ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 9, No. 3, 49-61 (2024)
www.astesj.com

ASTES Journal
ISSN: 2415-6698

Efficient Deep Learning-Based Viewport Estimation for 360-Degree Video Streaming

Nguyen Viet Hung[*,1,2], Tran Thanh Lam[2], Tran Thanh Binh[1], Alan Marshal[3], Truong Thu Huong[2]

[1]*School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam*

[2]*Faculty of Information Technology, East Asia University of Technology, Bacninh, Vietnam*

[3]*Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool, United Kingdom*

ARTICLE INFO

Article history:
Received: 16 March, 2024
Revised: 14 May, 2024
Accepted: 29 May, 2024
Online: 12 June, 2024

Keywords:
Video Streaming
360-degree Video
QoE
VR
Deep Learning

ABSTRACT

While Virtual reality is becoming more popular, 360-degree video transmission over the Internet is challenging due to the video bandwidth. Viewport Adaptive Streaming (VAS) was proposed to reduce the network capacity demand of 360-degree video by transmitting lower quality video for the parts of the video that are not in the current viewport. Understanding how to forecast future user viewing behavior is therefore a crucial VAS concern. This study presents a new deep learning-based method for predicting the typical view for VAS systems. Our proposed solution is termed Head Eye Movement oriented Viewport Estimation based on Deep Learning (HEVEL). Our proposed model seeks to enhance the comprehension of visual attention dynamics by combining information from two modalities. Through rigorous experimental evaluations, we illustrate the efficacy of our approach versus existing models across a range of attention-based tasks. Specifically, viewport prediction performance is proven to outperform four reference methods in terms of precision, RMSE, and MAE.

1. Introduction

The proliferation of mobile head-mounted display (HMD) devices has led to the widespread adoption of 360-degree video streaming within the consumer video industry [1]. Since 360-degree videos offer users immersive settings and improve the Quality of Experience (QoE) of both video-on-demand and live-video streaming, they have seen a growing amount of public interest [2]. Compared to standard videos, 360-degree videos provide a more engaging viewing experience [3]. Therefore, VR streaming has begun to be used for live broadcasts of events like sporting contests and breaking news, giving viewers access to instantly immersive experiences [1].

In comparison to traditional flat videos, 360-degree videos necessitate significantly higher resolution and frame rates. For instance, to deliver an authentic experience to consumers, 360-degree videos are expected to have resolutions up to 24K and frame rates of 60fps [4, 5]. Consequently, streaming 360-degree videos demands considerable network bandwidth.

In contrast to the approximate 25 Mb/s bandwidth requirement for traditional 4K videos, streaming a 360-degree video requires about 400 Mb/s to deliver a 4K resolution while offering a complete 360-degree viewing range [6]. However, traditional 360-degree panoramic video streaming solutions are ineffective because they download the complete picture, while the viewer only sees a small portion of 360-degree video known as the viewport. This approach can result in the wastage of over 80% of network bandwidth [7].

Cutting-edge 360-degree video streaming techniques aim at decreasing video streaming bandwidth while retaining a good user experience. One of these is the Viewport Adaptive Streaming (VAS), which is a technique used in 360-degree video streaming to dynamically adjust the parameters based on the user's viewport. Instead of streaming the entire panoramic video, VAS delivers only the portion of the content that is currently visible to the user, optimizing bandwidth usage and improving streaming quality. The fundamental concept involves delivering the viewport (i.e., the portion of the video visible to a user) at a high bitrate (ensuring high quality), while the remaining parts are provided at a lower bitrate (ensuring lower quality) [8].

In Viewport Adaptive Streaming (VAS), these videos are segmented into tiles, each assigned weights corresponding to the user's viewport, and then transmitted accordingly. We can reduce the bandwidth of tiles that users overlook while maintaining the quality of tiles that users pay attention to by allocating the weights among the most likely tiles. However, as users view a 360-degree video,

[*]Corresponding Author: Truong Thu Huong, Address: Dai Co Viet, Bach Khoa, Hai Ba Trung, Hanoi, Vietnam & Email: huong.truongthu@hust.edu.vn

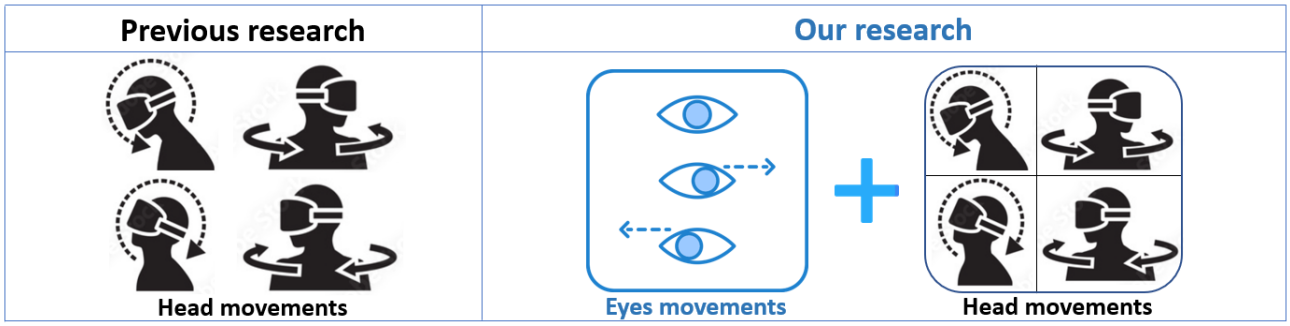


Figure 1: Head movements and eyes movements

their viewport positions frequently change. This dynamic nature underscores the necessity of viewport prediction. Viewport prediction refers to the process of forecasting where a user's viewport will be located in the near future. It is a fundamental component of delivering immersive 360-degree video experiences. By intelligently anticipating where the viewer's attention will be focused, streaming services can optimize resource usage, improve quality, reduce latency, enhance interactivity, and ultimately provide a more satisfying and immersive viewing experience.

Thus, a wrong viewport prediction will drastically lower the quality of the VAS system due to wasted bandwidth on unnoticed areas. Therefore, viewport prediction is a crucial requirement for VAS, which consequently requires a high level of precision in viewport prediction [9].

Nonetheless, forecasting the user's point of view with intensive accuracy is challenging. Users can shift their viewpoints as they rotate their heads in different directions. Additionally, the user's viewpoint may vary due to eye movements with or without moving their heads. Therefore, eye movements should be an important component to be considered. As Figure 1 illustrates, regarding the user's movements when watching 360 videos, while the recent methods only take the head movement into account, while our research considers both eye movements and head movements to determine the viewport more accurately.

Regarding viewport prediction, recently machine learning and deep learning-based solutions have been widely used for viewport prediction applications to enhance user experience [10, 11, 12, 13]. Based on past behavior and other characteristics, these solutions can learn and forecast the possibility that a user would pay attention to a particular area of the screen. Therefore, deployment of a proper Deep learning model that includes more actual behaviors of users potentially provides a more accurate predictive ability.

In this paper, we propose an accurate viewport prediction for 360-degree video viewport adaptive streaming that is based on a deep learning technique - LSTM and takes into account both the head and eye movements of viewers. Our solution is proven to outperform some current models like RNN [14], GRU [15], AEVE [16], and GLVP [9] in terms of Precision by 2.65%.

The remainder of this paper is structured as follows: The related work is discussed in Section 2. The proposed viewport estimation method is described in Section 3. The performance evaluation can be found in Section 4. Finally a discussion of our findings and pending issues is presented in 5.

