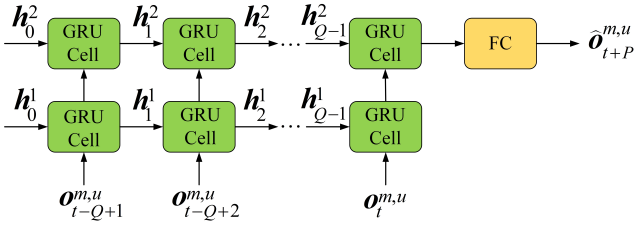

Fig. 2: GRU-based head movement prediction model.

$$L^u(\boldsymbol{\omega}^u) = \frac{1}{M^u} \sum_{m \in \mathcal{M}^u} \frac{1}{T^{m,u}} \sum_{t \in \mathcal{T}^{m,u}} \left\| \hat{\boldsymbol{o}}_{t+P}^{m,u} - \boldsymbol{o}_{t+P}^{m,u} \right\|^2. \quad (3)$$

B. Model Training Using PFL

In this section, we propose a PFL algorithm for training the head movement prediction model. The proposed PFL algorithm can address the data heterogeneity issue and preserve the viewers' privacy. Model decomposition has recently emerged as a promising method for PFL. For model decomposition, we use an approach similar to FedBABU [10], which has been used for image classification tasks. For each viewer $u \in \mathcal{U}$, we decompose the learning model $\boldsymbol{\omega}^u$ into a global learning model $\boldsymbol{\omega}^{\text{GRU}}$ and a local head model $\boldsymbol{\omega}^{\text{FC},u}$. We have $\boldsymbol{\omega}^u = \{\boldsymbol{\omega}^{\text{GRU}}, \boldsymbol{\omega}^{\text{FC},u}\}$. $\boldsymbol{\omega}^{\text{GRU}}$ and $\boldsymbol{\omega}^{\text{FC},u}$ contain the learning parameters of the GRU cells and the FC layer in Fig. 2, respectively. All the local head models $\boldsymbol{\omega}^{\text{FC},u}$, $u \in \mathcal{U}$, are initialized with the same random weights $\boldsymbol{\omega}^{\text{FC}}$ and are kept fixed during training of the global model $\boldsymbol{\omega}^{\text{GRU}}$. After convergence to a global model, each viewer u obtains its personalized prediction model including its personalized local head model $\boldsymbol{\omega}^{\text{FC},u}$ by fine-tuning the learning model $\boldsymbol{\omega}^u$ based on its local historical data.

The global model $\boldsymbol{\omega}^{\text{GRU}}$ is trained through communication rounds. Let $\mathcal{R} = \{1, \ldots, R\}$ denote the set of communication rounds. At the beginning of each communication round $r \in \mathcal{R}$, the viewers download the latest global model $\boldsymbol{\omega}_{r-1}^{\text{GRU}}$ from the server. $\boldsymbol{\omega}_0^{\text{GRU}}$ is initialized randomly by the server. Each viewer $u \in \mathcal{U}$ initializes its global model $\boldsymbol{\omega}_r^{\text{GRU},u}$ by the downloaded global model. We have $\boldsymbol{\omega}_r^{\text{GRU},u} = \boldsymbol{\omega}_{r-1}^{\text{GRU}}$,

