Tile-based edge caching for 360^o live video streaming

Pantelis Maniotis, *Student Member, IEEE* and Nikolaos Thomos, *Senior Member, IEEE*

***Abstract*—360^o video is becoming an increasingly popular technology on commercial social platforms and vital part of emerging Virtual Reality/Augmented Reality (VR/AR) applications. However, the delivery of 360^o video content in mobile networks is challenging because of its size. The encoding of 360^o video into multiple quality layers and tiles and edge cache-assisted video delivery have been proposed as a remedy to the excess bandwidth requirements of 360^o video delivery systems. Existing works using the above tools have shown promising performance for Video-on-Demand (VoD) 360^o delivery, but they cannot be straightforwardly extended in a live-streaming setup. Motivated by the above, we study edge cache-assisted 360^o live video streaming to increase the overall quality of the delivered 360^o videos to users and reduce the service cost. We employ Long Short-Term Memory (LSTM) networks to forecast the evolution of the content requests and prefetch content to caches. To further enhance the delivered video quality, users located in the overlap of the coverage areas of multiple Small Base Stations (SBSs) are allowed to receive data from any of these SBSs. We evaluate and compare the performance of our algorithm with Least Frequently Used (LFU), Least Recently Used (LRU), and First In First Out (FIFO) algorithms. The results show the superiority of the proposed approach in terms of delivered video quality, cache-hit-ratio and backhaul link usage.**

***Index Terms*—Tile-encoding, 360^o live video streaming, edge-caching, LSTM networks.**

I. Introduction

In recent years, we have witnessed the emergence of live streaming platforms such as YouTube Live, Facebook Live, Twitch, Periscope, and Meerkat [1]. This has been fostered by the proliferation of mobile devices that provide access to the Internet from “everywhere”. These platforms allow attendants of live events, e.g., games, shows, conferences, etc. to capture them using affordable 360^o cameras, and then broadcast them to potentially millions of viewers.

To provide users an immersive experience, 360^o videos may be watched with the help of Head-Mounted Displays (HMDs), e.g., Oculus Rift, Samsung Gear VR, and HTC Vive [2]. In HMDs, each 360^o scene is projected in the internal part of a spherical surface [3], and a user wearing an HMD watches only a Field of View (FoV) of the spherical scene, known as viewport. The projected viewport is controlled by the orientation of the viewer’s head in the x, y, and z axes, as shown in Fig. 1.

To prevent users from experiencing motion sickness, the mobile network’s response to the users’ head movements should be as fast as the HMD refresh rate [4]. Considering that the refresh rate is commonly 120Hz, the mobile network should deliver the requested viewport to the users in less than 10msec. The end-to-end delivery delay imposes a tight constraint that challenges mobile networks to respond to such demands by tracking and rendering only the requested viewport. Transmitting the whole scene could help overcome the above limitation, as there would be no need for real-time response to head movements. However, this is not an efficient strategy as it requires significant bandwidth resources due to the high resolution of 360^o videos (e.g., 4K, 8K, or higher) [5]. In addition, delivering the entire scene leads to significant bandwidth waste since only a part of the 360^o video will be displayed. The problem deteriorates when the network infrastructure must support a large number of users who may want to access the requested content from various locations.

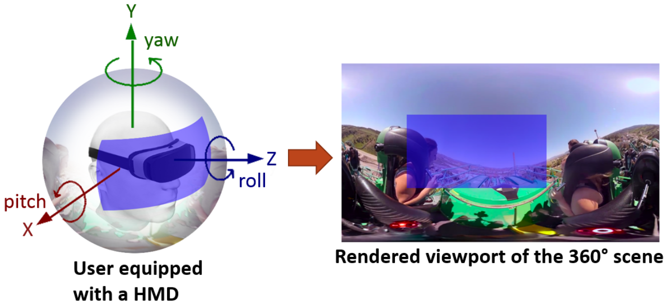

Fig. 1. User equipped with a Head Mounted Display, and the corresponding video scene.

Tile-based streaming has been proposed in [3], [6], [7] to reduce bandwidth requirements for delivering 360^o videos. In tile-based video streaming systems, the 360^o videos are encoded into independently encoded segments (see Fig. 2), known as tiles. Further, tiles can be encoded into multiple quality layers. The base layer which is the most important layer offers a reconstruction of a tile at the lowest available quality. The next layers (enhancement layers) contain information that can progressively improve each tile’s reconstruction quality. In order to reconstruct a tile at a requested quality (e.g., lth quality layer), the corresponding enhancement layer (lth enhancement layer) should be available along with all previous enhancement layers, including the base layer. Tile encoding facilitates streaming of only the requested viewport in high quality to a user. To this aim, the demanded viewport is predicted by considering past users’ navigation patterns. This allows prefetching of the tiles corresponding to the forecasted viewport to the user in order to meet the delivery deadlines. The rest of the scene is still transmitted to the user, but at a lower quality. This is needed to prevent rendering of black

P. Maniotis and N. Thomos are with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom (e-mail: {p.maniotis, nthomos}@essex.ac.uk).

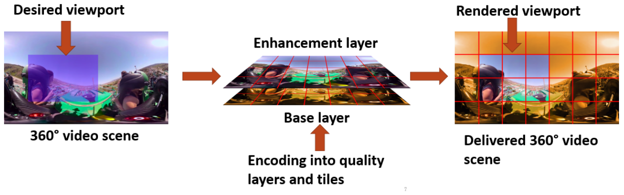

Fig. 2. Encoding of 360^o videos into multiple quality layers and tiles to facilitate rendering of the demanded viewport.

areas (i.e., areas with no content) in case of an erroneous estimation of the demanded viewport [8].

When multiple users request various 360^o videos, streaming the requested content independently to the users leads to waste of valuable bandwidth in the core network and the backhaul links. To avoid such inefficient use of bandwidth resources, edge caching has shown to be an efficient method to accommodate the demands of multiple users in cellular networks. In edge caching systems, Small Base Stations (SBSs), e.g., pico-cells and femto-cells, are equipped with a cache where they can store popular content files [9], [10]. The use of caches permits to serve the content from the SBSs and limits the need to retrieve it from distance servers through backhaul links. Content retrieval from caches also reduces the experienced latency. However, existing edge caching systems [11]–[13] that exploit 360^o video properties are appropriate for Video on Demand (VoD) systems, and they cannot be trivially used for 360^o live video streaming. Specifically, the works in [11]–[13] belong to the family of offline caching schemes and require the knowledge of the content popularity distribution.

In this paper, we propose a novel online caching framework to support 360^o live video streaming building on our early work [14]. Similarly to [11]–[13], we exploit encoding of the 360^o videos into multiple quality layers and tiles. To the best of our knowledge, this is the first sophisticated caching scheme that can support live streaming of 360^o videos. The proposed system aims at maximizing the overall video quality rendered by the users, and decrease the usage of the backhaul links. To this aim, it uses limited past observations of the users' requests for the various 360^o videos and viewports in order to determine the optimal cache eviction/placement strategy. When a decision is made by our system to cache a 360^o video, all the tiles of that video encoded in base quality are cached to ensure interactivity, and a number of tiles in high quality are also cached to enhance the quality of the demanded viewport. The number of cached tiles in high quality at the SBSs depends on both videos' popularity, i.e., for the most popular videos, more tiles are cached in high quality, as well as the tiles' popularity, which determines which tiles are most likely to be requested.

Our system uses Long Short-Term Memory (LSTM) networks, which are efficient for time series forecasting [15], to decide which tiles of the 360^o videos should be cached at the edge caches and in what quality. LSTM networks have been previously used for predicting video popularity [16], [17] of traditional videos in cache networks, but they have not been used for optimizing the cached content in cache networks for the case of 360^o videos we examine in this paper. Through the use of the LSTM networks, the popularity of the 360^o videos and tiles for the next Group of Pictures (GOP) is predicted with only a small error. The prediction outcomes are then used to determine the content that should be prefetched at the SBSs caches to enable on-time delivery of the 360^o video content to the users. We use both real and synthetic 360^o video traces to evaluate the performance of our scheme. We compare our solution with Least Frequently Used (LFU), Least Recently Used (LRU), and First In First Out (FIFO) algorithms, which are commonly used to support live streaming applications. The results show that the proposed method offers large gains in terms of the quality of the rendered video, the cache hit ratio, and the service cost. When users are located in the communication range of multiple SBSs, we associate users with multiple SBSs to further enhance the performance of all methods. Hence, users can access content stored in multiple caches from where they can satisfy their requests.

In summary, the contributions of our work can be summarized as follows:

- the introduction of a novel online caching framework to support 360^o live video streaming. Our framework takes into account both 360^o video popularity and viewports' popularity to decide which tiles to cache at the SBSs;
- the use of LSTM networks for optimizing cache decisions for 360^o videos in cache networks for the live streaming case. This permits prefetching of content that is likely to be popular in the future at the SBSs caches, and hence, facilitates the on-time delivery of 360^o video content to the users;
- extensive evaluation of the proposed solution with respect to various system parameters and comparisons with several state-of-the-art schemes in order to showcase the benefits coming from our LSTM empowered cache-assisted framework.

The rest of this paper is organized as follows. In Section II, we overview works related to 360^o videos, 360^o live video streaming, and edge caching. Then, in Section III, we describe the system setup, and afterwards, we provide the system model in Section IV. Next, in Section V, we evaluate the performance of the proposed scheme. Finally, conclusions are drawn in Section VI.