2. Related work

Viewport adaptive streaming techniques have been proposed to address the bandwidth limitations of 360-degree videos, as evidenced by studies [5, 8, 17, 18, 19, 20]. These methods involve dividing the video into tiles and encoding each tile in multiple quality versions. The tiles within the user's viewport are transmitted in high quality, while the remaining tiles are delivered in lower quality [8]. This approach reduces redundancy in transmission and allows for flexible adaptation of tile quality based on the predicted viewport, taking into account network capacity constraints [21]. In the realm of Machine learning, researchers in [22] have employed machine learning techniques to anticipate future head movements, indirectly inferring future viewport positions. By analyzing past head movement data and leveraging statistical patterns observed in other viewers of spherical videos, the author utilizes regression models, including linear regression, to estimate the head orientation at each time point during the video. Similarly, linear regression has been used to predict motion-based fixation [13]. However, the authors in [23, 3] have shown that linear regression may be less impressive compared to alternative algorithms. In [23], algorithms such as average (Avg), linear regression (LR), and weighted LR (WLR) are compared, with the weighted LR yielding the best results. Furthermore, [3] demonstrates the performance of three machine learning models, including linear regression, Bayesian regression, and random forest, with random forest achieving the best results. Moreover, researchers in [7] proposed an advanced machine learning approach utilizing 3DCNN (3D-Convolutional Neural Networks) and LSTM-based RNN (Long Short-Term Memory) to extract spatiotemporal features from videos. This approach is proven to outperform other strategies in terms of accuracy and prediction [7].

However, it is important to note that all of the above models only consider the head movements to estimate the user's viewport positions, which could be a limitation since other factors, such as eye movement, also contribute to the viewport and may result in non-generalizable predictions. Thus, there is an overlooked potential for accuracy enhancement by incorporating both head and eye movements to the viewport prediction task.

Therefore, in this article, we design a solution that uses a deep-learning machine to forecast the viewport for the VAS system, using Long-Short Term Memory (LSTM) cells in Figure 2. This deep learning solution with LSTM can achieve the ideal balance between accuracy and redundancy [9]. During the learning process, head and eye movements are taken into account to increase the accuracy of

the precognition of future viewports.

This deep learning solution with LSTM can achieve the ideal balance between accuracy and redundancy [9]. During the learning process, head and eye movements are taken into account to increase the accuracy of the precognition of future viewports.

3. Proposed viewport estimation method - HEVEL

The proposed solution is named HEVEL as an acronym for **H**ead **E**ye Movement oriented **V**iewport **E**stimation based on Deep **L**earning. Before the **HEVEL** design is described, we will formulate the video streaming problem with the assumptions and constraints in Section 3.

3.1. Problem Formulation

On the one hand, we define several concepts below to address the viewport prediction problem. Assume $P(t_0)$ is the position of the viewport at time t_0. A viewport's center point can be located by using the longitude and latitude values of the viewport [24, 25]. Figure 3 illustrates how spherical video captures a scene from every angle. It is the primary form of content used in Virtual Reality, giving viewers an immersive experience. The viewport is the portion of the video that a user can currently watch due to their field of vision.

On the other hand, predicting where the viewport $P(t_0 + m)$ will be in time t_0 in the future is the responsibility of the viewport predictor. At the same time, the letter m is used to indicate the forecast horizon. As shown in Figure 4, the predictor must offer a prediction for the interval $[t_0 + m, t_0 + m + s]$, where s stands for the segment duration because 360-video streaming is commonly done on a segment/adaptation interval basis [26, 27].

3.2. Viewport prediction and selection framework

In our proposed HEVEL framework, we employ the LSTM model because it can capture and model long-term dependencies within input data. By maintaining an internal state capable of retaining information from previous time steps, LSTM helps the HEVEL framework effectively model and utilize historical information of head and eyes movements, thereby enhancing its predictive or analytical capabilities.

Modeling these long-term dependencies is achieved through gating mechanisms that selectively enable the network to hold or discard information from past time steps [28]. In addition, HEVEL also helps to save memory usage since it eliminates unnecessary positions by adapting to both head and eye movements. As a result, our approach requires less memory while ensuring accurate viewport predictions. Each of the n inputs in the Long Short-Term Memory (LSTM) model corresponds to n video frames or images that have been taken out of a video. $\{x_1, x_2, ...x_{n-1}, x_n\}$ is a representation of this sequence. Each input x_i represents the visual information present in the video sequence's i-th frame. The LSTM analyzes this series of inputs in time steps while taking into consideration their temporal relationship.

The LSTM creates an output known as y_t at each time step t. This output is the anticipated viewport, which is the precise region of interest within the video frame that a viewer is most likely to pay attention to at that exact time step. The geographical position and visual information that are thought to be most important to the viewer's attention at the given time instant are essentially encapsulated by y_t.

In LSTM, the cell represents the long-term memory component. It is responsible for storing and updating information over time. The cell maintains a state vector that carries information from previous time steps and is passed along through the recurrent connections. In contrast, the hidden state represents the output of the LSTM unit at a given time step. It carries information relevant for making predictions at the current time step stored in the cell. Moreover, in LSTM network design, gates play an important role in regulating the flow of information through the network. Its purpose is to determine how much of the new information should be stored in the cell state. As Figure 5 illustrates, c_{t-1}, h_{t-1} are the cell state and hidden cell state respectively at time $t-1$. While c_t, h_t are the cell state and hidden state at time t that will be used as input for the next cell. At the end, h_t represents output y_t - the predicted viewport.

Next, $W_{i\alpha}$, $W_{i\beta}$, W_f, $W_{o\alpha}$, $W_{o\beta}$ are the recurrent weight matrices, $b_{i\alpha}$, $b_{i\beta}$, b_f, $b_{o\alpha}$, $b_{o\beta}$ are the bias terms.

Additionally, we employ tanh as the tanh function and σ as the logistic *sigmoid* function. The gates used in the cell are defined as follows:

- **Input gate -** i_t, $i_{\alpha t}$, $i_{\beta t}$: to decide which values should be stored in the cell state.
- **Forget gate -** f_t: to decide which information from the long-term memory should be kept or discarded.
- **Output gate -** $o_{\alpha t}$, $o_{\beta t}$, y_t: to decide which data from the current cell will be managed as output.

In more detail, the working mechanism of those gates can be described as follows:

Input gate: The input gate helps regulating the process of deciding what to forget and what new information to store in the cell state. The input gate with data from the current input x_t and short-term memory from the last time step h_{t-1} to determine what brand-new data should be kept. In this part, we have designed the input interface in two layers:

- **First layer:** The sigmoid activation function is used in the first layer to help an LSTM cell decide which information from the input and previous cell state is important to retain or forget. Because the Sigmoid Activation Function turns any input into a number between 0 and 1, the output determines what percentage of the Long-term Memory is remembered. Output 0 indicates that part of the information is unimportant, akin to forgetting that piece of information. In contrast, 1 suggests that the information will be retained. This is crucial for managing the memory of past viewport positions and deciding what new information to incorporate. Formula 1 encapsulates this process.

$$i_{\alpha t} = \sigma(W_{i\alpha} \otimes (h_{t-1}, x_t) + b_{i\alpha}) \quad (1)$$

- **Second layer:** The tanh activation function is used in the second layer to regulate the output gate of LSTM cells. To normalize the output values $i_{\beta t}$ obtained from equation (2) in the LSTM architecture, we normalize the values from the

