Viewport-Aware Edge Caching for 360° Video Streaming: A Compact State Actor-Critic Approach

Eduardo S. Gama, Eirina Bourtsoulatze and Nikolaos Thomos
School of Computer Science and Electronic Engineering at the University of Essex, UK
{es25591, e.bourtsoulatze and nthomos}@essex.ac.uk

***Abstract*—The growing popularity of immersive media applications calls for efficient 360° video streaming methods. However, delivering high-resolution spherical content imposes severe bandwidth demands that challenge modern networks. While viewport-adaptive tile-based streaming mitigates these demands by transmitting only the user's current Field of View (FoV), delayed tile fetching due to inaccurate viewport prediction can degrade the user's Quality of Experience (QoE). Edge caching offers an effective strategy to mitigate this problem, particularly when overlapping user requests are aggregated into virtual viewports. In this paper, to optimize caching performance, we propose a novel Deep Reinforcement Learning (DRL)-based mechanism for adaptive edge caching in 360° video streaming. The proposed approach formulates cache management as a sequential decision-making problem, enabling dynamic optimization of tile caching and replacement policies based on evolving system states. Specifically, our framework employs an Advantage Actor-Critic (A2C) architecture combined with Generalized Advantage Estimation (GAE) to improve learning stability in non-stationary caching environments. The simulation results demonstrate that our method significantly outperforms baseline algorithms in terms of PSNR, cache hit ratio, and backhaul traffic reduction. In particular, our A2C-based caching policy achieves a 10% to 20% improvement in cache hit ratio while increasing PSNR by approximately 2 dB.**

***Index Terms*—Deep Reinforcement Learning, Advantage Actor-Critic, 360° video, tile-encoding, viewport-aware caching**

I. Introduction

Immersive media applications, including Virtual Reality (VR) and Augmented Reality (AR), have surged in popularity with the emergence of 5G and forthcoming 6G networks [1], [2]. A core enabler of these applications is 360° video streaming, which allows users to navigate immersive spherical content. However, delivering such content remains challenging due to substantial bandwidth demands, as the delivered content should be high quality, and therefore it requires high data rates [3], [4]. These challenges call for efficient 360° video streaming methods.

To address this challenge, streaming systems employ viewport-aware, tile-based video streaming techniques that reduce transmission overhead by delivering only the user's current Field of View (FoV) at high quality [5], [6], [7]. In this approach, the spherical video is partitioned into non-overlapping regions that are encoded independently, referred to as tiles, and only the tiles corresponding to the user's viewport are streamed at the highest quality. Although this approach significantly reduces bandwidth usage, it may result in severe degradation of the user's Quality of Experience (QoE) when accurate viewport prediction fails. To further mitigate this issue, advanced systems often combine tile-based streaming with scalable video coding techniques, which can be particularly effective in heterogeneous networks serving multiple users with diverse viewport and video demands [7].

Edge caching networks enhance video delivery by enabling cache reuse by accounting for users' overlapping demands for both videos and viewports [8], [9], [7]. Specifically, the edge can alleviate core network congestion and reduce backhaul traffic by storing popular video content closer to end users. Edge caching is particularly beneficial for viewport-aware streaming, which demands timely access to specific high-quality tiles that form the requested viewports. This process can be further optimized by aggregating overlapping user requests into virtual viewports (viewports that comprise the most popular tiles) [7]) to maximize cache efficiency. More specifically, when multiple users request overlapping spatial tiles within their viewports, the system can prefetch and cache these in-demand tiles.

In this work, we consider a heterogeneous wireless network serving users with diverse demands for videos and viewports comprised of Base Stations (BSs) equipped with caches that can store a limited number of 360° videos. Similar to our previous works [9], [7], we exploit scalable video coding combined with tile encoding. To optimize cache use, we propose a Deep Reinforcement Learning (DRL)-based adaptive caching framework for 360° video streaming based on the Advantage Actor-Critic architecture (A2C). Our proposed system decomposes individual user requests for 360° video into tile subrequests, that may be encoded at a different quality, using scalable coding to facilitate viewport-aware streaming. Specifically, the base layer is processed as a single subrequest, while the viewport tiles in the enhancement layer are processed as separate subrequests. The proposed approach learns a caching policy that dynamically decides whether a given subrequest item should be cached at the edge based on the current system state. Further, our framework incorporates Generalized Advantage Estimation (GAE) with A2C to stabilize learning in non-stationary caching environments. To summarize, the main contributions of this work are:

- We formulate a novel adaptive 360° video edge caching problem as a Markov Decision process (MDP), enabling the system to dynamically optimize cached tiles and replace-

This research was supported by Leverhulme Research Project Grant with grant number RPG-2024-420

ment policies under continuously evolving network states. We further design a novel mechanism considering the κ least popular cached contents for defining the state and action spaces, significantly reducing complexity while maintaining high efficiency in caching decisions;
- We introduce a novel request-splitting mechanism that decomposes user 360° video requests into discrete base-layer and enhancement-layer tile subrequests, which provides an extra level of granularity for dynamically deciding which tiles to cache and at what quality;
- We evaluate the proposed mechanism against its counterparts in terms of PSNR, cache hit ratio, and backhaul traffic reduction. Our A2C-based caching policy achieves a 10% to 20% improvement in cache hit ratio while increasing PSNR by approximately 2 dB.

II. Related Works

One of the main challenges in viewport-aware streaming is accurately predicting future viewports to enable proactive network caching. To this end, Liu *et al.* [10] proposed a caching scheme that jointly optimizes tile placement and viewport rendering, achieving lower latency over decoupled caching and rendering approaches. However, their solution relies on precomputed visual and behavioral features, failing to account for real-time changes in user attention. In contrast, Le *et al.* [11] proposed a viewport-dependent streaming framework that dynamically reduces the tiles' quality or omits tiles outside the user's focus area. These frameworks efficiently reduce bandwidth, but rely on static user-context information, and hence cannot be trivially extended to multi-user request environments, as the one we consider in this work.

In the context of prefetching management for 360° video streaming, Maniotis *et al.* [9] propose a DRL-based technique that maximizes PSNR through the prefetching and caching of tiles encoded in different quality levels by means of scalable video coding. This approach introduces the concept of virtual viewports to reduce dimensionality and employs a Deep Q-Network (DQN) to optimize cached content. A similar setting is considered by Gao *et al.* [8] where DQN is combined with an optimization model to reduce the search space and computational complexity. Although both approaches are highly effective, they still face scalability and computation challenges.

Li *et al.* [12] introduced a proactive tile-based caching framework that results in a substantial reduction in backhaul traffic, but it relies on static popularity distributions and cannot adapt to real-time variations in multi-user viewport demands. Furthermore, Zhong *et al.* [13] proposed another proactive caching framework that integrates K-nearest neighbors (KNN) into a DRL architecture to reduce the action-space dimensionality. However, these approaches fail to address reward sparsity and variance in highly dynamic environments, nor do they mitigate the state-space explosion caused by scaling multi-user viewport demands.

III. System Model

In this section, we present the proposed cache-enabled, tile-based 360° video streaming framework, designed to support users with diverse video and viewport requirements.

A. Content Representation

Let $\mathcal{V} = \{1, \ldots, V\}$ denote the set of available 360° videos. Each video $v \in \mathcal{V}$ is encoded into G Groups of Pictures (GOPs), where $g \in \mathcal{G} = \{1, \ldots, G\}$ is the g-th GOP. Each GOP is encoded into M tiles, where $m \in \mathcal{M} = \{1, \ldots, M\}$ denotes the m-th tile. In addition, each video is encoded using scalable video coding into L quality layers, where $l \in \mathcal{L} = \{1, \ldots, L\}$ represents the l-th quality layer. Further, let o_{vglm} represent the size (in Mbits) of the m-th tile in the l-th quality layer in GOP g of video v. The parameter o_{vglm} determines whether a streaming data unit can be prefetched into the cache. The complete set of streaming units is therefore defined as:

$$\mathcal{O} = \{o_{vglm} \mid v \in \mathcal{V},\ g \in \mathcal{G},\ l \in \mathcal{L},\ m \in \mathcal{M}\}. \quad (1)$$

B. User Request Model

We assume time is slotted into T discrete time slots. At each time slot $t \in \{1, \ldots, T\}$, a user $u \in \mathcal{U}$ requests a GOP g of video v. We denote the request sequence of user u as:

$$W_u = \{w_u^1, w_u^2, \ldots, w_u^T\}. \quad (2)$$

Each request w_u^t is decomposed into multiple subrequests. The first subrequest, denoted by $w_{u,0}^t$, requests all GOP tiles at the quality corresponding to the base layer to ensure a minimum viewing quality. The remaining k subrequests, $\{w_{u,1}^t, \ldots, w_{u,k}^t\}$, correspond to requests for enhancement layer tiles within the user's viewport. Note that viewports consist of a subset of k tiles forming a contiguous rectangular region of the 360° video.