II. Related Work

During the past decade, the 360^o video attracted significant attention from the academic community. Different aspects of 360^o video have been studied, and systems for processing and streaming such content have been presented. For example, authors in [18] designed subjective experiments in order to analyze users' navigation patterns when viewing 360^o videos. They observed that for a short time period in the beginning of the 360^o video streaming, e.g., first 30 seconds, users tend to look around, and their attention is very low. However, after about 1 minute, users' attention becomes more focused. Authors in [19] propose a solution based on MPEG DASH in order to cope with the bandwidth limitations when streaming 360^o videos at a resolution of 16K with an FoV of 4K. A

Hadoop/Spark based transcoding solution is presented in [20] to facilitate the delivery of high resolution 360^o videos, i.e., 8K and above. The evaluation shows that real-time transcoding of 360^o videos with a resolution of 8K, split into 8x8 tiles, can be achieved at a rate of 99 fps. Differently from [20], in [21], it is investigated how different parts of a 360^o video can be encoded at different qualities. To maximize the quality of the 360^o videos displayed by the users, authors in [22] optimize the video encoding configurations so that 360^o videos are optimally encoded into multiple representations. To predict the demanded viewports by the users in the near future, authors in [23] present a contextual bandit algorithm. Differently from [23], authors in [24] propose a trajectory-based viewport prediction algorithm, aiming to predict the users' requested viewports in the long-term. To deal with the problem of inaccurate distortion measurements occurring when 360^o videos are projected from the spherical domain to 2D plane, the work in [25] proposes a method that optimizes the encoding process based on signals' distortion in spherical domain. The results show that their method achieves significant coding gains in each projection and results in large bit rate savings.

Tile-encoding has been exploited for 360^o live video streaming [26] in order to facilitate the delivery in high quality of only the requested FoV. The presented live streaming platform is supported by an architecture based on RTP and DASH. The results made apparent the trade-off between video quality and bandwidth usage and showed bandwidth savings of about 50%. Differently from [26], authors in [27] proposed a multicasting system to support 360^o video live streaming. Simulations have shown that users with low bandwidth capacities experienced a quality gain of 3 dB, while users with high bandwidth capacities saw a gain of 3.5-4 dBs. A live streaming system based on MPEG media transport (MMT) is proposed in [28] to guarantee the delivery of high-quality 360^o videos to the users. A generic measurement system is presented in [29] for collection of key performance statistics, e.g., video quality change, rebuffering events, that is used for evaluating the performance of live streaming platforms for 360^o videos. Authors in [2] proposed a mobile video telephony system over LTE cellular networks that dynamically adjusts the compression strategy in order to provide a high Quality of Experience (QoE) to the users.

Edge caching has been proposed as an enabling technology to support streaming of both standard videos [17], [30], and 360^o videos [11]–[13], [31], [32]. In edge caching architectures, SBSs cache popular content, from where users can retrieve the requested content directly. Caching is beneficial in terms of reducing the usage of the pricey backhaul links [10], [33], and helps users enjoy services with less latency. The advantages of using edge caches to store 360^o videos are demonstrated in [11], [12]. The decisions regarding where to cache a content and from where to retrieve it are made jointly on a per-tile and per-layer basis. Differently from [11], [12], authors in [13] examine 360^o video caching when tiles are encoded at different resolutions, and in multiple layers. A FoV-aware edge caching scheme is proposed in [31] which uses a probabilistic model based on the viewing trajectories of past requests. Differently from [31], authors in [32] propose a collaborative FoV prediction edge caching scheme by assigning different playback latencies to different users. The difference in the playback latencies creates the concept of "streaming flock", where the statistics regarding which viewports users are watching in the front of the flock are exploited to predict the viewports that users will watch at the back of the flock (at a later time instant). The above-mentioned methods [31], [32] assume users arriving at different time instants with the time between users' arrivals often exceeding 20 secs, which render them inappropriate to support live streaming with strict requirements

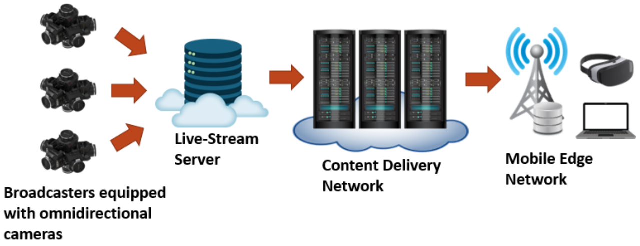

Fig. 3. Considered live streaming architecture.

III. System Setup

In this section, we first introduce the considered live streaming architecture. Then, we discuss transcoding of 360^o videos that is needed for encoding the videos into multiple quality layers and tiles, when this had not been done by the content producers. Finally, we describe the considered mobile edge network architecture, the users' requests model, and the end-to-end delay in mobile edge networks.

1) Live Streaming Architecture: A high-level representation of the considered 360^o video live streaming architecture is depicted in Fig. 3. We assume multiple users (broadcasters) that capture a 360^o FoV of a scene using omnidirectional cameras. The captured 360^o videos are first transmitted to the Live Stream server using the Real-Time Messaging Protocol (RTMP) [34]. RTMP is selected as it can ensure low end-to-end latency between the broadcaster and viewers. The Live Stream server transcodes the 360^o videos so that they consist of multiple quality layers and tiles. Transcoding is needed because the captured 360^o videos from the broadcasters may not necessarily be encoded in that format. The transcoded video streams are transmitted to the Content Delivery Network (CDN) using HTTP. The mobile edge servers are populated with content from the CDN according to the proposed cache optimization algorithm, which will be presented in Section IV. We would like to note that this work focuses on the cache optimization of the mobile edge caches.

2) Transcoding of 360^o videos: We assume that the total number of the captured 360^o videos is $V = |\mathcal{V}|$, where $\mathcal{V} = \{1, \dots v, \dots, V\}$ stands for the set of 360^o videos that comprise the content library. The video transcoding at the Live Stream servers is performed using H.265/HEVC, but our scheme is compliant with other video codecs. Each video stream is encoded into a number of L quality layers and M

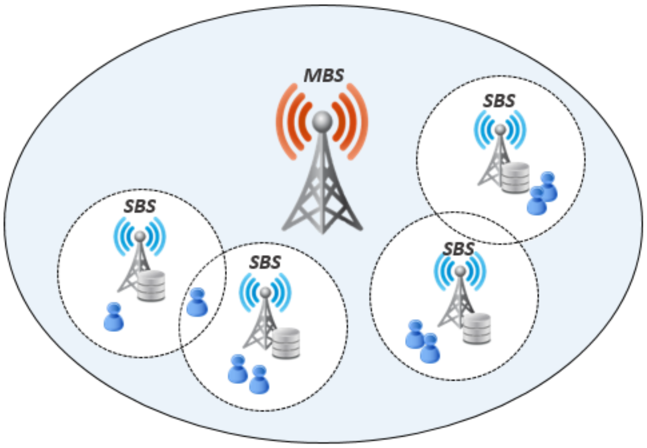

Fig. 4. Considered mobile-edge architecture.

tiles. The first quality layer is the base layer, while the rest $L-1$ layers are known as enhancement layers. The acquisition of a tile at the base quality layer offers a reconstruction of the tile at the lowest quality, while the progressive acquisition of up to the lth quality layer gradually enhances the quality of that tile. As the duration of each video may vary, each video consists of a number of GOPs, that their number depends on the duration of each video.

3) Content Delivery Network: The CDN comprises geographically distributed servers that collaboratively cache and distribute popular content to a global reach using high-speed links. Each CDN server can cache a number of video files. In a CDN, the requested content in a geographic area is commonly served from the cache of the CDN server that is closer to the users. In this way, the demanded content is retrieved by the users with less latency, as their requests do not have to be routed to the Live Stream server. The use of CDNs significantly reduces the traffic reaching the Live Stream server.

4) Mobile Edge Network: We consider a mobile edge network architecture as the one depicted in Fig. 4. This network consists of N Small Base Stations (SBSs), i.e., microcells, and a Macrocell Base Station (MBS). Let $\mathcal{N} = \{1, \dots n, \dots, N\}$ be the set of the N SBSs, and $N+1$ represent the MBS. The MBS is connected with the CDN through a high capacity backhaul link, i.e., optical fiber, while the connection of the SBSs with the CDN is established through the MBS by millimeter-wave links. For notational convenience, let the augmented set $\mathcal{N}_B = \mathcal{N} \cup N+1$ denote the set that includes all the SBSs along with the MBS. The communication ranges of the SBSs are represented by the set $\mathcal{P} = \{p_1, \dots p_n, \dots, p_N\}$, with p_n being the communication range of the nth SBS. We denote the communication range of the MBS by p_{N+1}. All SBSs are within range of the MBS; otherwise, they would not have access to the backhaul of the MBS. The cache capacity of the $n \in \mathcal{N}$ SBS is denoted by $C_n \geq 0$, $\forall n \in \mathcal{N}$.

5) Users: In the considered network architecture, we consider U users that form the set $\mathcal{U} = \{1, \dots u, \dots, U\}$. Users may be located in the overlap of the coverage areas of multiple SBSs, as shown in Fig. 4. When this happens, users may be associated with any of these SBSs. For each user, the primary SBS is the one that has the maximum signal-to-interference-plus-noise ratio (SINR). The association of a user with multiple SBSs enables requested tiles that are not cached at the primary SBS but are stored in the cache of one (or more) of the other SBSs the user resides to be delivered to the user from these caches. We assume that time is slotted in T time slots. In each time slot $t \in \mathcal{T}$, the request of user $u \in \mathcal{U}$ for a GOP of a 360^o video $v \in \mathcal{V}$ is denoted as w_u^t. Let $W_u = \{w_u^1, \dots, w_u^t, \dots, w_u^T\}$ be the set which contains T consecutive requests from user $u \in \mathcal{U}$. Hence, each request is indexed by the time slot t, which corresponds to the currently displayed GOPs. This removes the need to associate requests with the GOP index of each video.