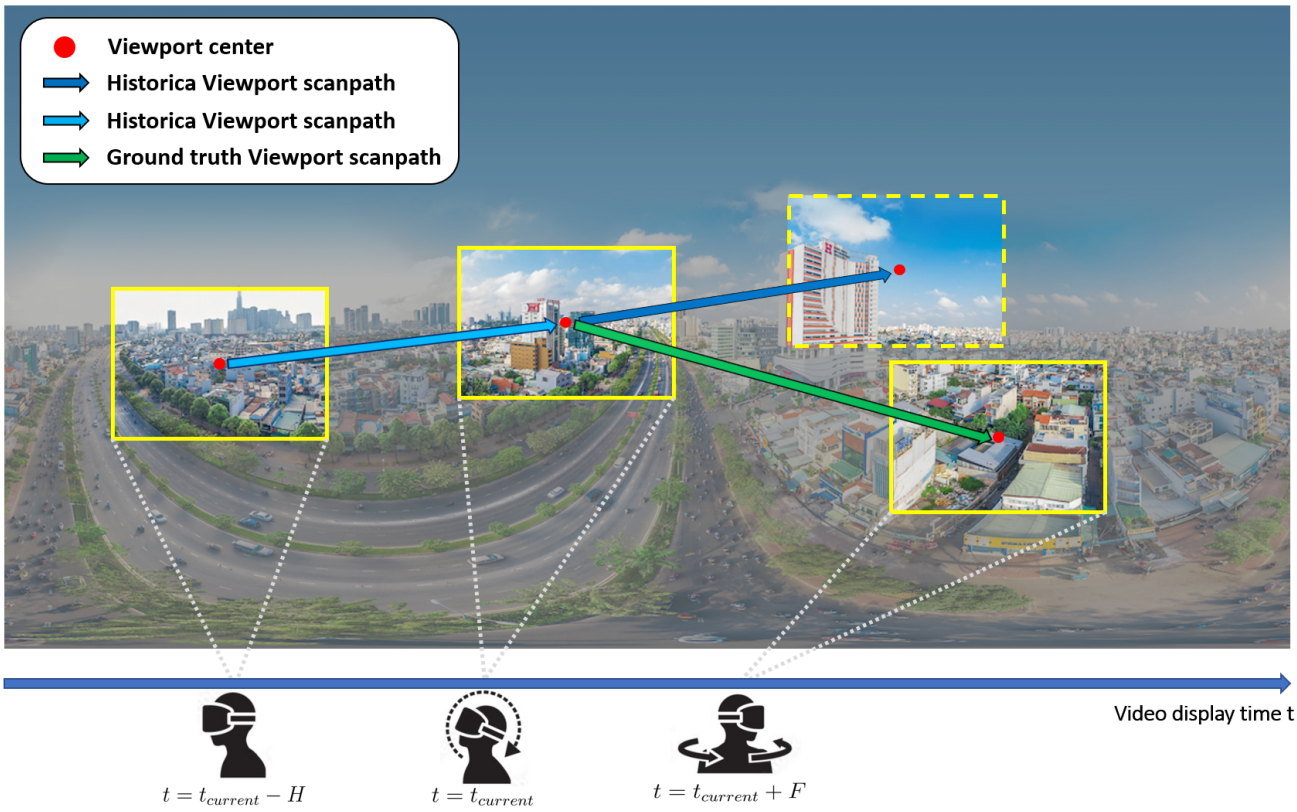


Figure 2: The Proposed model compares the viewport sweep in the past H-seconds to predict the viewport sweep in the future F seconds.

