Intelligent Video Caching at Network Edge: A Multi-Agent Deep Reinforcement Learning Approach

Fangxin Wang[◇], Feng Wang[‡], Jiangchuan Liu[◇], Ryan Shea[◇], Lifeng Sun[†]
[◇]School of Computing Science, Simon Fraser University, Canada
[‡]Department of Computer and Information Science, The University of Mississippi, USA
[†]Department of Computer Science and Technology, Tsinghua University, China

***Abstract*—Today's explosively growing Internet video traffics and viewers' ever-increasing quality of experience (QoE) demands for video streaming bring tremendous pressures to the backbone network. As a new network paradigm, mobile edge caching provides a promising alternative by pushing video content closer at the network edge rather than the remote CDN servers so as to reduce both content access latency and redundant network traffic. However, our large-scale trace analysis shows that different from CDN based caching, edge caching environment is much more complicated with massively dynamic and diverse request patterns, which renders that existing rule-based and model-based caching solutions may not well fit such complicated edge environments. Moreover, although cooperative caching has been proposed to better afford limited storage on each individual edge server, our trace analysis also shows that the request similarity among neighboring edges can be highly dynamic and diverse, which is drastically different from CDN based caching environment, and can easily compromise the benefits from traditional cooperative caching mostly designed based on CDN environment. In this paper, we propose `MacoCache`, an intelligent edge caching framework that is carefully designed to afford the massively diversified and distributed caching environment to minimize both content access latency and traffic cost. Specifically, MacoCache leverages a multi-agent deep reinforcement learning (MADRL) based solution, where each edge is able to adaptively learn its own best policy in conjunction with other edges for intelligent caching. The real trace-driven evaluation further demonstrates that MacoCache is able to reduce an average of 21% latency and 26% cost compared with the state-of-the-art caching solution.**

I. Introduction

Recent years have witnessed the explosive development of video content streaming. The global video traffic has accounted for 75% of the Internet traffic in 2017 and is estimated to grow four-fold by 2022 [1]. Meanwhile, people are having increasingly higher demands for quality of experience (QoE) on various kinds of video contents, e.g., in immersive 360-degree videos low latency is critical to guarantee smooth viewing. Thus, such massive video traffic and high QoE demands bring tremendous pressure to the backbone network.

Video content caching in content delivery network (CDN) is widely used to reduce duplicated traffic and improve QoE. The traffic between CDN servers and end viewers however can still be largely redundant, which also leads to poor QoEs with large content access latency. Fortunately, mobile edge caching (MEC) [2] provides an alternative solution by pushing video contents closer to end viewers. Particularly, in the emerging 5G network, base stations (BSs) are naturally equipped with edge servers [3] (e.g., Nvidia Jetson TX2[1]) to provide both storage and computation capacity for caching service. Through caching proper video contents at nearby edges, viewers can obtain their target videos locally instead of from remote CDN servers, which not only provides better QoE with lower latency but also saves the core network traffic cost.

Compared to CDN servers, the capacity of edge server is usually less abundant for video caching, which highly depends on carefully designed caching strategies to achieve good performance. Meanwhile, today's content providers mostly use simple *rule-based* solutions such as Least Recently Used (LRU), Least Frequently Used (LFU) and their variants [4], [5] for ease of implementation. Recently, *model-based* approaches have proposed specific models [6]–[8] according to the caching situations to estimate the corresponding caching strategy, with the content popularity usually assumed to be known.

However, different from CDN based caching environment, edge caching environment is much more complicated as verified by our large-scale data analysis on a real video watching trace. Moreover, our trace analysis also indicates that different edge areas can have quite diverse and dynamic request patterns, e.g., the request workload and the requested content in our trace have demonstrated high dynamics and diversities across both temporal and geographical dimensions, especially when considering a finer granularity. As such, although previous works can work well in particular situations by relying on dedicated models, they often turn out to be not adaptive enough under such complicated edge caching context.

On the other hand, as the density of base stations are much larger than CDN servers, e.g., the number of CDN in China is only several hundred[2] while the number of bast stations has achieved 3.72 million[3] by 2018, cooperative edge caching has been proposed to better afford the less abundant storage capacity on each individual edge server [3], [9]–[11]. Yet, our trace analysis shows that despite of certain similarities existing among neighboring edges to facilitate potential cooperations,

[1]https://developer.nvidia.com/embedded/jetson-tx2
[2]https://www.cdnplanet.com/geo/china-cdn/
[3]https://www.statista.com/statistics/989888/china-4g-mobile-base-station-number/

such similarities are also highly diverse and dynamic, which also renders previous solutions not adaptive enough to handle such a highly vibrant, massively diversified and distributed environment.