6) End-to-end delivery constraint: The end-to-end delay captures the overall delivery delay a user experiences. It is the time elapsed from when users request the data until the data is delivered to them. This delay depends on the location of the SBS from where the requested content is retrieved. Let d_n be the delay needed to transmit one Mbit from the cache of the nth SBS to a user within the SBS communication range. Similarly, let d_{N+1} be the delay needed to transmit one Mbit that is fetched from the backhaul of the MBS to a user. In order to guarantee the timely delivery of the tiles of each GOP to the users, the following constraint should be met:

$$\sum_{n \in \mathcal{N}_B} \sum_{l \in \mathcal{L}} \sum_{m \in \mathcal{M}} o_{vglm} \cdot d_n \cdot y_{vglm}^n \leq t_{disp}, \forall v \in \mathcal{V}, g \in \mathcal{G} \quad (1)$$

The parameter o_{vglm} in (1) denotes the size (in Mbits) of the mth tile of the lth quality layer of the gth GOP of the vth 360^o video. The variable y_{vglm}^n takes the value 1 when the the mth tile of the lth quality layer of the gth GOP of the vth 360^o video is delivered on time to the uth user from the nth SBS ($n \in \mathcal{N}$) or the MBS ($n = N+1$), and 0 otherwise. The parameter t_{disp} corresponds to the time needed for each GOP to be displayed.

IV. System Model

1) Caching Entity (CE): We consider that each SBS is equipped with a CE, as shown in Fig. 5. This entity is responsible for deciding which 360^o videos and tiles should be cached at each SBS. Each caching entity is composed of a number of modules, e.g., User Requests Processor, User Requests Forecasting, Feature Updater, etc., that their operation will be discussed later.

2) Users Request Processor (URP): The URP module is responsible for decomposing the user requests $w_u^t, t \in \mathcal{T}, u \in \mathcal{U}$ as follows. If a viewport consists of k tiles, each user request w_u^t is decomposed into $k+1$ requests $w_{u,i}^t$, as shown in Fig. 6. The first request $w_{u,0}^t$ is for receiving all the tiles of the requested 360^o video at the base quality. The rest k requests $\{w_{u,1}^t, \dots, w_{u,i}^t, \dots, w_{u,k}^t\}$ are for receiving each of the k tiles of the requested viewport in high quality. This decomposition provides to our system the flexibility to decide which 360^o videos (all tiles at the base quality) should be cached to ensure interactivity, and which tiles of these videos should be cached in high quality to enhance the quality of the rendered video.

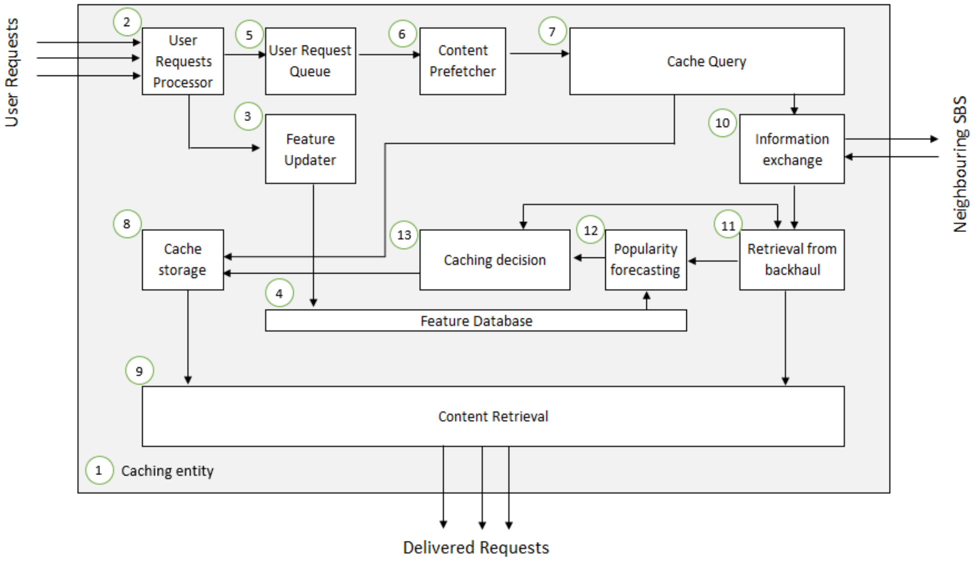

Fig. 5. Flow of operations in a caching entity.

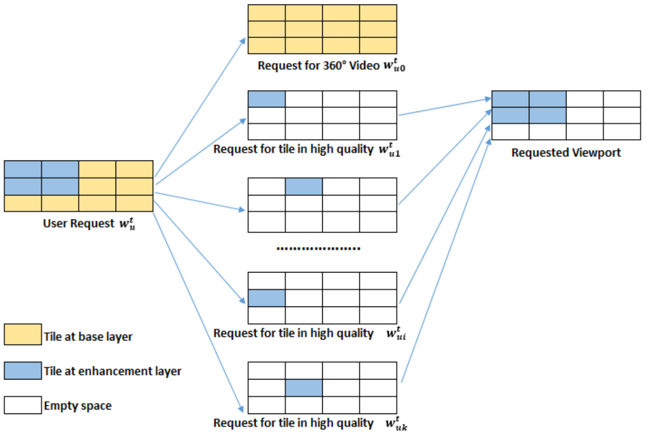
Request for 360° Video w^t_{u0}
Request for tile in high quality w^t_{u1}
Requested Viewport
User Request w^t_u
....
Request for tile in high quality w^t_{ui}
Request for tile in high quality w^t_{uk}
Tile at base layer
Tile at enhancement layer
Empty space

Fig. 6. Decomposition of user request w^t_u into $k+1$ requests.

Let the set $\mathcal{W}^t_u = \{w^t_{u,0}, w^t_{u,1}, \ldots, w^t_{u,i}, \ldots, w^t_{u,k}\}$ describe these $k+1$ requests.

3) Feature Updater (FU): The FU module is responsible for updating features related to the requests for the various 360^o videos and tiles. Specifically, this module calculates in each time slot $t \in \mathcal{T}$, the number of times each request (for receiving either a 360^o video at the base quality or a tile of a viewport in high quality) was encountered. The computed features are transferred to the Feature Database module, where this information is stored.

4) Feature Database (FD): The FD module stores the features computed by the FU module regarding the number of times the various 360^o videos (all tiles at base quality) and tiles (in high quality) were requested at an SBS.

5) Users' Requests Queue (URQ): After the decomposition of each user request into multiple requests by the URP module, the decomposed requests are directed to the URQ module. This module applies a technique called request coalescing [35]. According to this technique, when multiple requests for the same content arrive simultaneously at an SBS, the first request is prioritized for processing, while the rest of the user requests are held in a queue for later processing. This mechanism is needed because many people are watching the same content almost simultaneously in live streaming scenarios. Without such mechanism, in case a requested content is not cached at the SBS, a cache miss will occur for all the user requests for that content. This would cause all the traffic related to that content to be redirected to the origin CDN or even the Live Stream server, which in turn could cause the crashing of these servers.

6) Content Prefetcher (CP): Due to the end-to-end delay, SBSs cannot respond instantly to the users' head movements and transmit the demanded tiles by the users. This is avoided by using the CP module. Specifically, when the CP module receives requests $\mathcal{W}^t_u$ for the various 360^o videos (in base quality) and tiles (in high quality) from the URP at the time slot t, it decides what content should be prefetched for the various 360^o videos and tiles for the next time slot $t+1$. Let the set $\mathcal{Z}^{t+1}_u = \{z^{t+1}_{u,0}, z^{t+1}_{u,1}, \ldots, z^{t+1}_{u,i}, \ldots, z^{t+1}_{u,k}\}$ denote the content that will be prefetched to the users regarding the time slot $t+1$. Enabling prefetching allows the timely delivery of the content to the users. To decide which content should be prefetched, we use the Last Sample Replication (LSR) [8] algorithm. This algorithm was chosen because it has minimal computational overhead and hence is appropriate for real-time applications as the one we examine here. The use of a more advanced prediction algorithm would improve the delivered quality, however, it would not change the derived conclusions regarding the ability of the LSTM network to predict content popularity evolution. According to LSR, when the CP module receives a user request $w^t_{u,i}$ at time slot t, the content $z^{t+1}_{u,i}$ that is decided to be prefetched for the time slot $t+1$ is considered to be the same with the request $w^t_{u,i}$. For the sake of simplicity, we assume that for the first time slot, the content that will be prefetched to the users is the requested content.

7) Cache Query (CQ): The CQ module examines whether the content indicated by the CP module is already cached at the SBS. When this content is cached at the SBS, it is served to the users locally from the Cache Storage (CS) of the SBS. Differently, when the content indicated by the CP module is not available at the SBS, the Information Exchange (IE) module is activated to check whether a neighboring SBS can serve it to the user. This happens when the user is within the communication range of the neighboring SBS. In case the content is cached at a neighboring SBS, and the user is associated with that SBS, the content is delivered to the user from that SBS. Finally, when the content indicated by the CP module is not cached at any neighboring SBS the user is associated with, the content is fetched at the SBS through the backhaul of the MBS, from where it is served to the user.