Algorithm 1: PFL-based Training Algorithm

Set the number of communication rounds R; the number of local update iterations n; the learning rate η.
Initialize randomly $\boldsymbol{\omega}_0^{\text{GRU}}$ and $\boldsymbol{\omega}^{\text{FC}}$.
Set $\boldsymbol{\omega}^{\text{FC},u} := \boldsymbol{\omega}^{\text{FC}}$, $u \in \mathcal{U}$.
for each communication round $r \in \mathcal{R} := \{1, \ldots, R\}$ **do**
  for each viewer $u \in \mathcal{U}$ in parallel **do**
    Given $\boldsymbol{\omega}_{r-1}^{\text{GRU}}$ and $L^u(\boldsymbol{\omega}^u)$ in (3), perform n local update iterations to obtain the updated $\boldsymbol{\omega}_r^{\text{GRU},u}$.
  end for
  $\boldsymbol{\omega}_r^{\text{GRU}} := \sum_{u \in \mathcal{U}} \frac{N^u}{N} \boldsymbol{\omega}_r^{\text{GRU},u}$.
end for
for each viewer $u \in \mathcal{U}$ in parallel **do**
  Fine-tune the learning model $\boldsymbol{\omega}^u := \{\boldsymbol{\omega}_R^{\text{GRU}}, \boldsymbol{\omega}^{\text{FC},u}\}$ using the local historical data.
end for

$u \in \mathcal{U}$. Then, each viewer performs n local update iterations to update the global model using its local historical data. For each local update iteration, we have $\boldsymbol{\omega}_r^{\text{GRU},u} \leftarrow \boldsymbol{\omega}_r^{\text{GRU},u} - \eta \nabla_{\boldsymbol{\omega}^{\text{GRU}}} L^u(\boldsymbol{\omega}^u)\big|_{\boldsymbol{\omega}^{\text{GRU}} = \boldsymbol{\omega}_r^{\text{GRU},u}}$, where η is the learning rate. After finishing the local update iterations, each viewer uploads its updated global model $\boldsymbol{\omega}_r^{\text{GRU},u}$ to the server. At the end of each communication round r, the server computes the new global model $\boldsymbol{\omega}_r^{\text{GRU}}$ by aggregating the updated global models it has received from the viewers. We have $\boldsymbol{\omega}_r^{\text{GRU}} = \sum_{u \in \mathcal{U}} \frac{N^u}{N} \boldsymbol{\omega}_r^{\text{GRU},u}$, where $N^u = \sum_{m \in \mathcal{M}^u} T^{m,u}$ is the number of data samples for viewer u and $N = \sum_{u \in \mathcal{U}} N^u$. Algorithm 1 summarizes our proposed PFL-based training procedure for the head movement prediction model.

IV. Viewport Prediction Framework

In this section, we present a framework to predict the viewport at the $(t+P)$-th timestamp, when a viewer is watching a 360° video at timestamp t. Let $\mathcal{M}$ denote the set of 360° videos that can be streamed from the server to the viewers. Consider viewer $u \in \mathcal{U}$ is watching video $m \in \mathcal{M}$. At timestamp t, if the server does not have the saliency map for video frame v_{t+P}^m, the pre-trained SPCNN-based saliency detection model is used by the server to determine the saliency map $\hat{s}_{t+P}^m$. The viewer u also uses its personalized pre-trained head movement prediction model to obtain $\hat{\boldsymbol{o}}_{t+P}^{m,u}$ based on its current and previous head movements. The viewer sends the predicted head movement $\hat{\boldsymbol{o}}_{t+P}^{m,u}$ to the server. Given $\hat{\boldsymbol{o}}_{t+P}^{m,u}$ as the viewpoint, the server applies a Gaussian kernel to obtain the corresponding head orientation map $\chi_{t+P}^{m,u}$ for viewer u. Then, by using fusion techniques, the server integrates the predicted saliency and head orientation maps to obtain a fused feature map $f_{t+P}^{m,u}$.