To this end, we propose `MacoCache`, an intelligent edge caching framework that explores both adaptive intelligence and collaborative intelligence so as to better boost the edge caching performance to minimize the watching latency at viewer side as well as the traffic cost at the service provider side. Inspired by the recent advances of learning in various smart applications [12]–[14], we explore the deep learning, in particular, deep reinforcement learning in our framework to handle the dynamic and diverse situations. However, traditional DRL usually uses one centralized learning agent, which is not feasible for our framework since the massively distributed edges will incur explosive action space. We thus further expand it with multi-agent learning so that each edge can be treated as a learning agent with cooperations facilitated among their neighboring agents. Our multi-agent collaborative caching (`MacoCache`) framework therefore integrates multi-agent learning and DRL as *multi-agent deep reinforcement learning (MADRL)* to well adapt the massively distributed dynamics and diversities and achieve collaborative intelligence for edge caching.

We use the state-of-the-art advantage actor-critic method for the multi-agent learning where an actor network outputs the caching replacement actions and a critic network provides the feedback on the selected action. In MacoCache, each agent needs to consider not only its own caching strategy but also its neighbor agents' caching strategies since the caching behaviors of nearby agents are mutually influential. A simple yet popular approach is using independent Q-learning (IQL) where each agent learns its own policy independently by modeling other agents as part of the environment. This approach however leads to partially observable and non-stationary environment since other agents are updating their own policies at the same time. To stabilize the learning environment, the represented policy fingerprint of each agent is shared among the neighborhood and we scale down the neighborhood rewards to make each agent more locally focused. Besides, we also integrate long short term memory network (LSTM) with the actor-critic model to better handle time series dynamics and diversities by adapting the sequential characteristics from the historical requests. We conduct extensive real trace-driven experiments to evaluate the performance of our MADRL based solution. The results further demonstrate that MacoCache is able to reduce an average of 21% latency and 26% cost compared with the state-of-the-art learning-based caching solution.

The rest of this paper is organized as follow. Section II introduces the data analysis and the motivation of this work. Section III describes the system framework followed by the model formulation. Section IV introduces our MADRL model and its design in collaborative edge caching. We evaluate our solution in section V. Some related work is introduced in section VI. We at last discuss our work in section VII and conclude it in section VIII.

II. Data Analysis and Motivation

In this section, we analyze a real-world video watching dataset to understand the viewer request patterns that will be faced in a distributed edge environment. We collaborate with iQiYi[4], one of the largest video service providers in China in 2019, and collect the video watching trace for mobile users in Beijing for two weeks in May. The trace contains about 17 million sessions, recording the user ID, timestamp, video content name and the GPS locations of each request. Through the data analysis, 1) we have noticed the massive heterogeneity and dynamics of video requests in terms of request distribution and request content features, which demonstrates the necessity of a highly adaptive edge caching strategy; 2) our trace analysis also reveals the content similarities among neighboring edge areas, which indicates the importance of incorporating more adaptive and intelligent approaches for collaborative edge caching.

A. Analysis of Request Heterogeneity and Dynamics

Request workload analysis. We first study the viewer request distribution and the patterns hidden behind from both spatial dimension and temporal dimension. For ease of representation, we select a 10 km $\times$ 10 km target area in Haidian district in Beijing as a case study. Fig. 1 plots the heatmap of the request density for different areas using records of one typical day in our dataset. We can find that the content request demands are highly skewed from the spatial dimension. For example, the northeast area (knows as a mixed area mainly for business and university) has much higher request demands than the northwest area (known as a large scenic area).

We also examine the request workload characteristics from the temporal dimension. For more fine-grained display, we split the target area into 10 $\times$ 10 small grid areas where each area is treated as an edge area with a corresponding base station (or edge server) for request serving. Fig. 2 demonstrates the request workload of these edge areas in different hours of a week, where y-axis represents the edge area id (we index each area by geographic order), x-axis represents the time slot by hour, and the value represents the request workload. We can find that the request workload of most areas has shown a certain level of regularity, especially the periodicity. However, the temporal patterns also show tremendous dynamics in both duration and request amounts for different areas, e.g., for area index from 0 to 10 the high demand starts from the afternoon while for area index from 68 to 70 the high demand lasts all day long. Such heterogeneity of request workload in both spatial and temporal dimensions brings quite dynamic request features in different areas and different periods, which calls for a more adaptive caching strategy to capture the time and space diversified patterns to achieve even better performance.