C. Cache Model and Virtual Viewport

Fig. 1 illustrates the proposed cache architecture. When a request w_u^t arrives at the edge cache, it is partitioned into $k+1$ subrequests, where k is the number of tiles comprising the user's viewport. The system then queries the cache for requested tile layers, referred to as items. Upon a cache hit, any unmatched cached items are replaced with items corresponding to the newly predicted viewport of the same video. In contrast, when a cache miss occurs, the content is fetched from the origin server. In parallel, the mechanism evaluates whether the item should be prefetched for the subsequent time slot, $t+1$. This prefetching decision is governed by the Last Sample Replication (LSR) algorithm, which leverages virtual viewports to optimize overall cache performance.

Partitioning the main request into granular subrequests facilitates caching of individual tiles encoded at different quality layers. This abstraction permits capturing the popularity trends inherent in varied user viewing patterns. Nevertheless, the proposed cache abstraction relies on two structural assumptions: (i) although the base layer is segmented into tiles, the entire

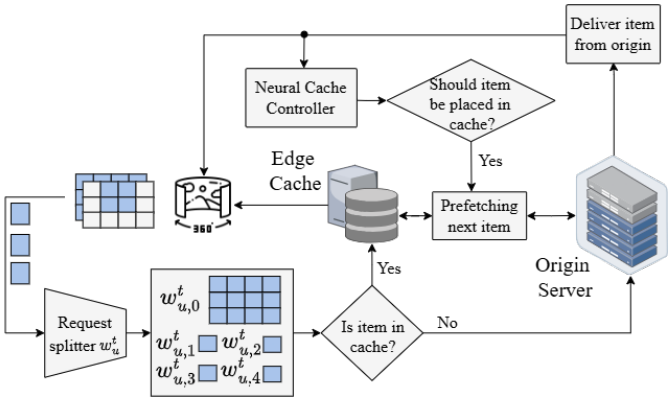

Fig. 1. Overview of the tile-based 360° video streaming and caching system.

base layer is fetched simultaneously via a single subrequest, to guarantee a minimum viewing quality, and (ii) the caching of an enhancement layer tile is strictly conditioned on the prior caching of all lower-quality layers associated with that tile, reflecting the constraints of scalable video coding.

The sequential nature of caching decisions under uncertainty motivates modeling the problem as an MDP. The MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ represents the state space, $\mathcal{A}$ denotes the action space, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor.

1) State Space: The state $s_t \in \mathcal{S}$ represents the system observation at time t, constructed from user request history over both short-term (H_s) and long-term (H_l) horizons. Since RL-based cache management suffers from the curse of dimensionality when large caches are considered [14], we define states only by the subset of cache entries relevant to the decision-making process. In particular, let $\mathcal{K}_t \subseteq \{1, \ldots, C\}$ denote the set of indices corresponding to the κ least popular cached items at time slot t. The state is defined based on this subset, as these cache entries are more likely to be replaced during cache updates. The state is represented by the tuple $s_t = (\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)$, described as follows:

- **Video popularity subset** ($\mathbf{x}_t$): This captures the request frequency of items indexed by $\mathcal{K}_t$. It is defined as $\mathbf{x}_t = [\mathbf{x}_t^s, \mathbf{x}_t^l]$, where $\mathbf{x}_t^f \in \mathbb{R}^\kappa$ for $f \in \{s, l\}$ represents the short- and long-term request frequencies of the items in $\mathcal{O}$, while preserving their ordering.
- **Candidate item features** ($\mathbf{y}_t$): This represents the request frequency of the current candidate item and is defined as $\mathbf{y}_t = [\mathbf{y}_t^s, \mathbf{y}_t^l] \in \mathbb{R}^2$.
- **Target subrequest indicator** ($\mathbf{z}_t$): This is a binary vector specifying the caching objective. It is defined as $\mathbf{z}_t \in \{0, 1\}^{k+1}$ with $k+1$ the total number of subrequests. Recall that k is the number of tiles forming a virtual viewport. For example, $\mathbf{z}_t = [0, 1, 0, 0, 0]$ indicates the current caching decision is for the second cached item.

