Available Online: 10 June 2025 DOI: 10.54254/2977-3903/2025.24025

User clustering-based GAN-LSTM model for viewport prediction in 360-degree video streaming

Yi Liang

Guilin University of Electronic Technology, Guilin, China

liangyiguet@163.com

Abstract. Accurate viewport prediction is crucial for enhancing user experience in 360-degree video streaming. However, due to significant behavioral differences among user groups, traditional single LSTM models tend to fall into local optima and fail to achieve precise predictions. To address this, this paper proposes a hybrid prediction model based on user clustering. First, a Density-Based Clustering Algorithm (DBSCAN) is used to group users with similar behavioral patterns. Then, a hybrid prediction model combining Generative Adversarial Networks (GANs) and Long Short-Term Memory networks (LSTMs) is designed to effectively mitigate data imbalance and overfitting through collaborative training. Experiments conducted on three real-world datasets from YouTube demonstrate that this approach significantly outperforms existing methods based on user trajectories or video saliency in terms of prediction accuracy and stability.

Keywords: 360-degree video streaming, viewport prediction, LSTM

1. Introduction

360-degree videos present content in a panoramic format, offering viewers an immersive watching experience [1]. To reduce the bandwidth required for streaming such videos, predicting users' future viewports becomes essential [2]. The core challenge in viewport prediction lies in identifying users' attention distribution and behavioral patterns. Several commonly used viewport prediction approaches have emerged.

Recent studies have focused on extracting and integrating features from both visual saliency and user trajectories, considering both factors simultaneously [3-5]. Visual saliency incorporates low-level visual features (e.g., color, brightness, contrast), high-level semantic features (e.g., object recognition, contextual relationships), as well as motion and depth information. These combined features allow saliency detection models to capture characteristics aligned with human visual perception [6]. Trajectory-based methods analyze users' gaze patterns while watching similar videos to predict gaze movement directions in new videos [7]. Some research has shown that users' visual attention is often influenced by their past behavior [8-12]. Historical data helps build more accurate attention models—for instance, repetitive behavior modeling reveals that users may repeatedly focus on specific features. Although combining these two factors leads to a more comprehensive model, it also increases the amount of training data required. Models trained on data from the entire user population may fail to predict viewports for certain users accurately, especially when their viewing patterns differ from the majority. This highlights room for improvement in the current models.

Some emerging models have attempted to segment users into groups and train separate models for different user types [13-15]. These models incorporate user-specific factors into feature vectors to improve alignment with individual users. However, current group-based prediction methods often divide users uniformly without truly analyzing user characteristics. This merely increases the number of sub-models without enabling data support across similar users, limiting the accuracy of predictions for niche user groups. Additionally, data volume inevitably decreases for each group after classification, potentially leading to limited feature learning, underfitting, and unstable model training. To address these issues, data augmentation techniques must be employed to compensate for the reduced sample size in each group and ensure sufficient data for neural network training.

This paper analyzes and clusters multi-user data, proposing a clustering-based viewport prediction method. Our approach comprises three components: saliency extraction, user clustering, and cluster-based viewport prediction. First, a pyramid attention mechanism in a convolutional neural network is used to analyze image features in video frames and generate saliency maps, which are then transformed into feature values and incorporated into the model's training. Next, a Density-Based Clustering Algorithm (DBSCAN) groups similar user trajectories, enabling group-wise analysis and training. To address data scarcity post-clustering,

we apply Generative Adversarial Networks (GANs) with high-quality generative capabilities to augment the data, enhancing the model's understanding of data distribution and improving generalization and robustness. Finally, for each data group obtained, a cluster-based prediction model is built. Based on the number of clusters nnn, nnn Long Short-Term Memory (LSTM) networks are trained on each user group's data. The resulting models are integrated using a weighted fusion approach. A loss function provides feedback to adjust the weight coefficients. When the target user changes, the loss function changes accordingly, dynamically influencing the weights and achieving adaptive viewport prediction. Compared with other prediction methods, our main contributions are as follows:

We design a clustering network that groups users with similar trajectories. To enhance the richness of data within each cluster and ensure the reliability of LSTM training, GANs are used to augment the data, thereby improving the model's generalization.

We develop a clustering-based viewport prediction framework that fully considers user-related information. Starting from user groups, the model jointly trains video content features and user tracking data to improve prediction accuracy. Multiple LSTM-based models are integrated using a weight-based fusion mechanism, where sub-models with lower loss values are assigned higher weights to enable dynamic adjustment.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 details the system design. Section 4 presents performance evaluation. Section 5 concludes the paper.