Different fusion techniques have been proposed in the literature to determine a fused feature map from multiple low-level feature maps [12], [16]. In this work, we investigate four fusion techniques, namely, *mean*, *max*, *and*, as well as *regional*. For *mean* fusion, we have $f_{t+P}^{m,u} = \frac{\hat{s}_{t+P}^m + \chi_{t+P}^{m,u}}{2}$. For *max* fusion, we have $f_{t+P}^{m,u} = \max\{\hat{s}_{t+P}^m, \chi_{t+P}^{m,u}\}$. For *and* fusion, we have $f_{t+P}^{m,u} = \hat{s}_{t+P}^m \times \chi_{t+P}^{m,u}$. For *regional* fusion, we first normalize each feature in the maps $\hat{s}_{t+P}^m$ and $\chi_{t+P}^{m,u}$ to be within $[0, 1]$. Then, as shown in Fig. 3, the predicted saliency and head orientation maps are divided into tiles. The maximum feature value is obtained in each tile. Let γ_s and γ_χ denote the average of the maximum feature values over all tiles of $\hat{s}_{t+P}^m$ and $\chi_{t+P}^{m,u}$, respectively. For *regional* fusion, we have $f_{t+P}^{m,u} = (1-\gamma_s)^2 \hat{s}_{t+P}^m + (1-\gamma_\chi)^2 \chi_{t+P}^{m,u}$. After obtaining the fused feature map $f_{t+P}^{m,u}$, considering the field-of-view (FoV) of the HMD, we predict the viewer's viewport based on the adjacent tiles with maximum feature values.

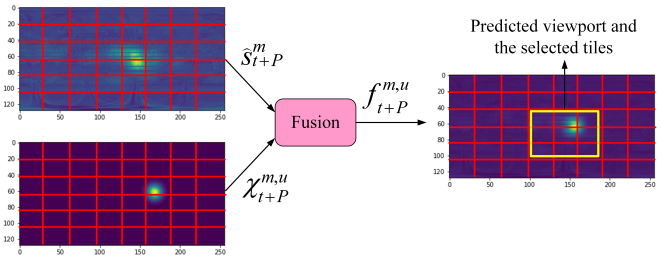

Fig. 3: Obtaining the fused feature map and the selected tiles using the *regional* fusion technique.

V. Performance Evaluation

A. Experimental Setup

Dataset: We conduct our experiments using a public 360° video dataset [14]. The dataset consists of 104 videos including five sports events: basketball, parkour, BMX, skateboarding, and dance. There are 27 viewers. Each video has been watched by at least 18 viewers. For each viewer, the eye gaze points are recorded when watching the videos. We consider 80 and 24 videos in the training and test datasets, respectively.

Parameters Setting: The FoV of the HMD is set to $90° \times 135°$. For training the SPCNN-based saliency detection model, we set the number of training epochs and the learning rate to be 100 and 0.01, respectively. For the head movement prediction model, we set $Q = 10$ and $d = 64$. In Algorithm 1, we set $R = 80$, $n = 2$, and $\eta = 0.01$. Unless stated otherwise, we consider 6×8 tiling pattern and $P = 5$. We perform the experiments in Python 3.6 with PyTorch library.

Benchmarks: We compare the performance of our proposed content-based viewport prediction framework with the following viewport prediction algorithms as benchmarks:

- Combined field-of-view (CFOV) prediction [4]: The viewport is predicted by combining the current viewport and another viewport obtained by spherical walk.
- GRU-based viewport prediction using FedAvg [8]: The viewport is predicted using the head movement prediction model trained by FedAvg.
- Common-personalized federated averaging (ComPer-FedAvg) prediction [11]: The head movement prediction model is trained by a PFL algorithm based on meta-learning.

Performance Metrics: We evaluate the viewport prediction performance using the *accuracy* and *precision* [2] as the performance metrics. Let $\mathcal{M}^{\text{ts},u} = \{1, \ldots, M^{\text{ts},u}\}$ denote