2) Action Space: At each time step t, the agent selects an action $a_t \in \mathcal{A} = \{0, 1, \ldots, \kappa\}$ to either cache the next requested item or take no action. We define a time-dependent mapping function $f_t() : \{1, \ldots, \kappa\} \to \mathcal{K}_t$ that maps each non-zero action index to a physical cache slot, where $\mathcal{K}_t$ denotes the indices of the κ least popular cached items. The action determines the target cache index k_t as:

$$p_t = \begin{cases} \text{do nothing}, & \text{if } a_t = 0, \\ f(a_t), & \text{if } a_t \in \{1, \ldots, \kappa\}. \end{cases} \tag{3}$$

Accordingly, $a_t = 0$ leaves the cache unchanged, while $a_t > 0$ triggers the eviction of the item at physical position index p_t to accommodate the new candidate.

3) Reward Function: The reward is designed to reflect the perceived quality improvement achieved through caching. In particular, it is based on distortion reduction resulting from a cache hit. Let $\mathcal{Q}^{w_{u,0}^t}$ denote the distortion reduction associated with the base layer, and $\mathcal{Q}^{w_{n,i}^t}$ the distortion reduction after receiving the i-th enhanced tile (and all previous enhancement layers for that tile), where $i \in \{1, \ldots, k\}$. The one-step reward is defined as:

$$r(s_t, a_t) = \mathcal{Q}^{w_{u,0}^t} + \frac{1}{k} \sum_{i=1}^{k} \mathcal{Q}^{w_{u,i}^t} \tag{4}$$

IV. Actor-Critic Solver

The cache optimization objective is to maximize the expected cumulative discounted reward:

$$J(\theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{5}$$

In our setting, we employ an Actor-Critic framework that is well-suited to the studied problem. AC combines a parameterized policy (the Actor) with a value-function baseline (the Critic) to stabilize learning in non-stationary caching environments. By optimizing the expected cumulative reward, Actor-Critic is effective for sequential decision-making problems, leading to better long-term performance than traditional optimization methods.

A. Actor and Critic Networks

The Actor network approximates the stochastic policy $\pi_\theta(a_t|s_t)$, which outputs a probability distribution over the action space $\mathcal{A}$ given the current state $s_t \in \mathcal{S}$. Actions correspond to cache updates for either the video's base layer or its enhancement layer tiles. Formally, the actor is parameterized by θ, and maps states to action probabilities:

$$\pi_\theta(a_t|s_t) = \mathbb{P}(a_t|s_t; \theta) \tag{6}$$

The Critic network, parameterized by ϕ, estimates the value state function $V_\phi(s_t)$, which represents the expected long-term return starting from state s_t under the current policy:

$$V_\phi(s_t) = \mathbb{E}_{\pi_\theta} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,\middle|\, s_t \right] \tag{7}$$

B. Generalized Advantage Estimation

During the learning phase, the agent samples a contiguous trajectory of N transitions (bounded by a predefined rollout horizon) from the replay buffer. Preserving the temporal sequence of transitions (s_t, a_t, r_t, s_{t+1}) is strictly required to compute sequential targets accurately.

To balance the bias-variance tradeoff in advantage estimation, we employ Generalized Advantage Estimation (GAE) [15]. The temporal difference (TD) residual at time step t is defined as:

$$\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t) \tag{8}$$

The GAE advantage $\hat{\Psi}_t$ is computed as an exponentially weighted sum of future TD residuals:

$$\hat{\Psi}_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{9}$$

where $\lambda \in [0, 1]$ is a smoothing parameter that controls the bias-variance tradeoff. Setting $\lambda = 1$ corresponds to empirical multi-step advantage (equivalent to a Monte Carlo-like return with lower bias but higher variance), while $\lambda = 0$ results in one-step TD advantage (high bias, low variance). We then obtain the GAE target returns as $G_t = \hat{\Psi}_t + V_\phi(s_t)$.