8) Cache Storage (CS): The CS module is responsible for storing the 360^o videos at the base quality and the tiles of the cached videos in high quality. Each SBS has a separate cache storage module with capacity C_n, where n is the SBS node index.

9) Content Retrieval (CR): The CR module is responsible for the delivery of the content to the users. This module retrieves the content either through: a) the CS module, when it is cached at the SBS, or b) the Retrieval from Backhaul (RFB) module that retrieves the content from the backhaul, when it is not cached at the SBS.

10) Information Exchange (IE): The communication of an SBS with its neighboring SBSs is done through the IE module. In particular, in case of a cache miss for a content at an SBS, the SBS checks whether that content may be served to the user from a neighboring SBS. The communication between the SBSs is accomplished by millimeter-wave links through the MBS. The delay needed for the above communication is captured by the delay parameter d_n. Recall, that this parameter denotes the delay needed to deliver one Mbit from the nth SBS to a user.

11) Retrieval from Backhaul (RFB): The RFB module is responsible for retrieving the content that will be prefetched to caches of the SBSs through the backhaul. After retrieving the content by the RFB module, its popularity is estimated by the Popularity Forecasting module. Also, a decision is made at the Caching Decision module about whether to cache that content.

12) Popularity Forecasting (PF): The PF module at each time t performs forecasting of the popularity of the content that will be prefetched for the time slot $t+1$. Specifically, the PF module uses a window of h time slots in order to estimate popularity trends of the content that will be prefetched to the users, and cache at the SBSs content that will be popular. To this aim, it uses the features (number of requests per video and per tile) stored at the FD module regarding the previous $h-1$ time slots along with the current time slot $t \in \mathcal{T}$. Let us denote by $\lambda^t_{n,v,0}$ the popularity of the 360^o video $v \in \mathcal{V}$ (base quality) at the SBS $n \in \mathcal{N}$ in the time slot $t \in \mathcal{T}$. Similarly, let $\lambda^t_{n,v,m}$ stands for the popularity of the tile $m \in \mathcal{M}$ of the video $v \in \mathcal{V}$ at the SBS $n \in \mathcal{N}$, where $\mathcal{M} = \{1, \dots, m, \dots M\}$. The popularities $\lambda^t_{n,v,0}, \lambda^t_{n,v,m}$ are defined as the number of times each request (for either a 360^o video at the base quality, or a tile of a viewport in high quality) was encountered. The popularity features were not normalized, as the total number of requests is not known in advance. However, had the normalization of the popularity features in the range $[0,1]$ been possible, it would not affect the performance.

One way to predict these popularities would be to use Recurrent Neural Networks (RNNs), as they are effective for time series data forecasting [36]. However, simple RNNs cannot capture long-term dependencies, as they lack control structures, which causes the norm of gradients to decay or explode during training [37]. To overcome this problem, LSTM networks have been proposed [38], which are a special type of RNNs able to learn long-term dependencies. Inspired by [16], we use an LSTM network for the forecasting of the popularity of the content retrieved by the RFB module. The LSTM network takes as input the features of a content regarding the previous $h-1$ time slots along with the current time slot t, and outputs the estimated popularity for the content for the time slot $t+1$. The LSTM is initially pre-trained offline (warm-up phase) with historic data profiles using the backpropagation through time method in order to find a good starting point for its weights. Then, these weights are used for the popularity prediction of the content retrieved by the RFB module.

13) Caching Decision (CD): The CD module makes decisions regarding whether to cache the retrieved content by the RFB module. To this aim, it uses the popularities predicted by the PF module.

Let us denote the total number of cached 360^o videos at the SBS $n \in \mathcal{N}$ at the base quality as b_n, and the total number of cached tiles in high quality as f_n. In addition, let the forecast popularities of the cached 360^o videos at the base quality at the time slot $t+1$ form the set $\mathcal{B}^{n,t+1} = \{B^{n,t+1}_1, \dots, B^{n,t+1}_i, \dots, B^{n,t+1}_{b_n}\}$, and the forecast popularities of the cached tiles in high quality form the set $\mathcal{F}^{n,t+1} = \{F^{n,t+1}_1, \dots, F^{n,t+1}_j, \dots, F^{n,t+1}_{f_n}\}$.

When the content that will be prefetched to a user is cached locally or at a neighboring SBS the user resides, it is delivered to the user from the SBS which possesses it. In such case, no cache update decision is made by the CD module. In a different case, the content is retrieved by the RFB module, and a decision is made about whether to cache it. Specifically, first a decision is made about whether to cache at the SBS the prefetched content $z^{t+1}_{u,0}$ regarding the 360^o video indicated by the CP module (when it is not cached). Let the predicted popularity by the PF module for the $z^{t+1}_{u,0}$ be $\lambda^{t+1}_{n,v,0}$. The prefetched content $z^{t+1}_{u,0}$ will be cached at the SBS in the place of the ith cached 360^o video at the base quality if $\lambda^{t+1}_{n,v,0} > B^{n,t+1}_i$, where $B^{n,t+1}_i = \min(\mathcal{B}^{n,t+1})$. Next, in case the 360^o video is cached at the base quality, a decision about whether they should be stored at the CS module is made for each one of the tiles in high quality $\{z^{t+1}_{u,1}, \dots, z^{t+1}_{u,i}, \dots, z^{t+1}_{u,k}\}$ that are not cached. Specifically, if the predicted popularity by the PF module for the tile $z^{t+1}_{u,i}$ is given by $\lambda^{t+1}_{n,v,m}$, the tile $z^{t+1}_{u,i}$ will be cached in the place of the jth cached tile in high quality if $\lambda^{t+1}_{n,v,m} > F^{n,t+1}_j$, where $F^{n,t+1}_j = \min(\mathcal{F}^{n,t+1})$. However, if the initial decision

Algorithm 1 Caching decisions using forecast popularities

```
Offline Phase
Pre-train the LSTM network with historic transition profiles, using the backpropagation through time method
Online Phase
for each time slot $t$ do
    for each user $u$ do
        for each user request $w_{u,i}^{t}, i \in \{0, 1, \ldots, k\}$ do
            if $z_{u,0}^{t+1}$ (all tiles at base quality) are not cached then
                if $\lambda_{n,v,0}^{t+1} > B_i^{n,t+1}$ and $\min(\mathcal{B}^{n,t+1}) = B_i^{n,t+1}$ then
                    Cache $z_{u,0}^{t+1}$ in place of the $i$th cached $360^o$ video at base quality
                end if
            end if
            if $z_{u,0}^{t+1}$ (all tiles at base quality) are cached then
                if $z_{u,i}^{t+1}$ (tile in high quality) is not cached then
                    if $\lambda_{n,v,m}^{t+1} > F_j^{n,t+1}$ and $\min(\mathcal{F}^{n,t+1}) = F_j^{n,t+1}$ then
                        Cache $z_{u,i}^{t+1}$ in place of the $j$th cached tile in high quality
                    end if
                end if
            end if
        end for
    end for
end for
```

regarding the request $z_{u,0}^{t+1}$ was to not cache it, no further cache update decision is made, and none of the content requests $\{z_{u,1}^{t+1}, \ldots, z_{u,i}^{t+1}, \ldots, z_{u,k}^{t+1}\}$ regarding the tiles in high quality are cached. The aforementioned cache update decision process is summarized in Algorithm 1. It takes place at the CD module using the forecast popularities provided by the PF module.

Following the above workflow, the cache is populated with the most popular 360^o videos at the base quality. Tiles in high quality are cached only for the videos with cached base layer tiles. The number of cached tiles in high quality for each cached 360^o video depends on videos' popularity, i.e., the more popular is a 360^o video, the more tiles are cached at the SBSs. Hence, for the least popular videos, a small number of tiles or even no tiles may be cached at the SBSs. This provides our scheme greater flexibility in deciding how many tiles to cache per video, helps increase the cache hits, and enhances the quality of the displayed video, as we see in the next section.

It is worth noting that our system forecasts the evolution of the popularities of all the cached items to be requested, i.e., sets $\mathcal{B}^{n,t+1}$ and $\mathcal{F}^{n,t+1}$ at each time slot. This can be effectively implemented online without the cache size being a limiting factor. This is because the predictions are made at the SBSs, which can be equipped with GPUs and make predictions in the order of μsecs, as was also shown in [17]. Furthermore, LSTM training is performed in the warm-up phase (e.g., off-peak hours), which further lowers the cost of deploying LSTM networks in practical systems.

TABLE I
NOTATION

Symbol	**Physical Meaning**
$\mathcal{N}$	Set of Small Base Stations
$\mathcal{N}_{\mathcal{B}}$	Set of SBSs and MBS
$\mathcal{V}$	Set of 360^o videos
$\mathcal{G}$	Set of GOPs
$\mathcal{L}$	Set of quality layers
$\mathcal{M}$	Set of tiles
$\mathcal{U}$	Set of users
$\mathcal{T}$	Set of time slots
δ_{vglt}	Distortion reduction from obtaining the tile $vglt$
o_{vglt}	Size of the tile $vglt$
C_n	Cache capacity of the SBS n
t_{disp}	Time duration of a GOP

The key notation of our problem is summarized in Table I.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed online edge cache-assisted framework to support 360^o live video streaming. For all the schemes under comparison, when users reside within the transmission range of multiple SBSs, they can be associated with any of these SBSs. This allows users to obtain their data from neighboring SBSs that possess the requested content when it is not available at the primary SBS. We should note that when the users' requests arrive at an SBS, the cache update decisions are made at that SBS regardless if the users obtained their data from a neighboring SBS or the backhaul.