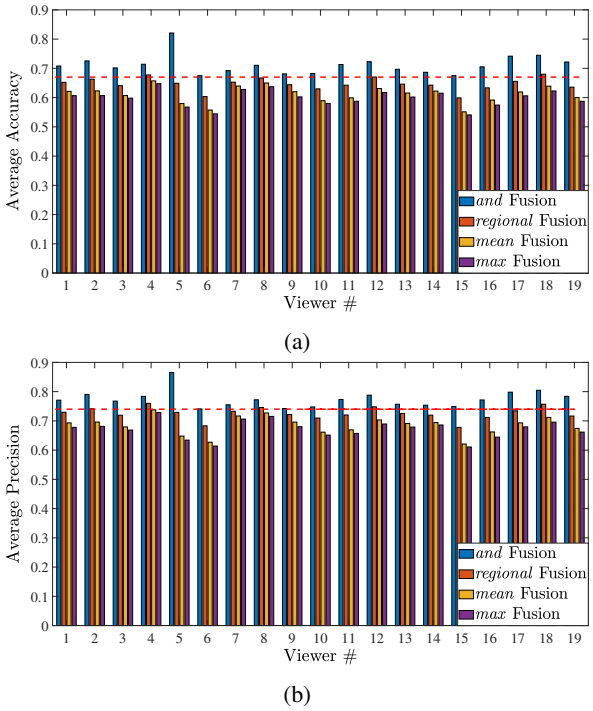

Fig. 4: (a) Average accuracy and (b) average precision of our proposed viewport prediction framework using different fusion techniques.

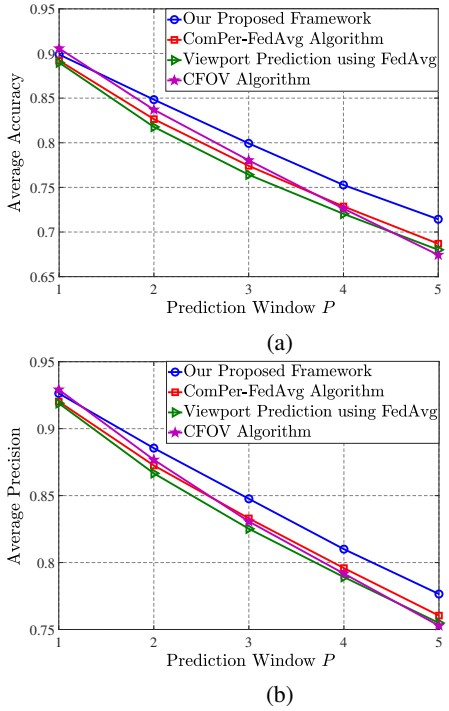

Fig. 5: (a) Average accuracy and (b) average precision versus prediction window P. We use *and* fusion for our proposed framework.

the set of videos watched by viewer $u \in \mathcal{U}$ from the test dataset. Let $\hat{\Upsilon}_t^{m,u}$ and $\Upsilon_t^{m,u}$, respectively, denote the set of tiles selected based on the predicted viewport and the set of tiles covering the ground truth viewport of viewer $u \in \mathcal{U}$ watching video $m \in \mathcal{M}^{\text{ts},u}$ at timestamp $t \in \mathcal{T}^{m,u}$. The accuracy and precision are defined as $\alpha_t^{m,u} = \frac{|\hat{\Upsilon}_t^{m,u} \cap \Upsilon_t^{m,u}|}{|\hat{\Upsilon}_t^{m,u} \cup \Upsilon_t^{m,u}|}$ and $\rho_t^{m,u} = \frac{|\hat{\Upsilon}_t^{m,u} \cap \Upsilon_t^{m,u}|}{|\hat{\Upsilon}_t^{m,u}|}$, respectively, where $|\cdot|$ denotes the cardinality of a set. A higher accuracy implies a lower number of missing tiles in set $\hat{\Upsilon}_t^{m,u}$ and a higher precision means lower number of incorrectly selected tiles in set $\hat{\Upsilon}_t^{m,u}$. For each viewer $u \in \mathcal{U}$, the average accuracy and precision are obtained as $\frac{1}{M^{\text{ts},u}} \sum_{m \in \mathcal{M}^{\text{ts},u}} \frac{1}{T^{m,u}} \sum_{t \in \mathcal{T}^{m,u}} \alpha_t^{m,u}$ and $\frac{1}{M^{\text{ts},u}} \sum_{m \in \mathcal{M}^{\text{ts},u}} \frac{1}{T^{m,u}} \sum_{t \in \mathcal{T}^{m,u}} \rho_t^{m,u}$, respectively.