C. Loss Functions and Parameter Updates

The Actor network parameters are updated via gradient ascent to maximize the expected return. To compute the Actor loss, we utilize the detached advantage estimates to scale the log-probabilities of the taken actions. The actor loss function minimized by the optimizer is given by:

$$\mathcal{L}_{actor}(\theta) = -\frac{1}{N} \sum_{t=1}^{N} \log \pi_\theta(a_t|s_t) \hat{\Psi}_t \tag{10}$$

The Critic network is trained to minimize the Mean Squared Error (MSE) between its state-value predictions and the computed GAE target returns G_t:

$$\mathcal{L}_{critic}(\phi) = \frac{1}{N} \sum_{t=1}^{N} \left(V_\phi(s_t) - G_t\right)^2 \tag{11}$$

To ensure stable optimization and prevent catastrophic parameter updates due to large policy gradients, we apply global gradient norm clipping to both the Actor and Critic networks before the parameter update step. The parameters are then updated simultaneously via backpropagation:

$$\theta \leftarrow \theta - \alpha_\theta \nabla_\theta \mathcal{L}_{actor}(\theta) \tag{12}$$

$$\phi \leftarrow \phi - \alpha_\phi \nabla_\phi \mathcal{L}_{critic}(\phi) \tag{13}$$

where α_θ and α_ϕ represent the learning rates for the actor and critic optimizers, respectively.

Algorithm 1 presents the training procedure for the proposed A2C caching framework. *Lines 1–2* show the start of the process by initializing the Actor (π_θ) and Critic (V_ϕ) networks, along with an empty rollout buffer. During each episode, the algorithm processes sequential sets of incoming requests

Algorithm 1 A2C Reinforcement Learning Training Loop

```
1:  Initialize Actor π_θ and Critic V_φ
2:  Initialize rollout buffer D ← ∅
3:  for episode e = 1 to E do
4:      Initialize the environment state s_0
5:      for step t = 0 to max_steps − 1 do
6:          Receive request set M_t = {m_1, m_2, ..., m_k}
7:          for each item m_k ∈ M_t do
8:              if item m_k is not cached then
9:                  Extract state s_t^k for request k
10:                 Sample and execute action a_t^k ~ π_θ(·|s_t^k)
11:             end if
12:         end for
13:         Observe global reward r_t and next state s_{t+1}
14:         for each item m_k ∈ M_t not cached do
15:             Store transition (s_t^k, a_t^k, r_t, s_{t+1}^k) in D
16:         end for
17:         if (t + 1) ≡ 0 (mod N_B) then
18:             Compute {δ_i} and {Ψ̂_i} Eq. (9)
19:             Compute ∇_θ L_actor and ∇_φ L_critic over D
20:             Update π_θ and V_φ
21:             Clear buffer D ← ∅
22:         end if
23:         s_t ← s_{t+1}
24:     end for
25: end for
```

($\mathcal{M}_t$). For each requested item that is not currently cached (*lines 8–12*), the agent extracts a localized state representation and samples a caching or replacement decision from the Actor's policy. Once all requisite actions are executed for the current timestep, the environment transitions to the next state and returns a global reward in *line 13*, which is appended to the rollout buffer alongside the individual state-action transitions (*lines 14-15*). To ensure stable and sample-efficient learning, the network parameters are updated periodically in batches of size $\mathcal{N}_B$. During this update phase, the agent computes GAE to evaluate the quality of its actions, calculates gradients for both the Actor and Critic losses over accumulated trajectories, updates the network weights, and clears the buffer for the subsequent rollout phase (*lines 17-22*).

V. Performance Evaluation

A. Simulation setup

In this section, we describe the setup of the proposed cache-enabled system for 360° video streaming. Each video is encoded into multiple Groups of Pictures (GOPs). Each video frame consists of 12 tiles arranged in a 3-by-4 grid. Furthermore, each individual tile is encoded by a scalable video encoder, such as SHEVC or SVVC, into a base layer and an enhancement layer without loss of generality. We assume that the distortion reduction achieved upon recovering the base layer is 30 dB, with the enhancement layer providing an additional reduction of 10 dB. The video catalog contains 500 videos, each consisting of 30 GOPs, with each GOP