A. Simulation Setup

The schemes under comparison, as well as the proposed scheme, are described below:

1) *Least Frequently Used (LFU):* In this scheme, the network operator keeps track of the number of requests that occurred for each cached 360^o video and tile in high quality of each GOP. When a request for the gth GOP of a 360^o video arrives at an SBS, the LSR algorithm is used to decide which content should be prefetched as follows: a) if no tiles of the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the 360^o video that was requested the least frequently will be evicted from the SBS cache. Then, the tiles indicated by the LSR algorithm will be cached in the corresponding places of the evicted tiles; b) if the tiles of the GOP $g+1$ are already cached at the base quality, but some (or all) of the requested tiles in high quality are not cached, the cached tiles in high quality that were requested the least frequently will be evicted, and the requested tiles that were not cached will be stored in the place of the evicted tiles.

2) *Least Recently Used (LRU):* In this scheme, the network operator keeps track of how recent are the requests that occurred for each cached 360^o video and tile in high quality of each GOP. When an SBS receives a user request for the GOP g of a 360^o video, according to the LSR algorithm: a) if no tiles of the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the 360^o video that was requested the least recently will be evicted from the SBS cache. Then, the tiles indicated by the LSR algorithm will be cached in the corresponding places of the evicted tiles; b) if the tiles of the GOP $g+1$ are already cached at the base quality, but some (or all) of the requested tiles are not cached in high quality, the cached tiles in high quality that were requested the least recently will be evicted, and the requested tiles that were not cached will be stored in their places.
3) *First In First Out (FIFO):* In this scheme, the network operator keeps track of when the requests for each cached 360^o video and tile in high quality of each GOP happened. For a user request for the GOP g of a 360^o video, according to the LSR algorithm: a) if no tiles for the GOP $g+1$ are cached at the SBS, all the tiles of the GOP $g+1$ of the 360^o video that was requested the earliest will be evicted from the cache at the SBS. Next, all tiles of the viewport predicted by the LSR algorithm for the GOP $g+1$ will be cached at the SBS; b) if the tiles of the GOP $g+1$ are already cached at the base quality, but some (or all) of the requested tiles in high quality are different from the tiles of the viewport indicated by the LSR, the cached tiles in high quality that were requested the earliest will be evicted, and the requested tiles that were not cached will replace these tiles.
4) *Proposed Scheme:* In the proposed scheme, the caching decisions are performed following the cache update framework described in Section IV. The proposed scheme uses popularity forecasting to decide with what content to populate the SBSs caches and how to update them. When a decision is made to cache a video at an SBS, all the tiles in base quality are cached in it. Also, for each GOP, a number of tiles that depends on their popularity are cached in high quality.

As is obvious from the description of the schemes under comparison, an item of the $g+1$ GOP under decision is always replacing another cached item of the GOP $g+1$. This approach was followed in order to allow all the GOPs of popular 360^o videos to remain in the cache for their whole video length. This permits users to watch popular videos at a later time instant by accessing the cached content (VoD case). Besides, evicting items corresponding to past GOPs would be possible, however such a strategy would introduce unnecessary complexity, as the entire cache space for multiple GOPs should have been examined.

It should be noted that for the schemes under comparison, when an update decision is made base layer tiles of a 360^o video replace the base layer tiles of another video. The same happens to the tiles of these videos in high quality. However, in this case, the removed tiles may belong to more than one videos. Hence, similarly to our scheme, LFU, LRU, and FIFO can cache an arbitrary number of tiles in high quality.

For all the conducted experiments, unless otherwise specified, we assume a cellular network with $N=3$ SBSs and an MBS covering the area of all SBSs. The coverage range of each SBS is p_n = 200m, and the coverage range of the MBS is p_{N+1} = 2000m. The cache capacity of the SBSs is set to be enough to cache 10% of the 360^o videos content library. This space is calculated assuming that for each GOP of each cached 360^o video, all the tiles at the base quality along with the tiles of one viewport in high quality are cached at the SBS. However, as we have already mentioned, for all schemes, the number of tiles in high quality that will be cached for each 360^o video depends on the corresponding caching policy of each scheme.

We consider that the total number of users is $U=540$, who are randomly placed in the coverage area of the SBSs. The delay at which data is delivered from the cache of the users' primary SBS is $d_{nu} = {}^1\!/_{14}$ sec/Mbit. The delay at which data is delivered from the cache of an SBS that is not the primary one to a user equals to $d_{nu} = {}^1\!/_{13}$ sec/Mbit. When a request for a 360^o video at base quality or tile in high quality is not cached at any of the SBSs the user resides, the delay needed to deliver that request from the backhaul equals to $d_{(N+1)u} = {}^1\!/_{2.9}$ sec/Mbit.

The content library contains $V=100$ videos. Each 360^o video has a duration of 300 GOPs, with each GOP lasting $t_{disp} = 1$ sec. Each GOP of the 360^o videos is encoded in $M=12$ tiles and $L=2$ quality layers, while the size of each viewport is 4 tiles. The considered viewports are depicted in Fig. 7. The bitrate of the base layer is 2 Mbps, while the bitrate of the enhancement layer is 12 Mbps. The distortion reduction achieved by acquiring the base layer of a tile is $\delta_{vg1m} = 30$ dB, while the distortion reduction achieved by obtaining the enhancement layer of a tile is $\delta_{vg2m} = 10$ dB. Although in this paper the 360^o videos are encoded into the same number of layers and tiles, and tiles' layers are encoded at the same bitrate, the proposed system could be extended to deal with tiles of different sizes. For example, a tile may be cached in the place of more than one cached tiles if the popularity of the former tile is higher than the sum of the popularities of the latter set of tiles, which should be next evicted. This approach was not implemented as our paper aims at showing the benefits coming from the exploitation of advanced coding tools for caching 360^o videos and the use of LSTM networks to predict popularity trends. The evaluation of our framework for the case of tiles with different sizes is part of our future work.

We assume that the popularity of the 360^o videos through the GOPs varies with the time. At any given moment, users may stop watching a 360^o video to view another. To this aim, we assume that for the first GOP, the users' requests for the various 360^o videos are decided by sampling a Zipfian distribution [39] with shape parameter $\eta_v = 1$. Hence, the probability of a 360^o video to be selected is given by:

$$p_v = \frac{1/v^{\eta_v}}{\sum_{v\in\mathcal{V}} 1/v^{\eta_v}} \tag{2}$$

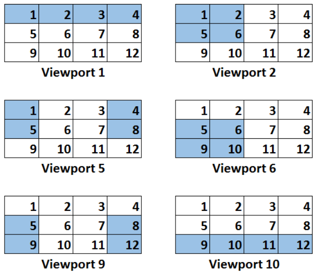

Fig. 7. Illustration of the considered viewports constructed from various tiles.

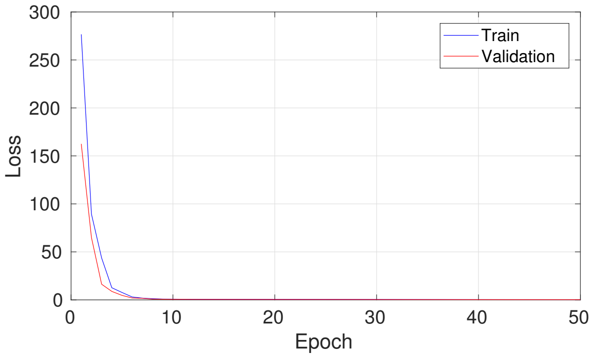

Fig. 8. Evolution of the loss function (estimation error of content's popularity) of with respect to epoch's number using LSTM networks.

To capture the evolution of the popularity of the 360^o videos with the time, inspired by [40], we assume that the probability of a user to stop watching a 360^o video at the GOP $g \in \{2, \dots, G\}$ follows a Weibull distribution. Further, following assumptions made in [40], the 360^o videos that comprise the content library belong to one of 14 video categories, e.g., News, Sports, Education, etc., with equal probability. The Weibull distribution parameters for each video category can be found in [40]. For each GOP, when users stop watching a 360^o video, the probability of selecting a different video to watch follows again a Zipfian distribution with the same shape parameter as the original video dataset.

For simulation purposes, we generate viewports' requests according to realistic navigation patterns obtained by the dataset described in [41]. To this aim, we randomly sample 10 different videos from the dataset, and for each sampled video we obtain 30 trajectories. To assign these trajectories to the considered user requests, we mapped with uniform probability, the 100 videos that comprise the content library to one of the 10 sampled videos. Then, for each user request for a specific 360^o video, according to its mapped index, we assigned with uniform probability one of the 30 available trajectories for that video.