Request content analysis. We next study the heterogeneity of the request contents among different edge areas. We define *mean content distribution* and *std content distribution* as the mean value and standard deviation of the request times for a

[4]https://www.iqiyi.com/

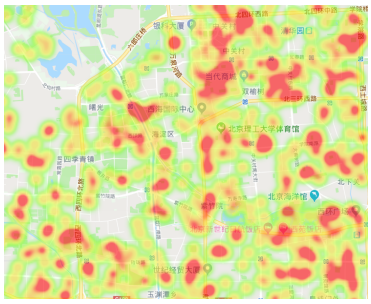


Fig. 1. The heatmap of viewer request density for one day in our dataset.

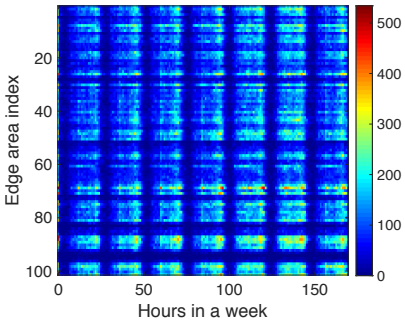


Fig. 2. The request demands for each small area during each time periods. X-axis indicates the time periods by hour and y-axis indicates the area id.

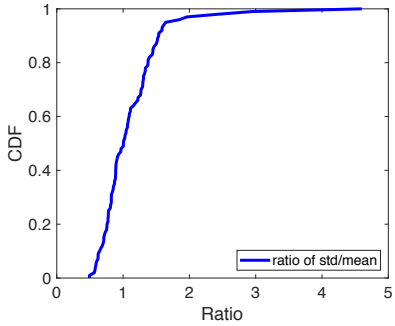


Fig. 3. The CDF plot of the std/mean ratio of different contents among all edge areas in one typical day.

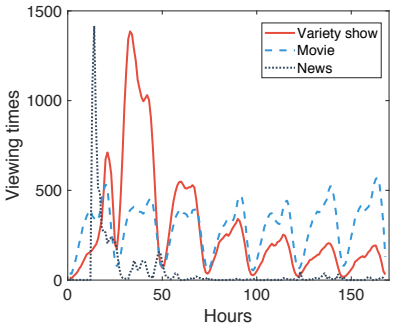


Fig. 4. The request demands of different video contents within each hour for one week.

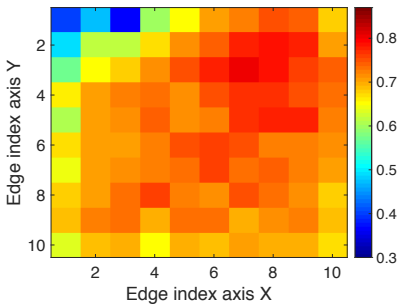


Fig. 5. The color map of average content similarity for each edge in one day.

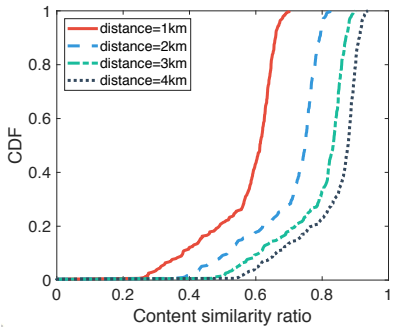


Fig. 6. The CDF plot of the content similarity ratio of each hour in a week when we average all edge areas.

particular video content among all the edge areas, respectively. And we denote the *std/mean ratio* as the ratio of the standard deviation value to the mean value. It is intuitive that a higher *std/mean ratio* means the requests of content are more imbalance among different edge areas. Fig. 3 shows the CDF plot of the *std/mean ratio* of the accessed videos in a typical day. We can find that for more than 50% contents the *std/mean ratio* is higher than 1, and for about 5% quite imbalanced contents the ratio even exceeds 2. This result indicates that the requested contents of different edge areas are highly variant and each edge area can have its specific features. Thus, it should be beneficial if different edges can adjust the caching strategy based on their own content access patterns.

We then examine the content access patterns from a perspective of the content itself. We select three representative video contents (i.e., a variety show, a movie and a news report) and illustrate the total request demands of each hour for one week in Fig. 4. We can find that the movie follows an obvious periodic access pattern during a long time and the request number is quite stable. The variety show also follows a periodic pattern, while the general popularity has a slowly decreasing trend with time. The request number of the seventh day is only 15% of the first day for this variety show. As to the news report, the demands first rise sharply and then vanish very quickly within two days. This observation reveals that the request patterns of different videos are also highly correlated to their own contents, e.g., time-sensitive videos usually disappear more quickly. From this perspective, it is important to design highly adaptive caching solutions that are capable to quickly identify and adjust to the characteristics of video contents so as to facilitate the caching performance.