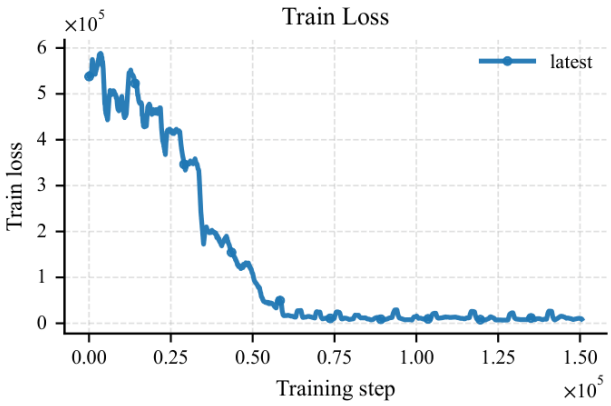

Fig. 2. Convergence of the loss function with respect to the training timesteps.

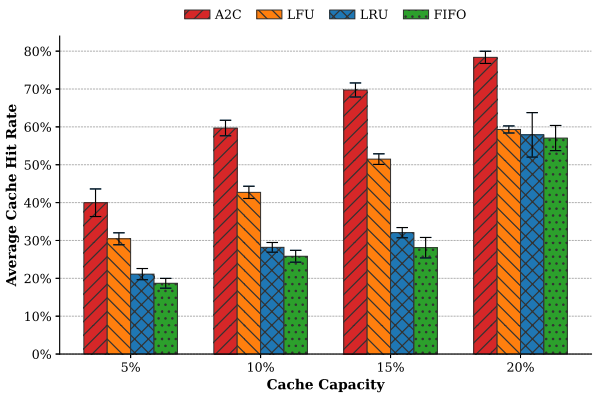

Fig. 3. Cache hit rate versus the cache capacity for the schemes under comparison.

lasting 1 second. The catalog is divided into 10 different video categories, e.g., sports, films, news, etc. To model user heterogeneity, video requests follow a Zipf distribution with parameter $\eta = 1$, while video session interruptions are modeled using a Weibull distribution [16]. During a viewing session, users may switch videos based on the shape parameter of the Weibull distribution for each category [16]. The cache size ranges between 5% to 20% of the video catalog. The system serves 200 users by a single Base Station, with each user's session lasting 30 seconds. We consider 10 distinct viewport trajectories [7] and 3,000 viewport requests. The short- and long-term horizons are set to 300 and 1000, respectively. Finally, the transmission delay for data served from the local cache is 1/6 Mbps, whereas retrieving 1 Mbit from the backhaul incurs a delay corresponding to 1/2 Mbps.

B. Hyperparameters and Training Phase

The input state dimension is determined by the reduced representation over the subset $\mathcal{K}_t$. Specifically, the state has a dimension $2\kappa + |\mathbf{y}_t| + |\mathbf{z}_t|$, which captures the short- and long-term popularity features of the κ selected cache items, the candidate-item frequency vector, and the subrequest indicator. The hidden layer has a size of 512, and the output dimension is $\kappa + 1$. The neural network parameters are optimized using the Adam optimizer. To ensure the Critic network learns faster than the Actor network, we set the Actor's learning rate to 0.001 and the Critic's to 0.0025. Training is performed with mini-batches of size 128 sampled at each update step. To capture long-term reward dependencies, the discount factor is set to $\gamma = 0.99$. The convergence of the loss function in our system for the baseline scenario is illustrated in Fig. 2. Similar convergence patterns are observed across different system configurations.

C. Baseline Algorithms

Our scheme is compared against three known heuristics:

- **Least Recently Used (LRU)**: In this scheme, when a cache miss occurs, the least recently used cached video is evicted. When a cache hit occurs, unmatched high-quality tiles in the cache are replaced with newly predicted viewport tiles.
- **Least Frequently Used (LFU)**: In this scheme, upon a cache miss, the least frequently used cached video is replaced. Upon a cache hit, unmatched high-quality tiles in the cache are replaced with newly predicted viewport tiles.
- **First-In-First-Out (FIFO)**: In this scheme, when a cache miss occurs, the earliest cached item is evicted. When a cache hit occurs, unmatched high-quality tiles in the cache are replaced with newly predicted viewport tiles.