2. Related work

Viewport prediction for 360-degree video streaming primarily focuses on analyzing users' attention distribution and behavioral patterns. Chen et al. proposed a viewport-aware real-time streaming framework for 360-degree videos to optimize end-to-end video delivery, thereby enhancing user experience [16]. However, their method requires significant computational resources. Nguyen et al. developed a head motion prediction algorithm to accurately forecast a user's future head orientation [17]. This method captures users' specific attention for augmenting critical visual features, but it falls short in supporting long-term prediction, which is essential for effective video buffering. Xu et al. performed multi-feature extraction on images and applied Graph Neural Networks (GNNs) for feature fusion [18]. While this approach improves feature processing and utilization, it also demands considerable resources for feature integration. Wang introduced a saliency model for video that incorporates attention mechanisms within a CNN-LSTM architecture to enable fast end-to-end saliency learning [19]. Although CNN-LSTM is one of the most advanced methods for saliency prediction in 2D videos, its effectiveness diminishes when applied to 360-degree video content. Xu et al. applied deep reinforcement learning to viewport prediction, using users' gaze trajectories as input, but this method lacks comprehensive integration of visual saliency [20]. Chopra et al. proposed a fast and efficient online Field-of-View (FoV) prediction model, PARIMA, which uses past FoV data and key object trajectories as proxies for video content [21]. Li et al. employed spherical convolution to reduce projection distortion in 360-degree video, effectively minimizing distortions [22]. However, the complexity of spherical convolution makes it less practical than projection-based methods. Li et al. designed a long-term prediction model that leverages a user's past FoV center trajectory along with other users' future FoV positions to forecast the target user's future viewport [23]. While this approach supports long-term prediction, it lacks the precision required for accurate viewport estimation. Dong et al. utilized historical gaze trajectories from other users to predict the target user's viewport, thereby reducing computational cost [14]. Although this model considers gaze location correlations among users, it does not implement systematic user clustering. Tu et al. focused on industrial control networks, designing a video prediction, tiling, and bitrate allocation scheme specific to such environments [5]. However, its application scenarios are limited, and it does not effectively leverage historical gaze information from other users. Chen proposed a viewport prediction approach centered on individual users, using their own trajectories and the historical behaviors of similar users [13].

In general, existing 360-degree viewport prediction methods for the overall user base leverage two main types of information: (1) features derived from video content itself, such as saliency and spatial characteristics; and (2) historical user behavior data used to train time-series models. However, these models often overlook individual differences between users and between videos. Using a single unified model fails to meet the needs of users with distinctive behavioral patterns.

3. System overview

In this section, we provide an overview of the 360-degree video streaming system based on viewport prediction. As illustrated in Figure 1, the client initiates video requests through a request generator to the server. Once video playback begins, the client records the user's viewport trajectory data, typically represented as a sequence of viewport centers over time.

To enable viewport prediction on the client side, the server receives the gaze trajectory from the client and combines it with stored offline video content and historical user data. This information is then passed into the viewport prediction module. After feature extraction, the data enters a clustering module to perform user clustering. Subsequently, it flows into a neural network combining Generative Adversarial Networks (GANs) and Long Short-Term Memory (LSTM) networks to generate the final prediction. Based on available bandwidth, the server adjusts the image quality accordingly. Upon receiving a request response, the server transmits the texture tiles to the client, which decodes and reconstructs the user's viewport region.