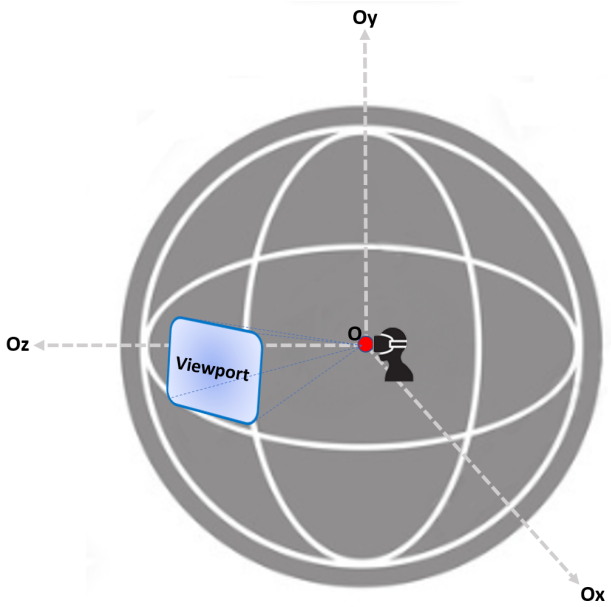


Figure 3: The viewport of a user at a time

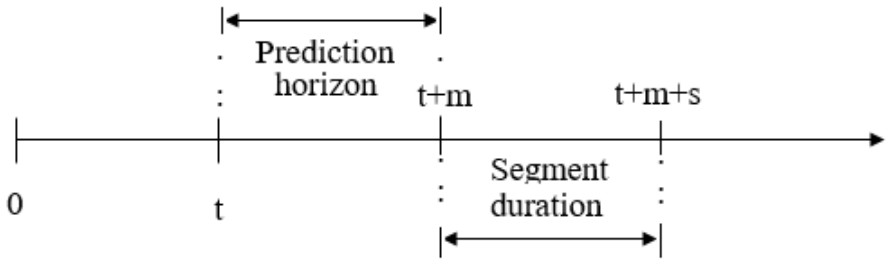


Figure 4: Problem formulation of Viewport Prediction

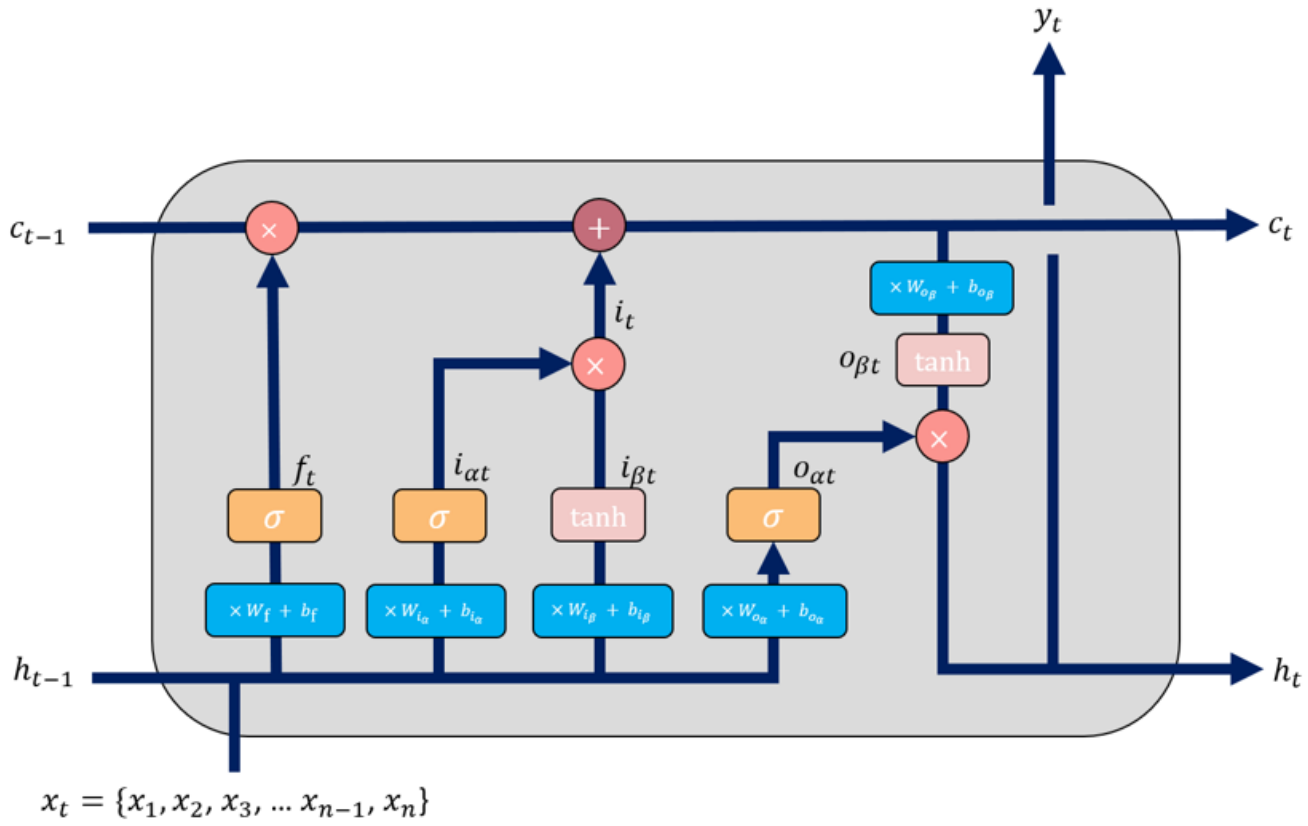


Figure 5: LSTM model for viewport estimation

Table 1: Characteristics of the five experimental videos

Video	Description
Turtle	People are releasing baby turtles into the sea on the beach during the day
Bar	Light, users moving, bartender at work
Ocean	Under the ocean, people are going underwater to see whales.
Sofa	People are sitting on sofas in the living room to talk
Po. Riverside	Riverside, outdoor, during the day, with human activities

range [-1, 1] to [0, 1] by the formula: $no_v = (ot_v + 1) / 2$, with ot_v represents the value obtained from the LSTM cell within the range [-1, 1], and no_v represents the corresponding normalized value within the range [0, 1]. Layer 2 combines the Short-term memory and the input to create a potential long-term memory. Formula 2 shows the process.

$$i_{\beta t} = \tanh(W_{i\beta} \otimes (h_{t-1}, x_t) + b_{i\beta}) \quad (2)$$

Note that, placing the tanh activation function in Layer 1 might disrupt the gating mechanism of the LSTM cell. The purpose of using sigmoid in Layer 1 is to regulate the flow of information, deciding what information to keep and discard. Tanh activation in this context might not provide the necessary gating behavior required for effective memory management.

Finally, we calculate i_t by multiplying the above two layers, and obtain the input gate by Formula 3:

$$i_t = i_{\alpha t} * i_{\beta t} \quad (3)$$

Forget gate: The forget gate f_t (Formula 4) selects which data from the long-term memory should be retained or erased. This is calculated using the previously hidden state h_{t-1} and the current input x_t with the *sigmoid* function, similarly to the first input layer $i_{\alpha t}$ but with different weights.

$$f_t = \sigma(W_f \otimes (h_{t-1}, x_t) + b_f) \quad (4)$$

The forget gate f_t is a scalar value between 0 and 1 used to weigh the previous cell state when combined with the current input to update the cell state. A value of 0 indicates the last cell state has been completely forgotten. Whereas a value of 1 indicates that have should be retained entirely. The LSTM cell can simulate long-term dependencies and recall pertinent information over time thanks to the forget gate's assistance in selectively maintaining and forgetting information from the input sequence.

Cell state: The cell state c_t (Formula 5) is a new feature of LSTM based on RNN to overcome the memory limitation of the old algorithm. It is a "memory" in an LSTM cell that stores information from the input sequence over time. The update at each time step is based on the input x_t, forget f_t, and the previous cell state c_{t-1}. The outcomes of the **Input** and **Forget gates** will be added pointwise to create a new version of the long-term memory that will be sent to the next cell.

$$c_t = c_{c-1} * f_t + i_t \quad (5)$$

Furthermore, the new long-term memory will be employed in the output gate, which is the last gate.

Output gate: The following hidden state's value is determined. This state stores information from previous inputs. Two layers can be described as follows:

- **First layer:** Once again, the current input and the hidden state are passed into a sigmoid function to generate the last filter $o_{\alpha t}$ (Formula 6):

$$o_{\alpha t} = \sigma(W_{o\alpha} \otimes (h_{t-1}, x_t) + b_{o\alpha}) \quad (6)$$

- **Second layer:** an activation tanh function is implemented on the newly created long-term memory from the cell state in Formula 7:

$$o_{\beta t} = \tanh(W_{o\beta} \otimes c_t + b_{o\beta}) \quad (7)$$

Finally, using Formula 6 and 7, new short-term memory is calculated according to the formula below:

$$h_t, y_t = o_{\alpha t} * o_{\beta t} \quad (8)$$

The new short-term memory refers to the hidden state h_t, and long-term memory c_t will be moved to the next cell, where the process will repeat. The outcome of each step will also derive from h_t, which we define as y_t. The Pseudo code in Algorithm 1 summarizes the entire process of estimating viewports.

4. Performance Evaluation

4.1. Experimental Settings

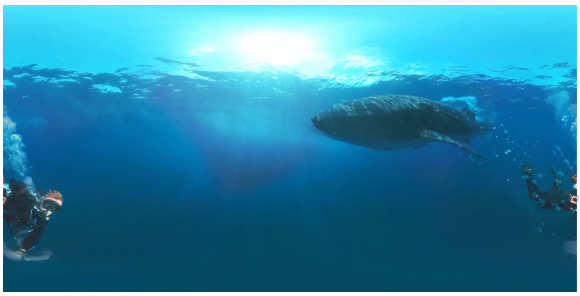
In our experiment, we employ five 360-degree videos with different contexts, as shown in Table 1, and Figure 6. The videos depict a variety of situations and environments. Using such a diverse set of videos, we can assess the model's ability to deal with diversified situations. For example, video Turle possesses turtle movement that may cause a user's eyes and head to continuously move to track this object location. While video Bar presents a complex scenario with its varying lighting and conditions, human presence, and sound. This intricate mix of factors may cause a user to focus at some random spots from one time to another time. In video Ocean, space, height, frequency, and direction of waves and marine life will affect observations through water clarity and visibility. Video Sofa offers a static setting, with only small movements and tends to be quiet in sound. Video Po. Riverside Video shows vivid and exciting portraits of the plant world, surrounding trees and birds, insects, and other animals. Each video contains a corresponding head-eye motion trace. Even while the head position is fixed, the eyes can still move to different positions and change the viewport. The video's head-eye movement traces are obtained from the dataset in [29] and illustrated in Figure 7. Each 60-second video is encoded and projected using equirectangular projection with a 4K quality (3840 × 1920). The video is divided into 24 tiles, each with a resolution of 480 × 480. In this context, "pitch" refers to the user's head-eye position's longitude, which ranges from -180 to 180 degrees. "Yaw" represents the latitude and has a range of -90 to 90 degrees.

On the other hand, we conduct the experiments using the same five videos to facilitate an unbiased comparison of the performance of HEVEL with the reference models.

4.2. Viewport prediction performance

The viewport prediction performance of HEVEL is compared with the current reference models such as GLVP [9], RNN [14], GRU [15], and AEVE [16] in terms of Precision, RMSE (Root Mean Square Error), MAE (Mean Absolute Error).