B. LSTM Neural Network training

We consider a deep LSTM network comprised of four layers. The input layer gets as input a 3D tensor with shape (samples, time-steps, 1), with "1" indicating the number of features in our system and "time-steps" the number of previous time slots considered for making the predictions. The feature component of the 3D sensor is set to 1, as there is only one feature for each one of the considered time-series. The input vector in the proposed system consists of different time-series of different objects. It is worth noting that the LSTM network's input is fed with the evolution of each content's popularity for each content retrieved by the RFB module, and not with the sequence of the users' requests. Each of the two hidden layers of the LSTM network has 100 LSTM cells, and the output layer is a dense layer with 1 unit. The deep LSTM network is implemented with the open-source library Keras in Python. The hidden layers have as activation function the ReLu function, and the employed optimizer is Adam. The LSTM network is initially pre-trained (warm-up phase) with 2000 samples of historic data profiles, as explained in Section IV. Each sample represents the evolution in the popularity of a 360^o video (i.e., tiles in base quality) or a tile in high quality over $h = 10$ consecutive time slots (time-steps). The LSTM is pre-trained using a batch size of 300 samples for 50 epochs using MSE as the loss function between the predicted popularities (outputs of the LSTM) and the actual popularities. The historic data profiles are split into a training set and validation set, where the training set contains 90% of the historic data profiles, and the validation set the rest 10%. After the pre-training of the LSTM, the trained weights are used by the PF module to forecast the future content popularities (online phase of Algorithm 1). Specifically, during the time slot t, for each content retrieved by the RFB module, the features regarding the $h-1 = 9$ previous time slots along with the current time slot t are given as an input to the LSTM network, while its output is the predicted popularity at the time slot $t+1$. During the online phase, the weights of the LSTM network are updated every 20 time slots. This is done by randomly sampling 100 samples from the FD module and training the LSTM network with that samples for 20 epochs. From Fig. 8, we can note that the MSE loss during the training of the employed LSTM decreases with the number of epochs in both training and validation sets. We can also see that the loss function converges to zero after only a few epochs, which means that the LSTM network can predict the popularity of the content that is prefetched to the caches of the SBSs with a small error.

C. Parameter Analysis

1) Cache Size: We first study the impact of the cache size on the overall quality of the rendered viewports. To this end, we vary the cache capacity C_n in the range [5, 25]% of the content's library size. We can note from Fig. 9 that the proposed scheme outperforms the schemes under comparison significantly in terms of the overall quality of the rendered viewports in all the range of cache sizes. Specifically, for small cache sizes, i.e., 5%, the performance gap between the proposed scheme and the LFU, LRU, and FIFO is approximately 1.7 dB, 2.1 dB, and 2.2 dB, respectively. For large cache size values, i.e., 25%, this gap grows to about 2.8 dB, 2.9 dB, and 3 dB, respectively. This performance gap is attributed to the fact that the proposed scheme achieves a better cache hit ratio compared to its counterparts, as is evident from Fig.

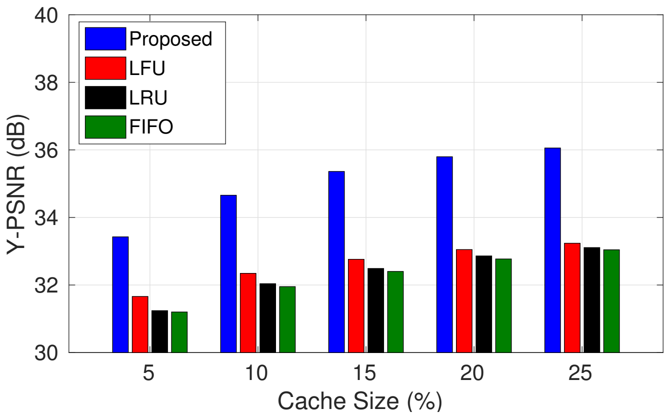

Fig. 9. Y-PSNR of the rendered viewports with respect to the cache size for all the schemes under comparison.

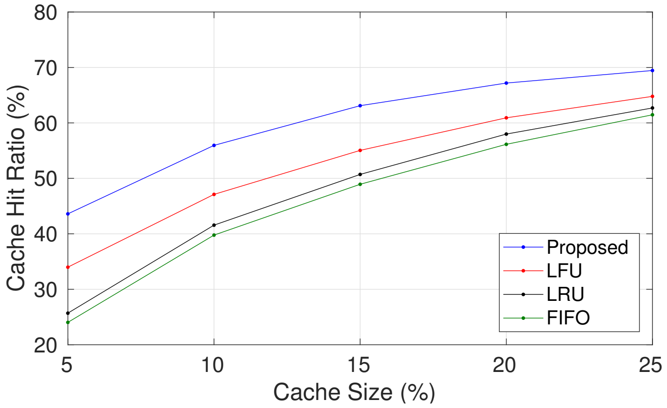

Fig. 10. Cache Hit Ratio with respect to the cache size for all the schemes under comparison.

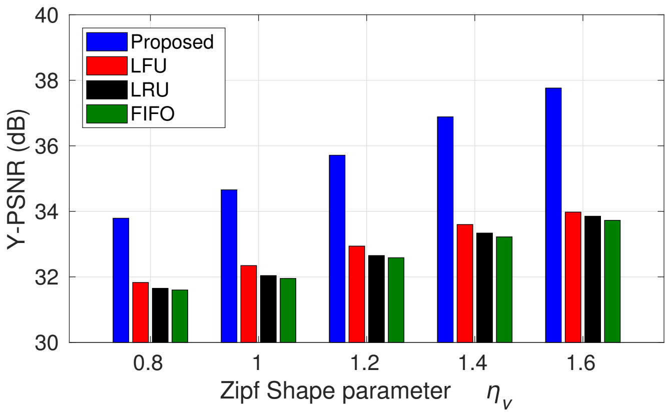

Fig. 11. Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the 360^o videos for all schemes under comparison.

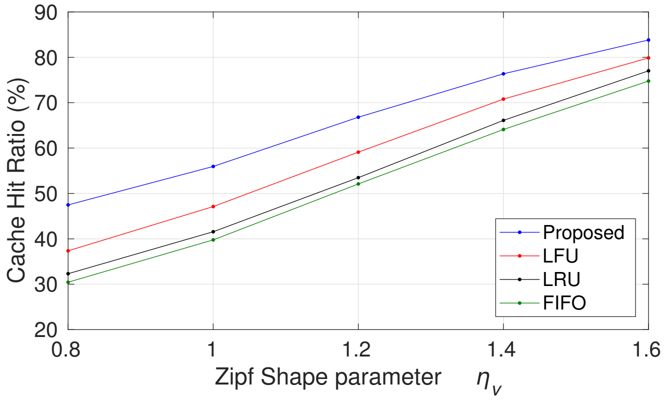

Fig. 12. Cache Hit Ratio with respect to the Zipf shape parameter of the 360^o videos for all schemes under comparison.

10. Specifically, the performance gap between the proposed scheme and the LFU, which is the second best performing scheme in terms of the cache hit ratio for small cache sizes (e.g., 5-10%) is about 9%. When the cache size takes large values, i.e., 25%, this gap closes to about 5%. This is because as the cache size increases, more content can be cached at the SBSs, and all schemes benefit from the additional cache space. The proposed scheme achieves the highest cache hit ratio due to the use of the LSTM network, which can help accurately predict the popularity evolution of the content that will be prefetched to the SBSs caches. Due to the increased cache hit ratio in the proposed scheme, more tiles are delivered in high quality to the users from the caches of the SBSs at a small delay. Thus, it is more likely a greater number of tiles in high quality to be delivered in total from the caches of the SBSs along with the backhaul of the MBS to the users under the tight end-to-end delivery constraint.

2) Video popularity distribution: In Fig. 11, we investigate the impact of the users' requests for the various 360^o videos on the quality of the rendered viewports. To this aim, we vary the Zipf shape parameter η_v in the range $[0.8, 1.6]$. As we can see, an increase in the Zipf shape parameter η_v leads to an increase in the overall quality of the rendered viewports for all the schemes. This is because as the parameter η_v increases, the video popularity distribution gets steeper, and a smaller number of 360^o videos becomes more popular. As a result, the overall cache efficiency increases, as shown in Fig. 12. Thus, more tiles will be served directly from the caches of the SBSs at a small delay, allowing more tiles to be served in total to the users under the end-to-end time constraint. Similarly to the previous comparison, the superiority of the proposed scheme regarding the overall cache hit ratio is attributed to the accurate prediction of the popularities of the content that is prefetched to the SBSs caches, using LSTM networks. As the value of the shape parameter increases from 0.8 to 1.6, the performance gap between the proposed scheme and the LFU widens from 1.9 dB to about 3.7 dB. Similar observations can be made by comparing the proposed scheme with the LRU and FIFO schemes.