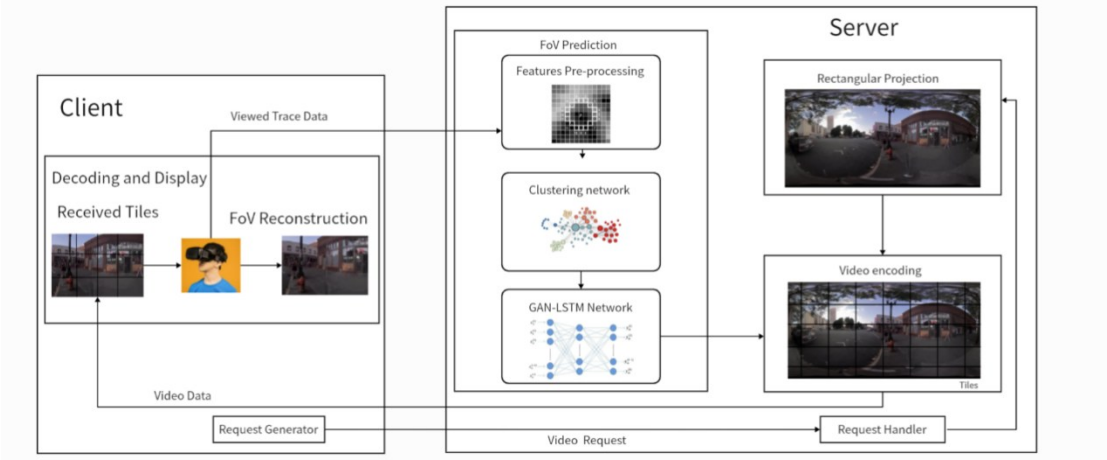

Figure 1. 360-degree video streaming system model

3.1. User clustering

Our observations indicate that users tend to produce highly similar viewing trajectories when watching the same video, making user clustering based on trajectory data both feasible and effective. The user clustering problem can be formulated as follows: for each video in the dataset, participants generate a time-ordered trajectory set. A trajectory for user i is defined as $T_i = \{P_1, P_2, \dots, P_N\}$, where N is the number of frames. Now consider two trajectories T_i and T_j corresponding to users i and j (belonging to different users). The distance between their viewport centers at frame $t(t \in N)$ (defined as the distance between their trajectory points at that frame) is given by: $d(i,j) = |P_{i,t} - P_{j,t}|$. Similarly, the overall distance between two trajectories can be represented using the Euclidean distance. This leads to the construction of a two-dimensional symmetric distance matrix, where each element d_{ij} represents the distance between trajectory i and trajectory j. For a dataset containing N user trajectories, the resulting distance matrix has dimensions $N \times N$. The Euclidean distance between any two trajectories T_i and T_j is computed using the following formula: $d(T_i, T_j) = \sqrt{\sum_{k=1}^{m}(T_{i,k} - T_{j,k})^2}$, where $T_{i,k}$ and $T_{j,k}$ represent the values of trajectories T_i and T_j in the $k\text{-}th$ dimension, and m denotes the dimensionality of the trajectory vectors. By calculating the pairwise Euclidean distances between all trajectories, we construct a symmetric distance matrix D, where each element D represents the distance between trajectory T_i and T_j. To build this matrix, we first initialize a zero matrix of size $N \times N$, then iterate through all trajectory pairs and compute their Euclidean distances $D[i,j] = d(T_i, T_j)$. Since the distance matrix is symmetric, we ensure that: $D[i,j] = D[j,i]$.

Pseudocode 1: User Clustering Using DBSCAN

```
Input: Dataset D (with initialized points), eps, MinPts
Output: Clusters and noise points

// Main clustering process
for each unvisited point P in D:
    if P.visited == FALSE:
        P.visited = TRUE
        NeighborPts = regionQuery(P, eps)

        if sizeof(NeighborPts) < MinPts:
            P.clusterId = NOISE
        else:
            ClusterId = ClusterId + 1
            expandCluster(P, NeighborPts, ClusterId)

Function expandCluster(P, NeighborPts, ClusterId):
    P.clusterId = ClusterId
```

```
    for each point P' in NeighborPts:
        if P'.visited == FALSE:
            P'.visited = TRUE
            NeighborPts' = regionQuery(P', eps)

            if sizeof(NeighborPts') >= MinPts:
                NeighborPts = NeighborPts ∪ NeighborPts'

        if P'.clusterId == UNCLASSIFIED or NOISE:
            P'.clusterId = ClusterId

Function regionQuery(P, eps):
    return {P' ∈ D | distance(P, P') ⩽ eps}
```

After obtaining the distance matrix, we apply the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm to perform user clustering using the precomputed distance matrix. The process is illustrated in Pseudocode 1, and proceeds as follows: Starting from a point in the dataset, we examine its neighborhood. If the number of points within this neighborhood is greater than or equal to a predefined threshold minPts, the point is designated as a core point. The core point, along with all other points in its neighborhood, forms a cluster. We then examine the neighborhoods of these neighboring points. If any of them are also core points, the cluster is further expanded. This process is repeated until all reachable points have been processed. If a point is neither a core point nor within the neighborhood of any core point (i.e., not a border point), it is labeled as noise. To ensure the clustering is reasonable, we examine the proportion of noise points after clustering—typically expected to be between 1% and 5%. A key advantage of DBSCAN is that it does not require the number of clusters to be specified in advance. It can automatically detect the number of clusters and is capable of identifying clusters of arbitrary shape. Compared with other clustering methods, DBSCAN's standout advantage is its robustness to noise. It effectively excludes noise points without affecting the quality of the clustering results.

3.2. GAN-LSTM combined neural network

Combined Neural Network Architecture. Figure 2 illustrates the overall architecture of the GAN-LSTM method. The model takes two types of input: Feature vectors extracted from video saliency, typically obtained using the YOLO algorithm; we apply a transfer learning approach to extract these features. User trajectory data $\{P_n\}$, which is first clustered as described in the previous section. Each user group's trajectory data $\{P_n\}$ is fed into a separate GAN, which augments the data before passing it into an LSTM network for model training. The outputs from these LSTM models are then fused in a feature fusion network that integrates the content-based feature vectors and trajectory predictions.