B. Experimental Results

Effect of Different Fusion Techniques: In Fig. 4, we study the impact of different fusion techniques on the performance of our proposed viewport prediction framework. Fig. 4 shows the average accuracy and precision for the viewers who have watched all the videos in the test dataset. As shown in Fig. 4, *and* fusion can achieve a higher average accuracy and precision compared with other fusion techniques. A region should be salient simultaneously in both the video frame saliency map and the head orientation map to be considered in the fused feature map obtained by *and* fusion. As shown in Fig. 4, for all the viewers, *and* fusion ensures an average accuracy and precision higher than 0.67 and 0.74, respectively.

Effect of Increasing the Prediction Window: A viewer watching a 360° video can prefetch the tiles for the next timestamps. Thus, the delay in delivering the video frames can be reduced by increasing the prediction window P. Fig. 5 shows the average accuracy and precision across the viewers who have watched all the videos in the test dataset versus P. Increasing P leads to the prediction performance degradation for all the algorithms. However, our proposed content-based viewport prediction framework provides a higher average accuracy and precision when compared with other benchmarks.

Effect of Different Tiling Patterns: In Table I, we consider that each video frame can be divided into either 2×3, 4×6, or 6×8 tiles. We compare the average accuracy of our proposed framework with the benchmarks. The results in Table I illustrate that all the algorithms have higher performance for 4×6 tiling pattern. Since the FoV of the HMD is fixed, sets $\hat{\Upsilon}_t^{m,u}$ and $\Upsilon_t^{m,u}$ contain less tiles for 2×3 tiling pattern. Thus, compared to 4×6 tiling pattern, missing a correct tile in set $\hat{\Upsilon}_t^{m,u}$ may have much impact on $\alpha_t^{m,u}$. As shown in Table I, for 2×3 tiling pattern, our proposed framework achieves a higher accuracy by using *regional* fusion. For 6×8 tiling pattern, our proposed framework using *and* fusion achieves the prediction accuracy which is on average 4.35%, 5.88%, and 7.46% higher than that of the proposed algorithms in [11], [8], and [4], respectively. Thus, tiling pattern and fusion techniques are two important factors for 360° video streaming using our proposed viewport prediction framework.

TABLE I: The average accuracy of our proposed framework and the benchmarks for different tiling patterns.

Algorithm	Tiling Pattern		
	2×3	4×6	6×8
Our Proposed Framework using *and* Fusion	0.74	0.77	0.72
Our Proposed Framework using *regional* Fusion	0.75	0.77	0.65
ComPer-FedAvg Algorithm [11]	0.71	0.76	0.69
Viewport Prediction using FedAvg [8]	0.70	0.75	0.68
CFOV Algorithm [4]	0.67	0.72	0.67

VI. Conclusion

In this work, we proposed a framework for content-based viewport prediction. To address the projection distortion problem caused by ERP for 360° videos, we exploited spherical convolution and developed a SPCNN-based architecture for saliency detection. We used a GRU-based DNN architecture for head movement prediction. We addressed the viewers' privacy concerns and data heterogeneity issue by using a PFL algorithm for training the head movement prediction model. Finally, to predict the viewport of each viewer, we integrated the video saliency map and the viewer's head orientation map using fusion techniques. We showed that our proposed framework can achieve a higher average accuracy than the state-of-the-art viewport prediction algorithms proposed in [4], [8], and [11]. One direction for future work is to consider priority weights for the tiles based on their feature values in the fused feature map for efficient 360° video delivery. Also, in practical FL systems, the viewers may have diverse and limited computational capabilities. One can consider employing masking vectors in a PFL algorithm (such as the PerFedMask algorithm proposed in [17]) to address both data and device heterogeneity issues.