3) Viewports popularity distribution: To examine the impact of the viewports' popularity on the overall quality of the rendered viewports, we assume that the viewports' popularity follows a Zipf distribution with shape parameter η_p. We vary the shape parameter η_p in the range $[0.5, 2.5]$, while we keep the cache capacity constant at $C_n = 10\%$. The performance of all the schemes under comparison is shown in Fig. 13. We can note that an increase in the value of the shape parameter η_p leads to an increase in the overall quality of the rendered viewports for all the examined schemes. This is attributed to the fact that as the shape parameter η_p increases, the requests

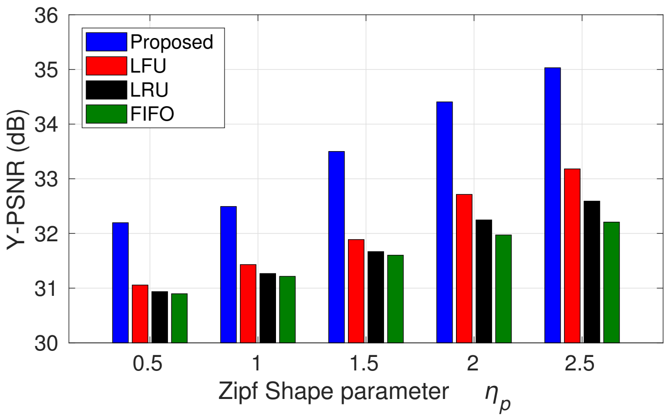

Fig. 13. Y-PSNR of the rendered viewports with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

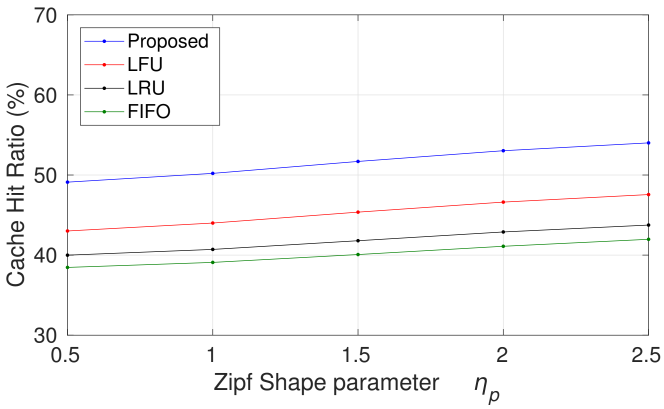

Fig. 14. Cache Hit Ratio with respect to the Zipf shape parameter of the viewports for all schemes under comparison.

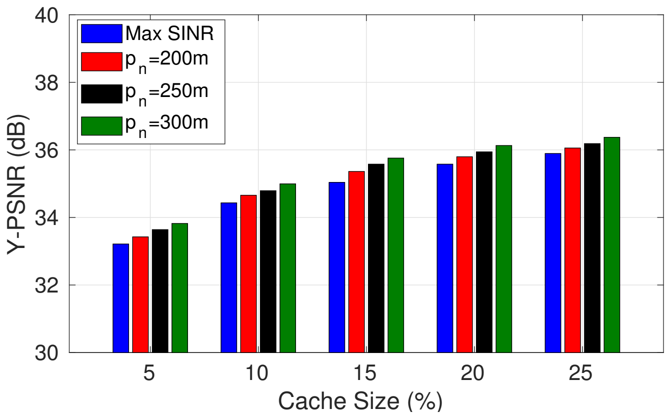

Fig. 15. Y-PSNR of the rendered viewports with respect to the cache size for various SBSs communication ranges.

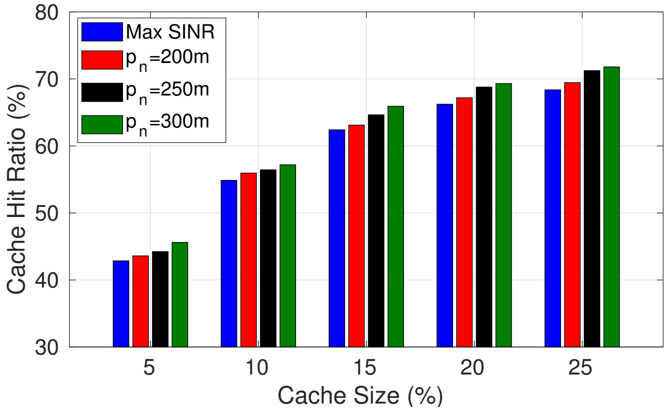

Fig. 16. Cache Hit Ratio with respect to the cache size for for various SBSs communication ranges.

for the viewports become less diverse, and a smaller number of viewports is more popular. This leads most of the requests for tiles in high quality to be served from the SBSs while respecting the end-to-end constraint. Overall, the cache space is better used, as is evident from Fig. 14. For small values of η_p, i.e., $\eta_p = 0.5$, the performance gap between the proposed scheme and the LFU, LRU, and FIFO schemes is about 1.1 dB, 1.2 dB, and 1.3 dB, respectively. When the shape parameter η_p is large, i.e., $\eta_p = 2.5$, the performance gap between the proposed scheme and the LFU, LRU, and FIFO grows to about 1.8 dB, 2.4 dB, and 2.8 dB, respectively.

4) SBS Radius: To understand the impact of users' association with multiple SBSs on the overall quality of the rendered viewports, we vary the SBSs radius p_n in the range $[200, 300]$m. As the radius of the SBSs increases, the overlap between the coverage areas of the SBSs also increases. This results in more users being within the transmission range of multiple SBSs, and hence being able to be associated with multiple SBS. For the sake of completeness, we also examine the case where users are assigned only to the SBS with the maximum SINR. In such a case, the increase of the transmission range of the SBSs from 200m to 300m does not affect the overall rendered quality of the viewports because users are always assigned to the same SBSs regardless of the increase in the SBSs' transmission range. The simulation results are depicted in Fig. 15. As we can see, an increase in the radius of the SBSs leads to an increase in the overall quality of the rendered viewports due to the increase of the cache hit ratio achieved because users will be associated with multiple SBSs (see Fig. 16) for all cache sizes. As expected, when users are assigned to the SBS with the maximum SINR, the overall quality of the rendered viewports is lower compared to its counterparts. This is due to the fact that users are associated with only one SBS from which they can download their data.

5) Backhaul Usage: In Fig. 17, we examine the backhaul usage of the MBS with respect to the cache size of the SBSs for all the schemes under comparison. To this aim, we vary the cache capacity of the SBSs in the range $[5, 25]\%$ of the content library. We can observe that an increase in the cache size of the SBSs leads to a decrease in the backhaul usage of the MBS for all schemes. This is because as the cache size increases, most of the demanded content is cached at the SBSs. As a result, more user requests will be served directly from the SBSs caches with no need to use the MSB's backhaul to fetch content from the core network. We can further note that as the cache size of the SBSs increases from 5% to 25%, the performance gap in terms of the backhaul usage

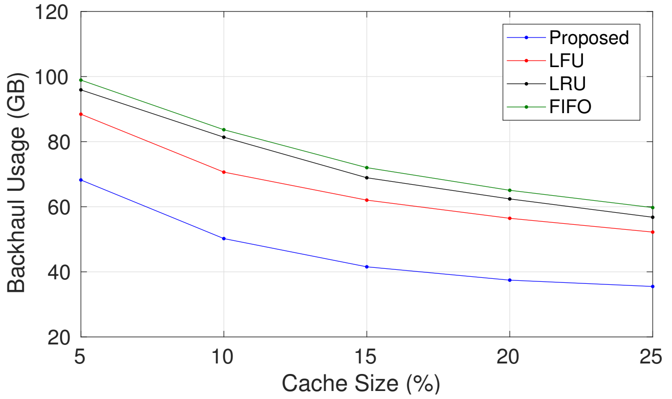

Fig. 17. Backhaul usage with respect to the Cache Size for all schemes under comparison.

between the proposed method and the LFU scheme closes from about 20.2 GB to about 16.7 GB. This is attributed to the fact that as the cache size increases, more content will be delivered to the users from the caches of the SBSs in both cases. Similar conclusions can be drawn when comparing the proposed scheme with the LRU and FIFO schemes.

VI. Conclusion

In this paper, we proposed an edge caching system to support live streaming of 360^o videos in mobile networks. The 360^o videos are encoded in multiple tiles and layers to have higher flexibility in deciding what content to cache at the SBSs. Our framework predicts the evolution of the popularity of video tiles in future GOPs using LSTM networks. The popularity estimates are then used to decide the cache updates at the SBSs and enable prefetching of the content so that it is available on time to the users. To further enhance our proposed method's performance, the potential association of users with multiple SBSs was exploited by allowing users located in the overlapping coverage areas of the SBSs to have access to all the caches of these SBSs at the cost of an increased communication delay. We tested our scheme for both real and synthetic navigation patterns and compared it with the LFU, LRU, and FIFO schemes. The results showed significant gains for the proposed methods over the schemes under comparison at affordable complexity. These gains were observed in terms of the overall quality of the rendered viewports, the cache hit ratio, and the backhaul usage. This is due to the fact that the majority of the demanded content is acquired through the SBSs, and the use of backhaul links is scarce. The evaluation also showed that LSTM networks accurately predict the evolution of tiles' and videos' popularities helping the proposed system make the most out of content prefetching and SBSs caching.

References

[1] K. Bilal and A. Erbad, "Impact of multiple video representations in live streaming: A cost, bandwidth, and QoE analysis," in *Proc. of IEEE Int. Conf. on Cloud Engineering (IC2E'17)*, Vancouver, BC, Canada, Apr. 2017, pp. 88–94.

[2] X. Xie and X. Zhang, "POI360: Panoramic mobile video telephony over LTE cellular networks," in *Proc. of the 13th Int. Conf. on Emerging Networking EXperiments and Technologies (CoNEXT '17)*, Incheon, Republic of Korea, Dec. 2017, pp. 336–349.

[3] A. T. Nasrabadi, A. Mahzari, J. D. Beshay, and R. Prakash, "Adaptive 360-degree video streaming using layered video coding," in *Proc. of IEEE Virtual Reality (VR'17)*, Los Angeles, CA, USA, Mar. 2017, pp. 347–348.

[4] X. Corbillon, G. Simon, A. Devlic, and J. Chakareski, "Viewport-adaptive navigable 360-degree video delivery," in *Proc. of IEEE Int. Conf. on Communications (ICC'17)*, Paris, France, May 2017.

[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 on Multimedia Systems Conf. (MMSys'17)*, Taipei, Taiwan, 2017, pp. 211–216.