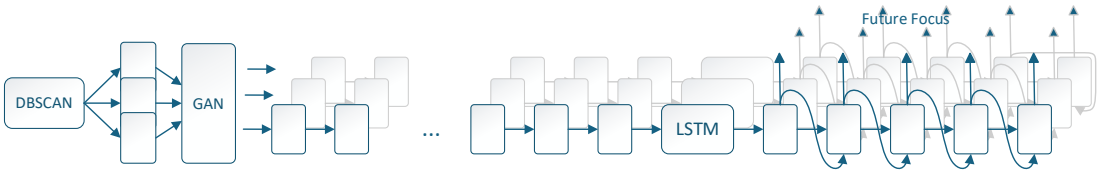

Figure 2. Overall architecture of the GAN-LSTM framework example with three clusters: DBSCAN clusters users into three groups, GANs augment the data, and three LSTM sub-networks predict future viewports

GAN Architecture. Following user clustering, each group contains fewer trajectory samples. In real-world scenarios, limited user data can hinder model training and reduce viewport prediction accuracy. The core components of a GAN are the Generator and the Discriminator. Through adversarial training, the generator learns to produce realistic samples, while the discriminator learns to distinguish real from fake data. As shown in Figure 3, we set the generator's input as a noise vector of length 10. The hidden layer has 128 neurons and uses the ReLU activation function. The output dimension is also 10, matching the data format. The discriminator input layer also has 10 dimensions to accept either real or generated samples. Its hidden layer contains 128 neurons and uses the LeakyReLU activation function to prevent dead neurons. The output layer uses the Sigmoid function to perform binary classification, indicating whether the input data is real. After training both the generator and discriminator, the GAN can generate synthetic samples that resemble the original dataset, thereby augmenting training data and improving model generalization—especially valuable when data is scarce. We apply this GAN-based augmentation to all user clusters to balance the sample sizes across groups and enhance model robustness.

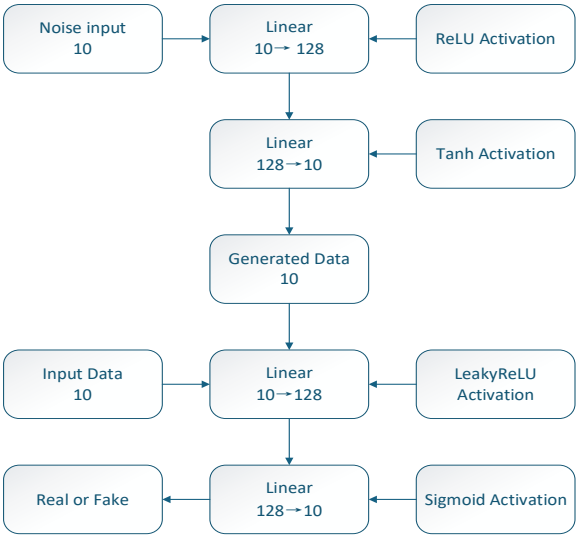

Figure 3. GAN network structure numbers indicate network dimensionality

LSTM Architecture. In the GAN-LSTM framework, the LSTM component is used to model and train user trajectory data—treated as a time series—for predicting the user's future gaze position. The trajectory data is represented as P_t, where the input format consists of the user's 2D coordinates. The first dimension corresponds to the time step, and the second dimension represents spatial coordinates. The spatial coordinates are extracted from the user's yaw and pitch angles, while the roll angle is omitted due to its negligible variation in the data. Simultaneously, video saliency features are input to enable feature fusion. The structure of this part of the network is shown in Figure 4. Here, P_t denotes the trajectory data, which is encoded by the LSTM to capture historical positional information. The saliency feature S_t is flattened via a feature flattening layer (Flatten) and passed through the LSTM network. Then, the outputs from the trajectory and saliency streams are concatenated in a feature fusion layer (Concatenate) to integrate positional and saliency information.

Subsequently, the model computes the predicted position increment to estimate the next-step trajectory P_{t+1}. This process is repeated cyclically to generate predictions of the user's future viewports over time.