References

[1] Y. Huo and H. Kuang, "TS360: A two-stage deep reinforcement learning system for 360-degree video streaming," in *Proc. of Int'l Conf. on Multimedia and Expo (ICME)*, Taipei, Taiwan, Jul. 2022.

[2] M. Wang, S. Peng, X. Chen, Y. Zhao, M. Xu, and C. Xu, "CoLive: An edge-assisted online learning framework for viewport prediction in 360° live streaming," in *Proc. of Int'l Conf. on Multimedia and Expo (ICME)*, Taipei, Taiwan, Jul. 2022.

[3] F. Qian, B. Han, Q. Xiao, and V. Gopalakrishnan, "Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices," in *Proc. of ACM Int'l Conf. Mobile Comput. Netw. (MobiCom)*, New Delhi, India, Oct. 2018.

[4] A. Yaqoob and G.-M. Muntean, "A combined field-of-view prediction-assisted viewport adaptive delivery scheme for 360° videos," *IEEE Trans. Broadcast.*, vol. 67, no. 3, pp. 746–760, Sept. 2021.

[5] X. Liu, X. Li, and Y. Deng, "Learning-based prediction and proactive uplink retransmission for wireless virtual reality network," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10 723–10 734, Oct. 2021.

[6] F.-Y. Chao, C. Ozcinar, and A. Smolic, "Transformer-based long-term viewport prediction in 360° video: Scanpath is all you need." in *Proc. of Int'l Workshop Multimedia Signal Process. (MMSP)*, Tampere, Finland, Oct. 2021.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of Int'l Conf. Artificial Intelligence and Statistics (AISTATS)*, Ft. Lauderdale, FL, Apr. 2017.

[8] X. Liu, Y. Deng, C. Han, and M. Di Renzo, "Learning-based prediction, rendering and transmission for interactive virtual reality in RIS-assisted terahertz networks," *IEEE J. Sel. Areas in Commun.*, vol. 40, no. 2, pp. 710–724, Feb. 2022.

[9] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach." in *Proc. of Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, Canada, Dec. 2020.

[10] J. Oh, S. Kim, and S.-Y. Yun, "FedBABU: Towards enhanced representation for federated image classification," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Apr. 2022.

[11] R. Zhang, J. Liu, F. Liu, T. Huang, Q. Tang, S. Wang, and F. R. Yu, "Buffer-aware virtual reality video streaming with personalized and private viewport prediction," *IEEE J. Sel. Areas in Commun.*, vol. 40, no. 2, pp. 694–709, Feb. 2022.

[12] J. Li, L. Han, C. Zhang, Q. Li, and Z. Liu, "Spherical convolution empowered viewport prediction in 360 video multicast with limited FoV feedback," *ACM Trans. on Multimedia Comput. Commun. Appl.*, vol. 19, no. 1, pp. 1–23, Jan. 2023.

[13] 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 ACM Int'l Conf. on Multimedia*, Seoul, Republic of Korea, Oct. 2018.

[14] Z. Zhang, Y. Xu, J. Yu, and S. Gao, "Saliency detection in 360° videos," in *Proc. of Eur. Conf. on Comput. Vis. (ECCV)*, Munich, Germany, Sept. 2018.

[15] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. of Eur. Conf. on Comput. Vis. (ECCV)*, Munich, Germany, Sept. 2018.

[16] K. Zhang and Z. Chen, "Video saliency prediction based on spatial-temporal two-stream network," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 12, pp. 3544–3557, Dec. 2019.

[17] M. Setayesh, X. Li, and V. W.S. Wong, "PerFedMask: Personalized federated learning with optimized masking vectors," in *Proc. of Int'l Conf. on Learning Representations (ICLR)*, Kigali, Rwanda, May 2023.