[6] S. Rossi and L. Toni, "Navigation-aware adaptive streaming strategies for omnidirectional video," in *Proc. of IEEE 19th Int. Workshop on Multimedia Signal Processing (MMSP'17)*, Luton, UK, Oct. 2017.

[7] A. Zare, A. Aminlou, M. M. Hannuksela, and M. Gabbouj, "HEVC-compliant tile-based streaming of panoramic video for virtual reality applications," in *Proc. of ACM on Multimedia Conf. (MM'16)*, Amsterdam, Netherlands, Oct. 2016, pp. 601–605.

[8] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "A two-tier system for on-demand streaming of 360 degree video over dynamic networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 43–57, March 2019.

[9] T. X. Vu, S. Chatzinotas, and B. Ottersten, "Edge-caching wireless networks: Performance analysis and optimization," *IEEE Trans. on Wireless Communications*, vol. 17, no. 4, pp. 2827–2839, Apr. 2018.

[10] J. Poderys, M. Artuso, C. M. O. Lensbl, H. L. Christiansen, and J. Soler, "Caching at the mobile edge: A practical implementation," *IEEE Access*, vol. 6, pp. 8630–8637, 2018.

[11] P. Maniotis, E. Bourtsoulatze, and N. Thomos, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," in *Proc. of IEEE 21st Int. Workshop on Multimedia Signal Processing (MMSP'19)*, Kuala Lumpur, Malaysia, Sep. 2019.

[12] ——, "Tile-based joint caching and delivery of 360^o videos in heterogeneous networks," *IEEE Trans. on Multimedia*, vol. 22, no. 9, pp. 2382–2395, Sep 2020.

[13] G. Papaioannou and I. Koutsopoulos, "Tile-based caching optimization for 360^o videos," in *Proc. of the 20th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing (Mobihoc'19)*, Catania, Italy, July 2019, pp. 171–180.

[14] P. Maniotis and N. Thomos, "Smart caching for live 360^o video streaming in mobile networks," in *Proc. of IEEE 22st Int. Wkshp on Multimedia Signal Proc. (MMSP'20)*, Tampere, Finland, Sep. 2020.

[15] J. Brownlee, "How to develop LSTM models for time series forecasting," 2020. [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting

[16] A. Narayanan, S. Verma, E. Ramadan, P. Babaie, and Z.-L. Zhang, "Making content caching policies "smart" using the deepcache framework," *SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 5, pp. 64–69, Jan. 2019.

[17] C. Zhang, H. Pang, J. Liu, S. Tang, R. Zhang, D. Wang, and L. Sun, "Toward edge-assisted video content intelligent caching with long short-term memory learning," *IEEE Access*, vol. 7, pp. 152 832–152 846, 2019.

[18] H. Huang, J. Chen, H. Xue, Y. Huang, and T. Zhao, "Time-variant visual attention in 360-degree video playback," in *Proc. of IEEE Int. Symp. on Haptic, Audio and Visual Environments and Games (HAVE'18)*, Dalian, China, Sep. 2018.

[19] L. Bassbouss, S. Pham, and S. Steglich, "Streaming and playback of 16K 360^o videos on the web," in *Proc. IEEE Middle East and North Africa Communications Conf. (MENACOMM'18)*, Jounieh, Lebanon, Apr. 2018.

[20] Y. Kim, J. Huh, and J. Jeong, "Distributed video transcoding system for 8k 360^o VR tiled streaming service," in *Proc. Int. Conf. on Information and Communication Technology Convergence (ICTC'18)*, Jeju, South Korea, Oct. 2018, pp. 592–595.

[21] Y. Li, J. Xu, and Z. Chen, "Spherical domain rate-distortion optimization for omnidirectional video coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1767–1780, 2019.

[22] C. Fan, S. Yen, C. Huang, and C. Hsu, "On the optimal encoding ladder of tiled 360 videos for head-mounted virtual reality," *IEEE Trans. on Circuits and Systems for Video Technology*, pp. 1–1, 2020.

[23] J. Heyse, M. T. Vega, F. de Backere, and F. de Turck, "Contextual bandit learning-based viewport prediction for 360 video," in *Proc. of*

IEEE Conf. on Virtual Reality and 3D User Interfaces (VR'19), Osaka, Japan, Mar. 2019, pp. 972–973.

[24] S. Petrangeli, G. Simon, and V. Swaminathan, "Trajectory-based viewport prediction for 360-degree virtual reality videos," in *Proc. of IEEE Int. Conf. on Artificial Intelligence and Virtual Reality (AIVR'18)*, Taichung, Taiwan, Dec. 2018, pp. 157–160.

[25] Y. Li, J. Xu, and Z. Chen, "Spherical domain rate-distortion optimization for omnidirectional video coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1767–1780, Jun. 2019.

[26] M. Jeppsson, H. Espeland, C. Griwodz, T. Kupka, R. Langseth, A. Petlund, P. Qiaoqiao, C. Xue, K. Pogorelov, M. Riegler, D. Johansen, and P. Halvorsen, "Efficient live and on-demand tiled HEVC 360 VR video streaming," in *Proc. of IEEE Int. Symp. on Multimedia (ISM'18)*, Taichung, Taiwan, Dec. 2018, pp. 81–88.

[27] R. Aksu, J. Chakareski, and V. Swaminathan, "Viewport-driven rate-distortion optimized scalable live 360^o video network multicast," in *Proc. of IEEE Int. Conf. on Multimedia Expo Workshops (ICMEW'18)*, San Diego, CA, USA, Jul. 2018.

[28] Y. Hu, S. Xie, Y. Xu, and J. Sun, "Dynamic VR live streaming over MMT," in *Proc. of IEEE Int. Symp. on Broadband Multimedia Systems and Broadcasting (BMSB'17)*, Cagliari, Italy, Jun. 2017.

[29] X. Liu, B. Han, F. Qian, and M. Varvello, "LIME: Understanding commercial 360^o live video streaming services," in *Proc. of the 10th ACM Multimedia Systems Conf. (MMSys '19)*, Amherst, Massachusetts, Jun. 2019, pp. 154–164.

[30] C. Ge, N. Wang, W. K. Chai, and H. Hellwagner, "QoE-assured 4K HTTP live streaming via transient segment holding at mobile edge," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 8, pp. 1816–1830, Aug. 2018.

[31] A. Mahzari, A. Taghavi Nasrabadi, A. Samiei, and R. Prakash, "FoV-aware edge caching for adaptive 360^o video streaming," in *Proc. of the 26th ACM Int. Conf. on Multimedia (MM'18)*, Oct. 2018, pp. 173–181.

[32] L. Sun, Y. Mao, T. Zong, Y. Liu, and Y. Wang, "Flocking-based live streaming of 360-degree video," in *Proc. of the 11th ACM Multimedia Systems Conf. (MMSys'20)*, Jun. 2020, pp. 26–37.

[33] D. Liu, B. Chen, C. Yang, and A. F. Molisch, "Caching at the wireless edge: design aspects, challenges, and future directions," *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, Sept. 2016.

[34] X. Lei, X. Jiang, and C. Wang, "Design and implementation of streaming media processing software based on rtmp," in *Proc. of 5th Int. Congress on Image and Signal Processing (CISP'12)*, Chongqing, China, Oct. 2012, pp. 192–196.

[35] [Online]. Available: https://engineering.fb.com/ios/under-the-hood-broadcasting-live-video-to-millions/

[36] T. Ergen and S. S. Kozat, "Online training of LSTM networks in distributed systems for variable length data sequences," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 5159–5165, Oct 2018.

[37] N. M. Vural, S. Ergut, and S. S. Kozat, "An efficient and effective second-order training algorithm for LSTM-based adaptive learning," 2019. [Online]. Available: http://arxiv.org/abs/1910.09857

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[39] L. Alexander, R. Johnson, and J. Weiss, "Exploring zipf's law," *Teaching Mathematics and Its Applications: Int. Journal of the IMA*, vol. 17, no. 4, pp. 155–158, Dec 1998.

[40] 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 Communications and Networking Conf. (WCNC'19)*, Marrakesh, Morocco, Apr. 2019.

[41] F. Duanmu, Y. Mao, S. Liu, S. Srinivasan, and Y. Wang, "A subjective study of viewer navigation behaviors when watching 360-degree videos on computers," in *Proc. of IEEE Int. Conf. on Multimedia and Expo (ICME'18)*, San Diego, CA, USA, July 2018.

Pantelis Maniotis received his diploma in Electrical and Computer Engineering from the Aristotle University of Thessaloniki in 2015 and his PhD from the School of Computer Science and Electronic Engineering at the University of Essex in 2020. His interests fall in the areas of multimedia technologies, wireless communications, Virtual and Augmented Reality, edge caching, and machine learning.

Nikolaos Thomos (S'02-M'06-SM'16) is an Associate Professor at the University of Essex, UK and the deputy director of research at School of Computer Science and Electronic Enineering. Previously, he was senior researcher at the Ecole Polytechnique Fédérale de Lausanne (EPFL), and the University of Bern, Switzerland. He received the Diploma and Ph.D. degrees from Aristotle University of Thessaloniki, Greece in 2000 and 2005 respectively. He is an elected member of IEEE MMSP Technical Committee (MMSP - TC) for the period 2019 - 2022. His research interests include machine learning for communications, multimedia communications, network coding, information-centric networking, source and channel coding, device-to-device communication, and signal processing. He received the highly esteemed Ambizione career award from Swiss National Science Foundation (SNSF) in 2008.