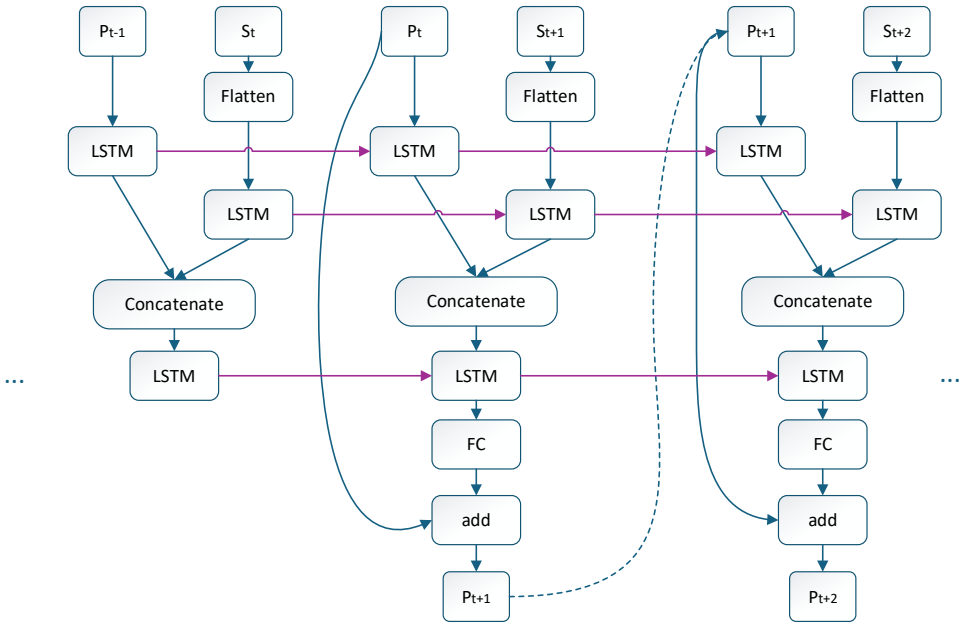

Figure 4. LSTM network unit

The specific design of the LSTM unit is as follows. The forget gate is calculated using Equation (1), where f_t denotes the gate output, and σ is the sigmoid activation function used to constrain the output within the range. Here, h_{t-1} represents the hidden state from the previous time step, x_t is the current input, and W_f, b_f denote the weight matrix and bias for the forget gate. The

input gate, which determines how much of the current input affects the LSTM cell state, is defined by Equations (2) and (3). The candidate cell state is denoted by $\tilde{C}_t$, and the gate output is denoted by i_t. W_i, W_C, b_i, b_C are the corresponding weights and biases for updating the LSTM unit C_t. After that, the cell state is updated as shown in Equation (4). Finally, the output gate generates the current cell output o_t, as given in Equation (5), and the hidden state h_t is updated as shown in Equation (6).

$$f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right) \tag{1}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{3}$$

$$C_t = f_t.C_{t-1} + i_t.\tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t.tanh(C_t) \tag{6}$$

3.3. Dynamic feature fusion network

For the three user groups obtained from clustering, we construct three independent LSTM-based viewport prediction models. Each LSTM model is trained on trajectory data that has been clustered and augmented in the previous steps. When the trajectory data of a target (or current) user is received, each LSTM model generates a prediction of the future viewport. These predictions are then integrated using a set of weights α, as defined in Equation (7):

$$O_t = \sum_1^n \alpha_n o_n \tag{7}$$

$$D_n = \left|o_{n,t} - P_t\right| \tag{8}$$

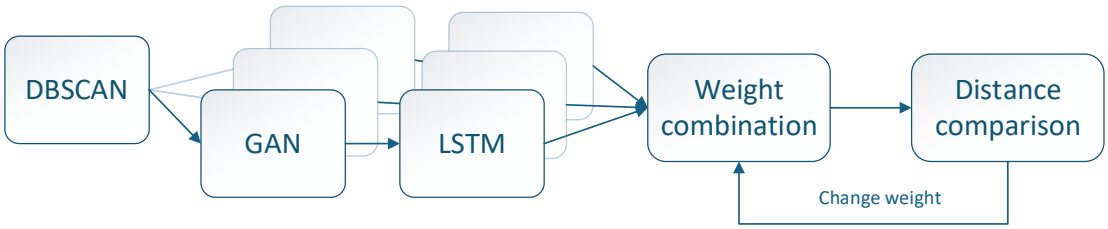

Figure 5. Dynamic feedback structure

To enhance the performance of the viewport prediction model, we employ a feedback-based strategy during the prediction integration stage. Figure 5 illustrates the structure of this dynamic feedback mechanism, which operates as follows: Initially, in the integration phase, the predicted future viewport is obtained by averaging the outputs from multiple LSTM models. The system then evaluates the loss value of the prediction. If the loss is too high, it computes the distance between each LSTM model's predicted point and the ground truth point, denoted as $D_{1,2,3,\ldots n}$, as defined in Equation (8). Here, $o_{n,t}$ represents the predicted result from each LSTM model, and P_t is the true position. The LSTM model with the minimum distance D_n is identified as the best-performing model, and its integration weight α_n is set to 50%. The remaining 50% is distributed evenly among the other models. Notably, the best-performing model's weight is intentionally capped at 50% to avoid underutilization of other models, which could reduce diversity and generalization capacity.

This cluster-based viewport prediction model, constructed through the above components, not only delivers high prediction accuracy but also exhibits strong adaptability to sudden or unexpected user behavior. In the integration stage, dynamic adjustment of model weights via feedback allows the system to flexibly adapt to changes. As a result, the model's responsiveness and robustness are significantly improved.

4. Experimental results and analysis

4.1. Experimental setup

In this section, we design a series of experiments to validate the effectiveness of the proposed GAN-LSTM method for 360-degree viewport prediction. The dataset used (dataset source and size to be filled in) consists of videos split into training and testing sets. The dataset includes trajectory data from 20 participants under 20 different viewing modes, yielding a total of 400 gaze-tracking samples. We use a 4:1 ratio to divide the data into training and validation sets. The GAN-LSTM model is implemented in

TensorFlow and trained using CUDA acceleration for multi-task LSTM processing. Table 1 summarizes the training parameters for both GAN and LSTM modules.