VI. Experimental Results

We first evaluate the average cache hits under cache capacities ranging from 5% to 20% in Fig. 3. As observed, the proposed A2C policy consistently outperforms the baseline schemes across the entire range of cache capacities. At 10% cache capacity, A2C achieves a 40% improvement in average cache hits over the best baseline, LFU. Although the number of cache hits increases with cache capacity for all policies, it consistently maintains a clear advantage, as evidenced by the non-overlapping confidence intervals across multiple runs. Among the baseline schemes, LFU consistently outperforms both LRU and FIFO in all evaluated scenarios. As discussed in Section III-C1, by selecting a subset of the κ least popular cached items ($\mathcal{K}_t$), the A2C algorithm mitigates the curse of dimensionality typically associated with large-scale edge caches and takes full advantage of encoding in tiles and layers.

In Fig. 4, we investigate the achieved PSNR for different cache capacities. Consistent with the cache hit results, the proposed A2C-based scheme significantly outperforms all compared baselines under comparison for all evaluated cache sizes. Large gains are noticed when cache resources are scarce. For instance, at a 10% cache capacity, A2C yields a roughly 2 dB improvement in average PSNR over the strongest performing baseline, LFU. While PSNR increases for all schemes with increased cache capacity, A2C maintains a clear performance advantage (approximately 2 dB), supported by tight, non-overlapping variance intervals across the evaluations. Among the traditional heuristics, LFU strictly outperforms both LRU and FIFO for all scenarios. In addition, the performance gap among LFU, LRU, and FIFO gradually decreases as the cache capacity grows, eventually converging to similar values at a

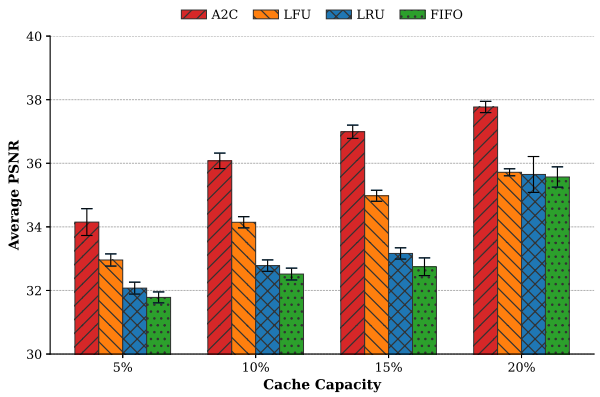

Fig. 4. PSNR of the rendered viewports with respect to the cache size for all the schemes under comparison.

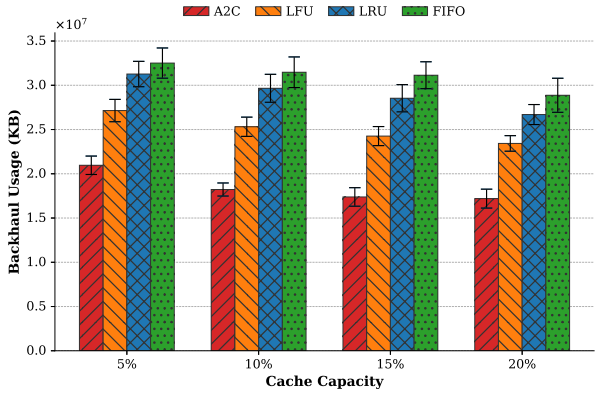

Fig. 5. Backhaul usage with respect to the cache size for all schemes under comparison

20% cache capacity. This superior behavior is driven by our architectural design as the reduced subset $\mathcal{K}_t$ reduces the state space dimensionality and enables A2C to fully exploit the underlying tile- and layer-based video encoding. As a result, the actor–critic networks can evaluate caching decisions with reduced variance and converge more efficiently toward an effective policy.

In Fig. 5, we evaluate the performance of the proposed system in terms of backhaul data usage. Our experiments demonstrate that the proposed A2C-based scheme satisfies user requests while consuming significantly less backhaul traffic than the compared baselines. As expected, increasing the cache capacity reduces backhaul load across all settings, since larger caches accommodate more video content and can serve more users locally. Furthermore, we observe that as the cache capacity scales from 5% to 20%, the performance gap between the schemes narrows. This happens as the increased cache allows all methods to serve a larger portion of the content locally, though our framework consistently maintains the lowest backhaul traffic load.