B. Analysis of Content Similarity among Neighboring Edges

We next look into the content access similarity between neighboring areas. We also split the target area into 10×10 small grid areas and each edge area is a 1km*1km square. And we define two areas as neighbors if their distance is within a given neighboring distance threshold. The total content requests inside one area i within time period T are denoted as R_i^T. We define a request r_s in R_i^T as a *similar request* if the requested video content of r_s is the same as any other request in R_j^T, where $j \in neighbor(i)$ indicates any neighbor area of area i. We thereby define the *content similarity ratio* CSR_i^T for area i in time period T as

$$CSR_i^T = \frac{\|\sum r_s \in R_i^T\|}{\|R_i^T\|}, \forall r_s \ is \ a \ similar \ request \quad (1)$$

Fig. 5 plots the color map of the general average content similarity of each edge area for one day with the default distance threshold as 1km, where the x- and y-axis indicate the edge area index and the value of each area represents the content similarity value. We can find that the content similarity varies a lot among different areas. For example, edge areas in the northeast have relatively higher content similarity, which

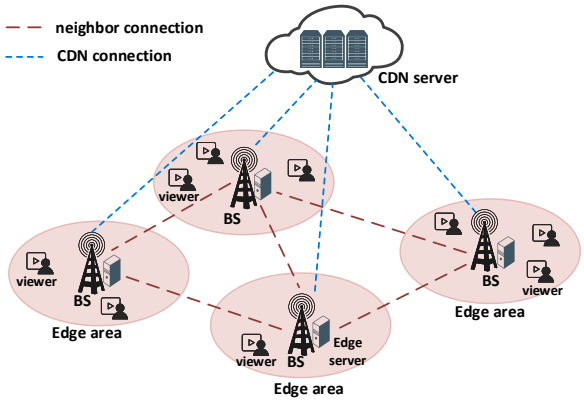


Fig. 7. Our edge caching framework.

is probably because several universities reside in that region and students may have similar content preferences. Note that the content similarity and request density are not exactly correlated. For example, comparing Fig. 1 and Fig. 5 we observe that the southeast area has quite dense video requests while the content similarity is not that high.

Since the CSR is affected by the neighboring distance threshold we set, we also investigate the impact of this threshold setting on CSR values of different edge areas. We show the characteristics from the perspective of time. Fig. 6 demonstrates the CDF plot of the CSR of each hour in a week when we average all edge areas. This result indicates that the CSR of edge areas is also time-varying since in different time periods people's viewing preferences can also be quite different.

These analysis insights above reveal both the potentials and the challenges for cooperative caching among neighboring edges. On one hand, the average high content similarity among neighboring areas is greatly beneficial to cooperative caching since missed requests in one edge area are very likely to be served by nearby neighbors. On the other hand, the content similarity is quite dynamic with time and heterogeneous across different edges so that the inherent characteristics should be carefully considered. Thus, we need a more intelligent and adaptive design for cooperative edge caching strategy so as to jointly consider the caching decisions of neighboring edges given their content similarities may vary a lot over both time and geographical locations.

III. Framework and System Model

A. Edge Caching Framework

We consider a typical edge caching scenario for mobile video streaming, as illustrated in Fig. 7. BSs are distributed in a citywide area equipped with edge servers, which provide storage capacity for video caching and computation ability for cache decision making. Each BS serves the local video requests within its coverage range. BSs are connected to the remote CDN server via the backbone network. We assume that the CDN server has enough storage capacity and has already cached all the requested video contents. As a comparison, the storage capacity of each BS is usually limited and can only cache a small portion of popular video contents.

In the 5G network, BSs are able to communicate with other neighboring BSs rather than work individually [3]. Each BS can retrieve the requested video contents from its neighboring BSs via the fronthaul links (e.g., using the high-bandwidth and low-latency CPRI links for data transmission [9], [15]). Since obtaining video contents from a neighboring BS is faster and more cost-effective than from the CDN server [10], CDN fetching will have the lowest priority. Thus, serving a request should have three statuses, i.e., *local hit*, *neighbor hit* and *CDN hit*, with the following steps. First, when a request arrives, the local BS returns the cached content immediately if found in local cache. Second, if local cache misses, the local BS turns to its neighboring BSs for such content and returns the content if it exists. Third, the local BS fetches the content from the CDN server to serve this request. In our context, time is divided into continuous caching periods, where each edge conducts cache replacement only at the end of each period.