(a) Bar



(b) Ocean

(c) Porto Riverside

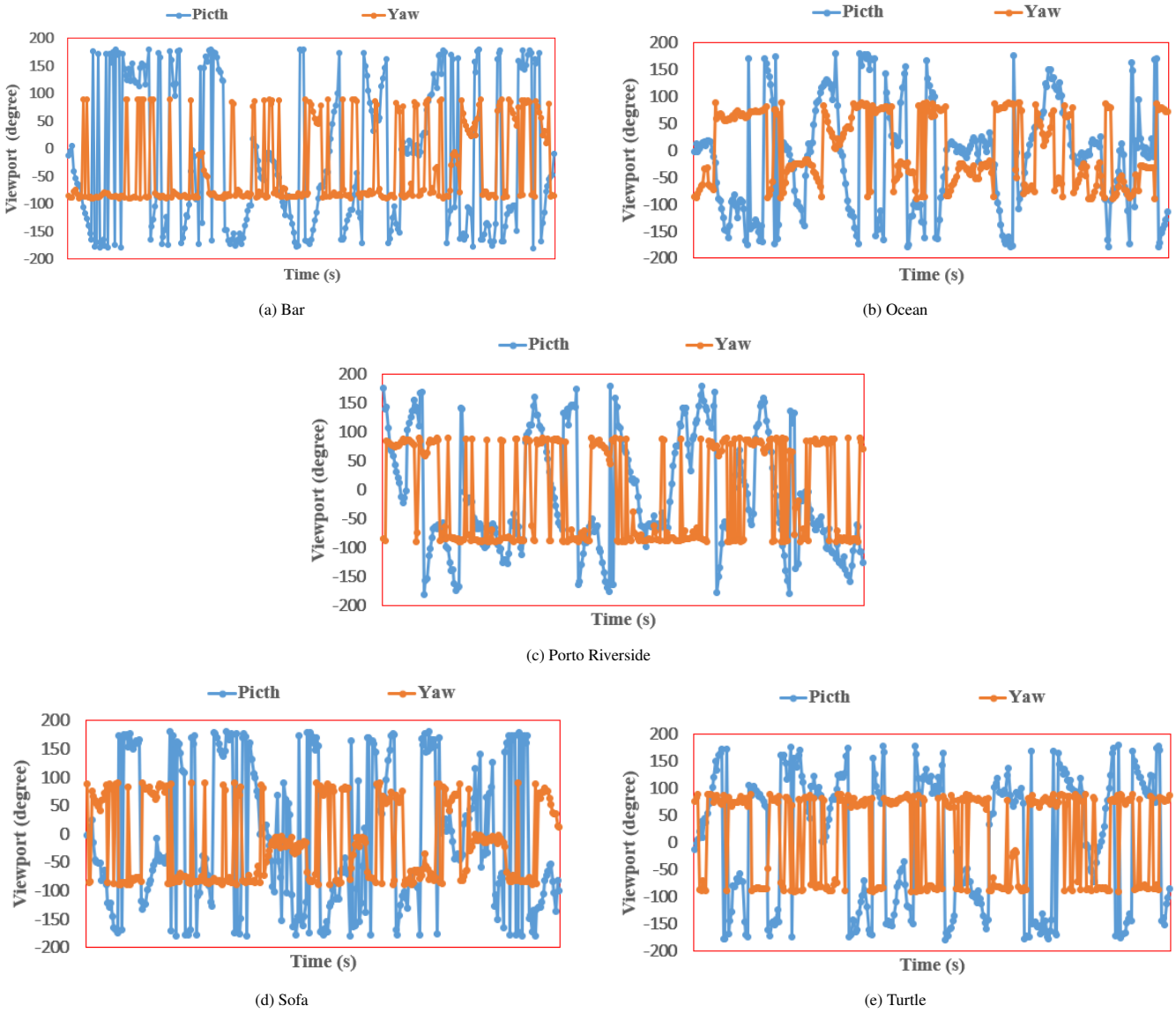
(d) Sofa

(e) Turtle

Figure 6: Experimental 360-Degree videos

Algorithm 1: *Viewport Estimation*

Input: c_{t-1}, h_{t-1}, x_t
Output: c_t, h_t, y_t
for $t = 1$ ***to*** N **do**
  Calculate $i_{\alpha t} = \sigma(W_{i\alpha} \otimes (h_{t-1}, x_t) + b_{i\alpha})$
  Calculate $i_{\beta t} = \tanh(W_{i\beta} \otimes (h_{t-1}, x_t) + b_{i\beta})$
  Calculate $i_t = i_{\alpha t} \odot i_{\beta t}$
  Calculate $f_t = \sigma(W_f \otimes (h_{t-1}, x_t) + b_f)$
  Calculate $c_t = c_{t-1} \odot f_t + i_t$
  Calculate $o_{\alpha t} = \sigma(W_{o\alpha} \otimes (h_{t-1}, x_t) + b_{o\alpha})$
  Calculate $o_{\beta t} = \tanh(W_{o\beta} \otimes (c_t) + b_{o\beta})$
  Calculate $h_t = o_{\alpha t} \odot o_{\beta t}$
  Calculate $y_t = h_t$
end



(a) Bar

(b) Ocean

(c) Porto Riverside

(d) Sofa

(e) Turtle

Figure 7: Head and eye movements of experimental videos over time

Precision: a fraction of correctly-predicted viewport positions relative to all the viewport positions predicted.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

Where:

- TP is the number of correctly predicted viewport positions.
- FP is the number of incorrectly predicted viewport positions.

In addition, to evaluate the precision of our prediction solution, each VR video is divided into small cells as illustrated in Fig. 8. Within Figure 8, you can observe two regions depicting the actual and predicted regions evolving over time. When the expected and actual regions overlap, as demonstrated in Figure 8a, the prediction is classified as a TP (True Positive). If the predicted and actual regions fail to overlap, the prediction is considered an FP (False Positive), as shown in Fig. 8b. If the overlapped area of the actual and estimated viewport exceeds a threshold of 50%, it is considered a TP; otherwise, it is counted as an FP.

RMSE: the average magnitude of the errors between predicted and observed viewport positions.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{N}(Act_t - Pre_t)^2}{N}} \tag{10}$$

MAE: the average absolute magnitude of the errors between predicted and observed (true) viewport positions.

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|Act_t - Pre_t| \tag{11}$$

Where:

- Pre_t is the predicted viewport position for data point i
- Act_t is the actual viewport position in the testing data set
- N is the number of data points.