Table 1. GAN-LSTM model training parameters

GAN-net	Hidden layer size	128
	Batch processing size	5
	Optimizer	Adam
	Training rounds	10,000
	Learning rate	0.0002
LSTM-net	Prediction window	20
	Optimizer	Adam
	Number of neurons	1,024
	Learning rate	0.0005

During GAN training, we set the hidden layer size to 128, the batch size to 5, the number of training epochs to 10,000, and the learning rate to 0.0002. Both the generator and discriminator are trained using the Adam optimizer. The training process involves alternating updates to the generator and discriminator losses. The discriminator loss measures how well it distinguishes between real and generated data, while the generator loss measures the generator's ability to deceive the discriminator.

As shown in Table 2, after 5,000 epochs, the discriminator loss decreases to near zero, indicating that the GAN effectively generates samples similar to the real data.

Table 2. GAN training results

Epoch	1,200	1,500	1,800	2,100	2,400	3,000	5,000
d_loss	0.050	0.040	0.030	0.010	0.008	0.002	0.001
Epoch	2,500	5,000	7,500	10,000	12,500	15,000	17,500
g_loss	6.0	9.0	12.5	15.0	17.5	20.0	21.5

4.2. Evaluation

Comparison Methods: We compare our proposed GAN-LSTM method with five state-of-the-art viewport prediction methods: DPH [20], Fixation [24], MFF [18], Graph Learning [3], and Sparkle [13]. We follow the default settings of each baseline and fine-tune them using our training dataset.

Viewport prediction for 360° video aims to forecast the user's future field of view, which cannot be fully captured within a single viewport. Consequently, some existing methods—especially those focused on egocentric gaze prediction for a single subject—are not directly applicable to Viewport Prediction (VS). Our task involves predicting viewports for multiple users with diverse characteristics. To highlight the advantages of our method in handling user diversity, we do not compare it with methods designed for egocentric (single-user) attention prediction.

Evaluation metrics: We employ three widely used metrics to evaluate the performance of each method: Accuracy: The ratio of correctly predicted tiles to the union of predicted and actual viewed tiles. F-score: A harmonic mean of precision and recall, commonly used in classification tasks. Mean Squared Error (MSE): A standard regression metric that measures the average of the squared differences between predicted and actual values. The length of each scanpath is set to the average scanpath length in the training dataset. For evaluation, each predicted scanpath is compared against all actual scanpaths from the participants to compute the metrics. Higher accuracy and F-score values indicate better prediction quality, while a lower MSE reflects better regression performance.

Table 3 presents the prediction results of our method compared with other methods on two datasets: [sport] and [diving]. Our method significantly outperforms all five baseline methods across all three metrics on both datasets. These results demonstrate the effectiveness and generalization capability of our GAN-LSTM approach for Viewport Prediction (VS).

Table 3. Accuracy, F-score, and MSE on two datasets

Approaches	Salient360 dataset			HTRO dataset		
	Accuracy ↑	F-score ↑	MSE ↓	Accuracy ↑	F-score ↑	MSE ↓
DHP	79.21%	0.51	0.23	72.56%	0.47	0.24
Fixation	84.22%	0.53	0.19	83.15%	0.51	0.20
MFF	81.82%	0.63	0.17	82.13%	0.64	0.17
Graph Learning	82.32%	0.87	0.12	83.17%	0.86	0.13
Sparkle	88.66%	0.88	0.10	89.27%	0.87	0.11
OURS	92.26%	0.89	0.09	92.12%	0.88	0.09

Next, we visualize the predicted trajectories of several methods alongside the actual trajectories. As shown in Figure 6, our method produces predicted trajectories that align more closely with the ground truth compared to other methods. In summary, both the qualitative and quantitative results demonstrate that our method is highly effective for Viewport Prediction (VS) and performs significantly better than existing approaches.