VII. Conclusion

In this paper, we proposed a novel DRL-based edge caching framework that adapts content placement under dynamic and diverse requests for 360° video content by tracking both short- and long-term popularity. We show that the curse of dimensionality in tile-based streaming can be effectively alleviated by a compact state space representation restricted to the κ least popular items. This policy allows preserving only critical demand characteristics that facilitate rapid policy convergence. Using an A2C architecture with GAE, the proposed method incorporates contextual conditioning to effectively handle stochastic overlapping user requests at the edge. Simulation results demonstrate significant improvements over the schemes under comparison in terms of cache hit ratio, PSNR, and backhaul traffic reduction. Future work will extend this framework to multi-base-station scenarios to investigate cooperative caching under heterogeneous network conditions and dynamic and diverse user demands.

References

[1] J. Chakareski, M. Khan, and M. Yuksel, "Toward enabling next-generation societal virtual reality applications for virtual human teleportation: A novel future system concept and computation-communication-signal representation trade-offs," *IEEE Signal Process. Mag.*, vol. 39, no. 5, pp. 22–41, 2022.

[2] X. S. Shen, J. Gao, M. Li, C. Zhou, S. Hu, M. He, and W. Zhuang, "Toward immersive communications in 6G," *Frontiers in Computer Science*, vol. 4, 2022.

[3] E. Bastug, M. Bennis, M. Medard, and M. Debbah, "Toward interconnected virtual reality: Opportunities, challenges, and enablers," *IEEE Comm. Mag.*, vol. 55, no. 6, pp. 110–117, 2017.

[4] R. Shafi, W. Shuai, and M. U. Younus, "360-degree video streaming: A survey of the state of the art," *Symmetry*, vol. 12, no. 9, 2020.

[5] W.-C. Lo, C.-L. Fan, J. Lee, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "360° video viewing dataset in head-mounted virtual reality," in *Proc. of the 8th ACM Multimedia Syst. Conf.*, MMSys '17, pp. 211–216, 2017.

[6] L. De Cicco, S. Mascolo, V. Palmisano, and G. Ribezzo, "Reducing the network bandwidth requirements for 360° immersive video streaming," *Internet Technol. Lett.*, vol. 2, no. 4, p. e118, 2019.

[7] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360° videos in heterogeneous networks," *IEEE Trans. Multimedia*, vol. 22, no. 9, pp. 2382–2395, 2020.

[8] C. Gao and T. Braun, "Dual-engine intelligent caching: A joint optimization framework for 360° mobile VR video edge caching," *IEEE J. Sel. Areas in Commun.*, vol. 43, no. 10, pp. 3532–3547, 2025.

[9] P. Maniotis and N. Thomos, "Viewport-aware deep reinforcement learning approach for 360° video caching," *IEEE Trans. Multimedia*, vol. 24, pp. 386–399, 2022.

[10] Y. Liu, J. Liu, A. Argyriou, L. Wang, and Z. Xu, "Rendering-aware VR video caching over multi-cell MEC networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2728–2742, 2021.

[11] T. T. Le, J.-B. Jeong, S. Lee, J. Kim, and E.-S. Ryu, "An efficient viewport-dependent 360 VR system based on adaptive tiled streaming," *Comput., Mater. and Continua*, vol. 66, no. 3, pp. 2627–2643, 2020.

[12] C. Li, T. Ye, T. Zong, L. Sun, H. Cao, and Y. Liu, "Coffee: Cost-effective edge caching for live 360 degree video streaming," *Computer Networks*, vol. 269, p. 111461, 2025.

[13] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proc. of 52nd Annual Conf. on Inf. Sciences and Syst. (CISS)*, pp. 1–6, 2018.

[14] H. Kim, T.-J. Sun, and E.-N. Huh, "T-CacheNet: Transformer-based deep reinforcement learning for next-generation internet content caching," in *Proc. of the 13th Int. Conf. on Netw., Commun. and Comput.*, ICNCC '24, pp. 9–16, 2024.

[15] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proc. of the Int. Conf. on Learn. Representations (ICLR)*, 2016.

[16] E. Baccour, A. Erbad, A. Mohamed, K. Bilal, and M. Guizani, "Proactive video chunks caching and processing for latency and cost minimization in edge networks," in *Proc. of IEEE Wireless Commun. and Netw. Conf. (WCNC)*, pp. 1–7, 2019.