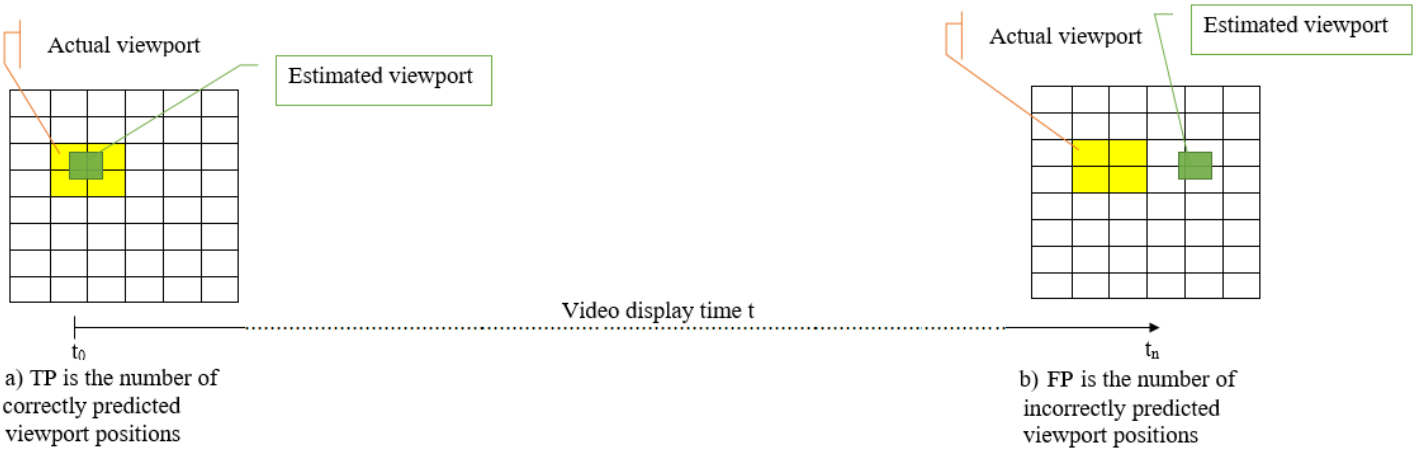


Figure 8: Calculation of TP and FP

Table 2: Accuracy (%) of HEVEL compared to the reference methods

Videos	GRU	RNN	GLVP	HEVEL
BAR	61.65	62.12	74.21	**84.54**
Ocean	60.52	71.30	73.21	**85.86**
Po. Riverside	88.65	87.23	90.11	**91.62**
Sofa	74.21	58.16	74.21	**85.86**
Turtle	72.57	71.62	74.21	**75.58**
Average	**70.09**	**68.62**	**76.64**	**84.54**

For the RMSE metric, as shown in Figure 9, the CDF curves for HEVEL is the steepest curve among the curves for RNN, AEVE, GRU,GLVP, this means that HEVEL generally exhibits smaller RMSE values compared to those reference methods across a range of data points. This suggests that HEVEL exhibits better prediction performance than the reference methods.

For the MAE metric, as shown in Figure 10, the CDF for HEVEL is also steeper than the curves for RNN, AEVE, GRU, and GLVP. The results demonstrate that those reference methods generally have larger MAE values compared to HEVEL across a range of data points. This suggests that HEVEL is better than those reference methods in terms of MAE.

The result is shown in Figure 11 in which HEVEL has the steepest CDF curve. It indicates that HEVEL consistently achieves higher precision across various decision thresholds. Therefore, HEVEL is proven to be more effective at correctly identifying correct viewport positions and avoiding false viewport prediction compared to the other reference methods.

As Table 2 shows, HEVEL improve accuracy in comparison with to the reference methods by from 1.37% to 27.71%. It consistently exhibits higher accuracy, with the lowest recorded accuracy in the Turtle video being approximately 75.58%. In the Ocean video, the proposed method surpasses GRU by 25.35%, RNN by 14.56%, and GLVP by 12.65% in terms of accuracy.

Therefore, HEVEL is proven to be more effective at correctly identifying correct viewport positions and avoiding false viewport prediction compared to the other reference methods.

The findings indicated above can come from the facts that LSTM, leveraging its capacity to grasp long-term relationships in sequential data, effectively preserves and transmits information across extended sequences. This makes it particularly adept for tasks such as viewport prediction, which heavily relies on prolonged dependencies, consequently enhancing overall performance. Moreover, LSTM allows it to selectively update and forget unnecessary information, allowing the model to focus on relevant information while ignoring noise input. This helps the LSTM process be more effective.

LSTM's gating mechanism and memory cell provide a more stable and efficient training process than other models like GLVP, GRU, RNN, and AEVE. This stability and efficiency ensure more reliable convergence and improved performance, even in challenging scenarios like shaky videos.

Moreover, since HEVEL incorporates eye and head movements, it gains a more holistic perspective on the user's focus and intention. This input understanding allows the model to make more accurate prediction and adapt to various scenarios, leading to improved performance.

In summary, our prediction model has been proved to guarantee users to be consistently provided the most relevant and engaging content at any given moment.

4.3. Training time evaluation

In general, for such a video application, once a viewport prediction model has been built, we can use it for a long time. Because the data pattern and user's behavior of such an application do not change rapidly over time. However, in some specific case, it might be necessary to re-train the learning model regularly in real-time. In that case is good to have more insight into the training time of the