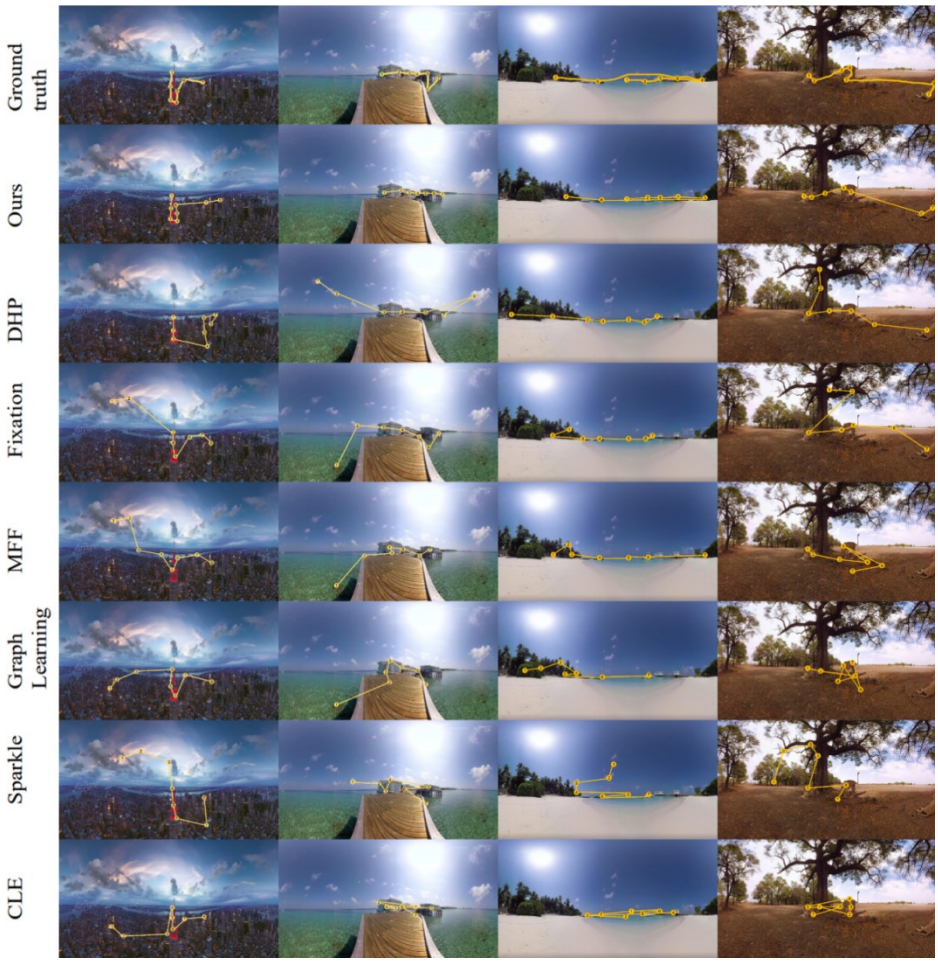

Figure 6. Qualitative results – trajectory prediction comparison

5. Conclusion

In this study, we propose an effective method for predicting future viewports in 360-degree videos. Our review of existing literature reveals that some users exhibit distinct behavioral patterns, making global models insufficient for accurately predicting their viewports. Therefore, successful viewport prediction must include both the analysis of user behavior and the integration of historical gaze trajectories and video saliency through a robust predictive model. This leads us to formulate viewport prediction as a multi-task learning problem. Experimental results on two datasets confirm the advanced performance of our proposed GAN-LSTM model in 360-degree video viewport prediction.

Predicting future viewport content is crucial for delivering a high-quality immersive experience in panoramic video environments. This paper focuses on multi-factor-based viewport prediction, taking into account variations in user viewing behaviors. Predicting viewer gaze trajectories from a user-group-based perspective presents a promising research direction. Such prediction can be enhanced through distributed learning frameworks, offering greater accuracy in viewport prediction.

The proposed method also shows strong potential for future applications. For instance, the rapid development of live-streamed and interactive media will further increase the difficulty of model training. By modeling user behavioral patterns, our approach can help improve the efficiency and accuracy of viewport prediction in panoramic video services.

References

[1] Bastug, E., Bennis, M., Medard, M., & Debbah, M. (2017). Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Communications Magazine*, *55*(6), 110–117. https://doi.org/10.1109/MCOM.2017.1601089

[2] Mao, Y., Sun, L., Liu, Y., & Wang, Y. (2020). Low-latency FoV-adaptive coding and streaming for interactive 360° video streaming. *Proceedings of the 28th ACM International Conference on Multimedia* (pp. 3696–3704). https://doi.org/10.1145/3394171.3413751

[3] Zhang, X., Cheung, G., Zhao, Y., Le Callet, P., Lin, C., & Tan, J. Z. G. (2021). Graph learning based head movement prediction for interactive 360 video streaming. *IEEE Transactions on Image Processing*, *30*, 4622–4636. https://doi.org/10.1109/TIP.2021.3073283

[4] Wang, S., Yang, S., Su, H., Zhao, C., Xu, C., Qian, F., Wang, N., & Xu, Z. (2024). Robust saliency-driven quality adaptation for mobile 360-degree video streaming. *IEEE Transactions on Mobile Computing*, *23*(2), 1312–1329. https://doi.org/10.1109/TMC.2023.3294698

[5] Tu, J., Chen, C., Yang, Z., Li, M., Xu, Q., & Guan, X. (2023). PSTile: Perception-sensitivity-based 360° tiled video streaming for industrial surveillance. *IEEE Transactions on Industrial Informatics*, *19*(9), 9777–9789. https://doi.org/10.1109/TII.2022.3216812

[6] Jiang, Z., Ji, B., Wu, F., Liu, Y., & Zhang, Y. (2020). Reinforcement learning based rate adaptation for 360-degree video streaming. *IEEE Transactions on Broadcasting*, *67*(2), 409–423. https://doi.org/10.1109/TBC.2020.3034157