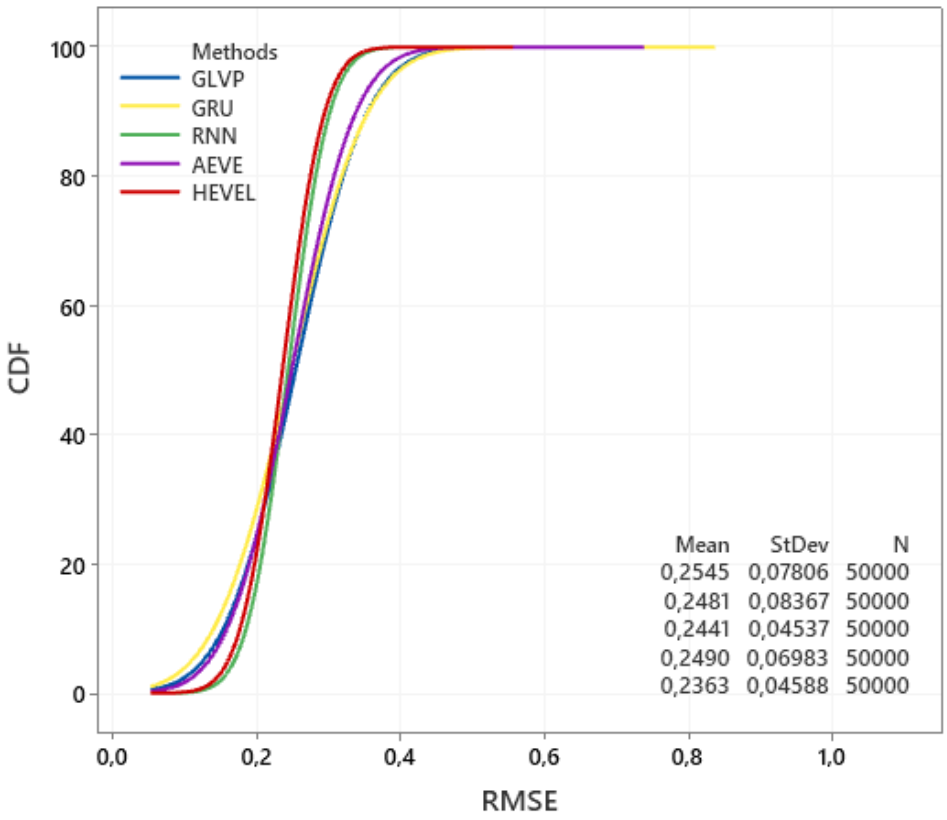


Figure 9: CDF of RMSE of HEVEL vs. the reference methods

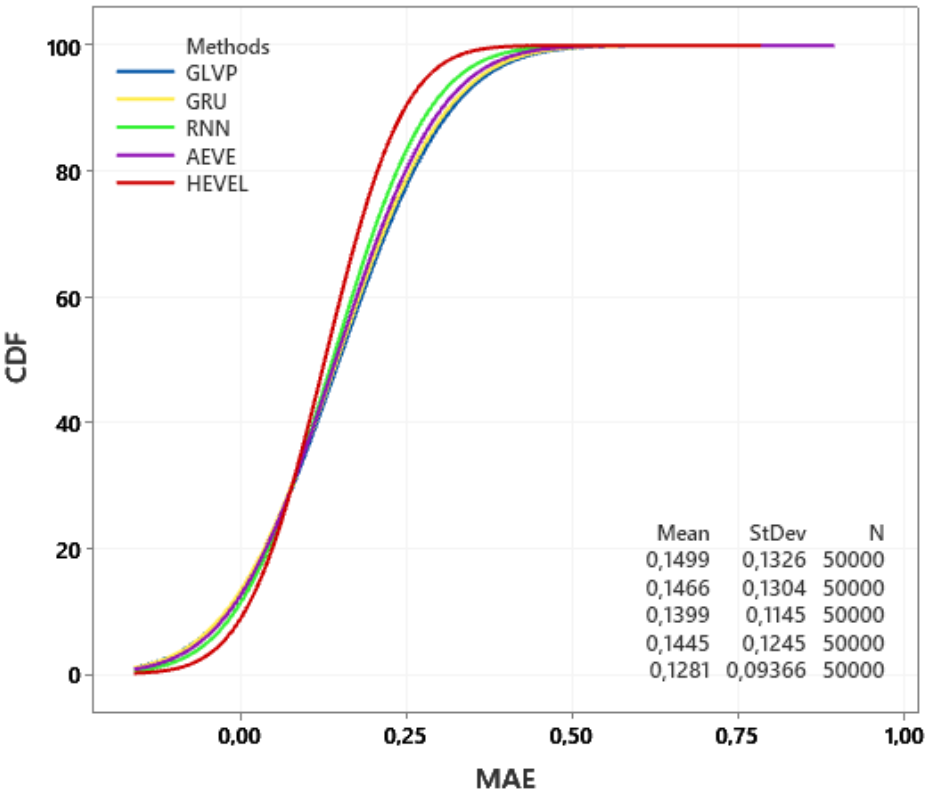


Figure 10: CDF of MAE of HEVEL vs. the reference methods

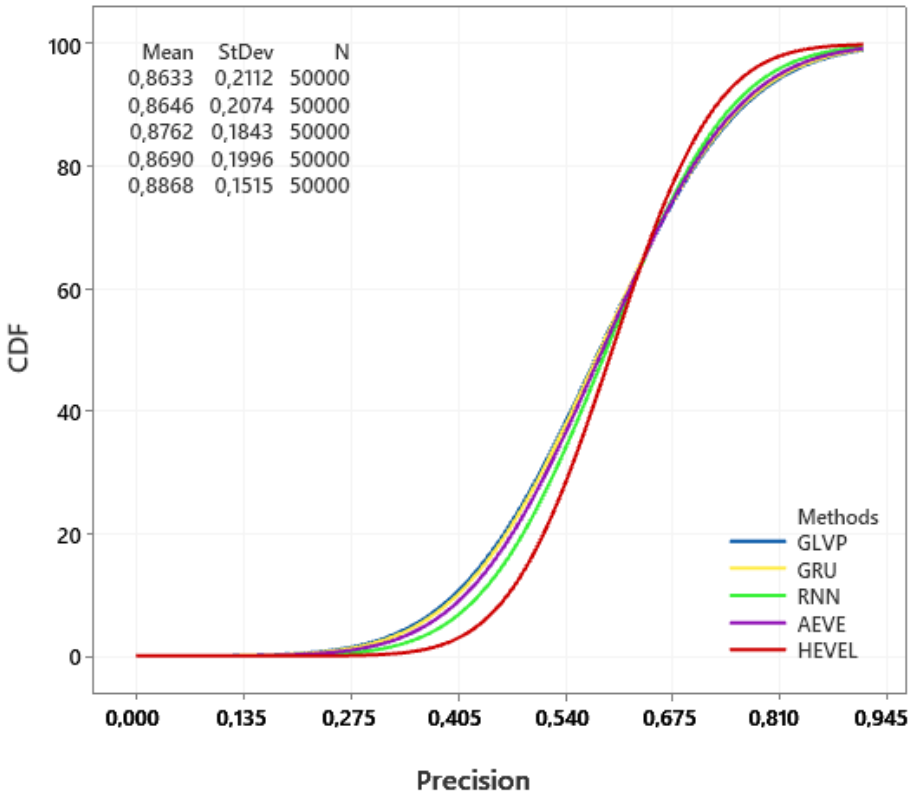


Figure 11: CDF of Precision of HEVEL vs. the reference methods

current prediction model solutions.

Figure 12 shows that the reference methods GRU, AEVE, GLVP, and our proposed HEVEL take a training time of more or less than 0.1s. In summary, our approach addresses the dual concerns of training efficiency and real-time responsiveness. The experiment's outcomes firmly position our proposed HEVEL method as a promising solution for practical applications on devices with inherent resource constraints.

5. Conclusions, Discussion and Future work

In this research, we have delved into a specific focus on addressing the challenging viewport prediction task. The proposed solution has been proven to outperform the 4 reference methods in several critical evaluation metrics, including Precision, RMSE, and MAE. By accurately predicting the user's viewport, VR video streaming has taken one more step toward reducing the latency and improving the quality of VR content delivery, ultimately enhancing user satisfaction on a more immersive and enjoyable VR services in multiple domains, including gaming, education, training.

For the future work, there are opportunities for further exploration and refinement of our approach such as adapting our solution to different VR hardware configurations. It would entail tailoring our approach to work seamlessly with the various VR devices that are currently available. This could include popular headsets like the Oculus Rift, HTC Vive, or PlayStation VR, as well as emerging technologies like augmented reality (AR) glasses. Each hardware configuration may have distinct technical specifications and interaction mechanisms, necessitating adjustments to improve our solution's performance and usability. In addition, we need understanding and modeling more user behaviors in VR environments to improve the overall VR experience. Our approach could concentrate on specific user behaviors, like locomotion and object manipulation. However, many other aspects of user interaction and engagement in VR can be investigated and integrated. This could entail researching and incorporating interaction patterns such as more behaviors, communication cues, and collaborative actions to create more immersive and realistic virtual worlds.

Furthermore, more research could be done to investigate user preferences, comfort levels, and physiological responses in VR. Understanding how users perceive and interact with virtual environments allows us to tailor our approach to better meet their needs and preferences. This could include conducting user studies, gathering feedback, and iteratively optimizing our solution based on the findings.

Another space for exploration can be investigation on incremental or online learning paradigm where the model is trained continuously as new data becomes available, without needing to retrain on the entire dataset.

The investigation can be extended to combining LSTM with other deep learning model such as GRU in order to boost up the training and processing data like VR services faster.

Last but not least, using information about the head and eyes movement may pose some challenges in capturing long-term depen-

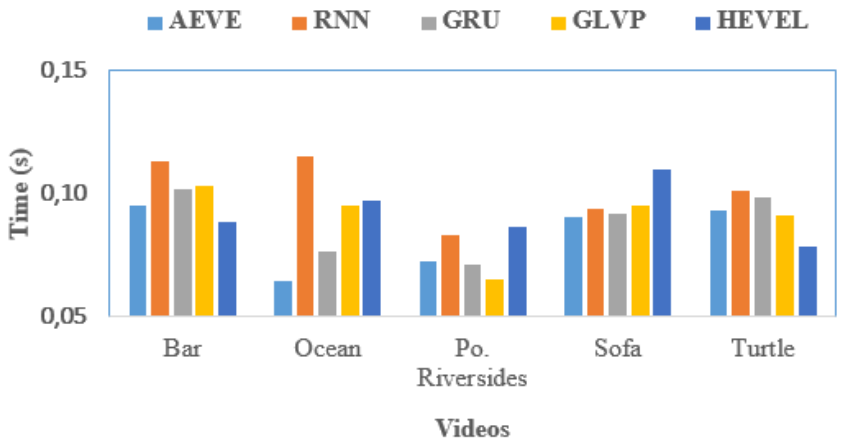


Figure 12: Training time overview

dencies. That is, if there is a long period between when the person looks at an object and then moves their head to look at a new location. This creates a significant time delay between related events. LSTM is designed to understand and remember long-term data patterns through its LSTM gate mechanism, helping it retain important information and forget unnecessary information. However, if the delay between events is too large, LSTM may struggle to maintain long-term dependencies and retain complete information. Thus, our next work will involve around the question: how to preprocess the input data to cope with the large delay between 2 events.

Acknowledgment

This research is funded by the Hanoi University of Science and Technology (HUST) under Project number T2022-PC-012; and partly funded by Project 2395 of the Ministry of Science and Technology of Vietnam.

References

[1] X. Feng, Y. Liu, S. Wei, "LiveDeep: Online Viewport Prediction for Live Virtual Reality Streaming Using Lifelong Deep Learning," in 2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR), 800–808, 2020, doi:10.1109/VR46266.2020.00104.

[2] Y. Zhang, et al, "DRL360: 360-degree Video Streaming with Deep Reinforcement Learning," in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, 1252–1260, 2019, doi:10.1109/INFOCOM.2019.8737361.

[3] M. M. Uddin, J. Park, "Machine learning model evaluation for 360° video caching," in 2022 IEEE World AI IoT Congress (AIIoT), 238–244, 2022, doi:10.1109/AIIoT54504.2022.9817292.

[4] N. V. Hung, B. D. Tien, T. T. T. Anh, P. N. Nam, T. T. Huong, "An efficient approach to terminate 360-video stream on HTTP/3," in AIP Conference Proceedings, volume 2909, AIP Publishing, 2023.

[5] D. V. Nguyen, et al, "Scalable 360 Video Streaming using HTTP/2," in 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP), 1–6, 2019, doi:10.1109/MMSP.2019.8901805.

[6] M. Zink, et al, "Scalable 360° Video Stream Delivery: Challenges, Solutions, and Opportunities," Proceedings of the IEEE, **107**(4), 639–650, 2019, doi:10.1109/JPROC.2019.2894817.

[7] S. Park, A. Bhattacharya, Z. Yang, S. R. Das, D. Samaras, "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, 2021, doi:10.1109/TNSM.2021.3053183.

[8] H. L. Dieu Huong, et al, "Smooth Viewport Bitrate Adaptation for 360 Video Streaming," in 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), 512–517, 2019, doi:10.1109/NICS48868.2019.9023807.

[9] H. Nguyen, et al, "An Accurate Viewport Estimation Method for 360 Video Streaming using Deep Learning," EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, **9**(4), e2, 2022, doi:10.4108/eetinis.v9i4.2218.

[10] Y. Jiang, et al, "Robust and Resource-efficient Machine Learning Aided Viewport Prediction in Virtual Reality," arXiv preprint arXiv:2212.09945, 2022.

[11] J. Adhuran, et al, "Deep Learning and Bidirectional Optical Flow Based Viewport Predictions for 360° Video Coding," IEEE Access, **10**, 118380–118396, 2022, doi:10.1109/ACCESS.2022.3219861.

[12] L. Zhang, et al, "MFVP: Mobile-Friendly Viewport Prediction for Live 360-Degree Video Streaming," in 2022 IEEE International Conference on Multimedia and Expo (ICME), 1–6, 2022, doi:10.1109/ICME52920.2022.9859789.

[13] Y. Ban, et al, "Exploiting Cross-Users Behaviors for Viewport Prediction in 360 Video Adaptive Streaming," in 2018 IEEE International Conference on Multimedia and Expo (ICME), 1–6, 2018, doi:10.1109/ICME.2018.8486606.

[14] C.-L. Fan, et al, "Optimizing Fixation Prediction Using Recurrent Neural Networks for 360° Video Streaming in Head-Mounted Virtual Reality," IEEE Transactions on Multimedia, **22**(3), 744–759, 2020, doi:10.1109/TMM.2019.2931807.

[15] C. Wu, R. Zhang, Z. Wang, L. Sun, "A Spherical Convolution Approach for Learning Long Term Viewport Prediction in 360 Immersive Video," Proceedings of the AAAI Conference on Artificial Intelligence, **34**(01), 14003–14040, 2020, doi:10.1609/aaai.v34i01.7377.

[16] D. Nguyen, "An evaluation of viewport estimation methods in 360-degree video streaming," in 2022 7th International Conference on Business and Industrial Research (ICBIR), 161–166, IEEE, 2022, doi:10.1109/ICBIR54589.2022.9786513.

[17] D. V. Nguyen, et al, "An Optimal Tile-Based Approach for Viewport-Adaptive 360-Degree Video Streaming," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, **9**(1), 29–42, 2019, doi:10.1109/JETCAS.2019.2899488.

[18] D. Nguyen, et al, "Scalable multicast for live 360-degree video streaming over mobile networks," IEEE Access, **10**, 38802–38812, 2022, doi:10.1109/ACCESS.2022.3165657.

[19] N. V. Hung, P. H. Thinh, N. H. Thanh, T. T. Lam, T. T. Hien, V. T. Ninh, T. T. Huong, "LVSUM-Optimized Live 360 Degree Video Streaming in Unicast and Multicast Over Mobile Networks," in 2023 IEEE 15th International Conference on Computational Intelligence and Communication Networks (CICN), 29–34, IEEE, 2023, doi:10.1109/CICN59264.2023.10402136.

[20] N. V. Hung, et al, "Flexible HTTP-based Video Adaptive Streaming for good QoE during sudden bandwidth drops," EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, **10**(2), e3–e3, 2023, doi:10.4108/eetinis.v10i2.2994.

[21] N. Kan, et al, "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, **32**(3), 1607–1623, 2022, doi:10.1109/TCSVT.2021.3076585.

[22] J. Vielhaben, et al, "Viewport Forecasting in 360° Virtual Reality Videos with Machine Learning," in 2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), 74–747, 2019, doi:10.1109/AIVR46125.2019.00020.

[23] F. Qian, et al, "Optimizing 360 Video Delivery over Cellular Networks," in Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges, ATC '16, 1–6, Association for Computing Machinery, New York, NY, USA, 2016, doi:10.1145/2980055.2980056.

[24] D. V. Nguyen, et al, "An Evaluation of Tile Selection Methods for Viewport-Adaptive Streaming of 360-Degree Video," ACM Trans. Multimedia Comput. Commun. Appl., **16**(1), 2020, doi:10.1145/3373359.

[25] V. H. Nguyen, et al, "Retina-based quality assessment of tile-coded 360-degree videos," EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, **9**(32), 2022.

[26] T. C. Thang, et al, "An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming," IEEE Journal on Selected Areas in Communications, **32**(4), 693–705, 2014, doi:10.1109/JSAC.2014.140403.

[27] V. H. Nguyen, D. T. Bui, T. L. Tran, C. T. Truong, T. H. Truong, "Scalable and resilient 360-degree-video adaptive streaming over HTTP/2 against sudden network drops," Computer Communications, **216**, 1–15, 2024, doi:10.1016/j.comcom.2024.01.001.

[28] T. T. Huong, et al, "Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach," Computers in Industry, **132**, 103509, 2021, doi:10.1016/j.compind.2021.103509.

[29] E. J. David, et al, "A dataset of head and eye movements for 360 videos," in Proceedings of the 9th ACM Multimedia Systems Conference, 432–437, 2018, doi:10.1145/3204949.3208139.