[7] Zhang, J., Qin, Q., Wan, T., & Luo, X. (2022). Perception-based pseudo-motion response for 360-degree video streaming. *IEEE Signal Processing Letters*, *29*, 1973–1977. https://doi.org/10.1109/LSP.2022.3200882

[8] Park, S., Lee, J., & Choi, J. (2021). Mosaic: Advancing user quality of experience in 360-degree video streaming with machine learning. *IEEE Transactions on Network and Service Management*, *18*(1), 1000–1015. https://doi.org/10.1109/TNSM.2020.3047821

[9] Hou, X., Wang, S., Zhou, Z., Wang, Y., & Wu, D. (2020). Predictive adaptive streaming to enable mobile 360-degree and VR experiences. *IEEE Transactions on Multimedia*, *23*, 716–731. https://doi.org/10.1109/TMM.2020.2990446

[10] Zou, J., Hao, T., Yu, C., & Sun, H. (2019). Probabilistic tile visibility-based server-side rate adaptation for adaptive 360-degree video streaming. *IEEE Journal of Selected Topics in Signal Processing*, *14*(1), 161–176. https://doi.org/10.1109/JSTSP.2019.2951538

[11] Yuan, H., Chen, Y., Yu, M., & Zhu, W. (2019). Spatial and temporal consistency-aware dynamic adaptive streaming for 360-degree videos. *IEEE Journal of Selected Topics in Signal Processing*, *14*(1), 177–193. https://doi.org/10.1109/JSTSP.2019.2952001

[12] Hassan, S. M. H. U., Lee, Y., & Kim, M. (2023). User profile-based viewport prediction using federated learning in real-time 360-degree video streaming. *2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting* (pp. 1–7). https://doi.org/10.1109/BMSB58369.2023.10211189

[13] Chen, J., Sun, Y., He, D., & Wu, E. (2021). Sparkle: User-aware viewport prediction in 360-degree video streaming. *IEEE Transactions on Multimedia*, *23*, 3853–3866. https://doi.org/10.1109/TMM.2020.3030440

[14] Dong, P., Zhang, S., Zhang, H., & Xu, Y. (2023). Predicting long-term field of view in 360-degree video streaming. *IEEE Network*, *37*(1), 26–33. https://doi.org/10.1109/MNET.123.2200371

[15] Jin, Y., Zhang, Z., Wang, W., & Li, S. (2023). Ebublio: Edge-assisted multiuser 360° video streaming. *IEEE Internet of Things Journal*, *10*(17), 15408–15419. https://doi.org/10.1109/JIOT.2023.3283859

[16] Chen, J., Sun, Y., He, D., & Wu, E. (2023). Live360: Viewport-aware transmission optimization in live 360-degree video streaming. *IEEE Transactions on Broadcasting*, *69*(1), 85–96. https://doi.org/10.1109/TBC.2022.3220616

[17] Nguyen, A., & Yan, Z. (2023). Enhancing 360 video streaming through salient content in head-mounted displays. *Sensors*, *23*(5), Article 2470. https://doi.org/10.3390/s23052470

[18] Xu, X., Chen, L., & Liu, Y. (2023). Multi-features fusion based viewport prediction with GNN for 360-degree video streaming. *2023 IEEE International Conference on Metaverse Computing, Networking and Applications* (pp. 57–64). https://doi.org/10.1109/MetaCom57706.2023.00020

[19] Wang, W., Shen, J., Guo, F., Cheng, M., & Borji, A. (2018). Revisiting video saliency: A large-scale benchmark and a new model. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4894–4903). https://doi.org/10.1109/CVPR.2018.00514

[20] Xu, M., Song, Y., Wang, J., Qiao, M., Huo, L., & Wang, Z. (2017). Predicting head movement in panoramic video: A deep reinforcement learning approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(11), 2693–2708. https://doi.org/10.1109/TPAMI.2018.2858783

[21] Chopra, L., Chakraborty, D., & Mondal, A. (2021). PARIMA: Viewport adaptive 360-degree video streaming. *Proceedings of the Web Conference 2021* (pp. 2379–2391). https://doi.org/10.1145/3442381.3449857

[22] Li, J., Zhu, C., Xu, Z., Liu, Y., & Zhang, Z. (2023). Spherical convolution empowered FoV prediction in 360-degree video multicast with limited FoV feedback. *IEEE Transactions on Circuits and Systems for Video Technology*, *33*(12), 7245–7259. https://doi.org/10.1109/TCSVT.2023.3275976

[23] Li, C., Xu, M., Zhang, S., & Le Callet, P. (2019). Very long term field of view prediction for 360-degree video streaming. *2019 IEEE Conference on Multimedia Information Processing and Retrieval* (pp. 297–302). https://doi.org/10.1109/MIPR.2019.00061

[24] Fan, C., Lee, J., Lo, W., Huang, C., Chen, K., & Hsu, C. (2017). Fixation prediction for 360° video streaming in head-mounted virtual reality. *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video* (pp. 1–6). https://doi.org/10.1145/3083165.3083181