B. Problem Formulation

We first study the offline scenario of the edge caching problem assuming the video requests are known in advance, which tries to minimize both the content access latency and the traffic cost. We assume there are a set of $\mathbf{E} = \{1, 2, ..., E\}$ edges (or BSs) distributed at fixed locations in a large service region. For each edge e, its neighbors are denoted as $\mathcal{N}_e$. The video content files $\mathbf{F} = \{1, 2, ..., F\}$ can be cached in these edges to serve video requests. For ease of presentation, we assume that all the content files have the same unit size since we can divide video files into chunks of the same size. Each edge e has a maximum storage capacity C_e. The CDN server is denoted as c with enough capacity to cache all the contents.

Since every viewer request belongs to one particular edge area and will be served by the BS therein, we aggregate each viewer request to its corresponding BS (referred to as the *home* BS) equivalently. We denote the number of requests for content f in edge e within time period t as $d^t_{e,f}$. A binary variable $x^t_{e,f,j}$ is used to represent whether the video requests for content f belonging to edge e at time t should be served by server j ($x^t_{e,f,j} = 1$) or not ($x^t_{e,f,j} = 0$), where $j \in \{\mathbf{E}, c\}$ can be either an edge or the CDN server. We also use a binary indicator $y^t_{f,j} \in \{0, 1\}$ to represent whether video content f is cached in server j ($y^t_{f,j} = 1$) or not ($y^t_{f,j} = 0$).

In our edge caching context, there may exist three components of latency, including viewer-to-edge latency, edge-to-edge latency, and edge-to-CDN latency, according to different cache hit situations. Without loss of generality, we neglect the viewer-to-edge latency since it is contained in all requests. Thus, we calculate the total transmission latency $\mathcal{L}^t_e$ for edge e within time t as:

$$\mathcal{L}^t_e = \sum_{f \in \mathbf{F}} \sum_{j \in \{\mathcal{N}_e, c\}} d^t_{e,f} x^t_{e,f,j} l_{e,j} \tag{2}$$

where $l_{e,j}$ indicates the latency between edge e and another server j. Similarly, for the video traffic cost, we neglect the

viewer-to-edge cost. And we can calculate the video access traffic cost $\mathcal{C}_e^t$ as:

$$\mathcal{C}_e^t = \sum_{f \in \mathbf{F}} \sum_{j \in \{\mathcal{N}_e, c\}} d_{e,f}^t x_{e,f,j}^t p_{e,j} \tag{3}$$

where $p_{e,j}$ is the traffic cost between e and server j. At the end of each caching period, if the videos to be cached at the next moment are not exactly the same as the currently cached, each edge needs to fetch the new ones from neighbor edges or the CDN server, which will also introduce extra traffic cost. We denote such cost as video replacement traffic cost and calculate it for edge e at time t as

$$\mathcal{R}_e^t = \sum_{f \in \mathbf{F}} \sum_{j \in \{\mathcal{N}_e, c\}} max\{y_{f,e}^t - y_{f,e}^{t-1}, 0\} x_{e,f,j}^{t-1} p_{e,j} \tag{4}$$

Integrating Eq. 2 to Eq. 4 together, we can formulate cache replacement problem (Ω) that aims to minimize the total content access latency and traffic cost (including video access cost and video replacement cost) for all requests as:

$$min: \quad \sum_{t \in T} \sum_{e \in \mathbf{E}} (\alpha \mathcal{L}_e^t + \beta \mathcal{C}_e^t + \beta \mathcal{R}_e^t) \tag{5}$$

$subject\ to:$

$$\sum_{j \in \{\mathcal{N}_e, c\}} x_{e,f,j}^t = 1, \forall e \in \mathbf{E} \tag{6}$$

$$\sum_{f \in \mathbf{F}} y_{f,e}^t \leq C_e, \forall e \in \mathbf{E} \tag{7}$$

$$x_{e,f,j}^t \leq y_{f,j}^t \tag{8}$$

$$x_{e,f,j}^t \in \{0, 1\} \tag{9}$$

$$y_{f,j}^t \in \{0, 1\} \tag{10}$$

where α and β are weighted factors to adjust the preference between latency and traffic cost. Eq 6 guarantees that each request will be served by only one edge or the CDN server. Eq. 7 indicates that the storage usage of each edge should not exceed the capacity. Eq. 8 ensures that a viewer request can only be served by edges or the CDN server that have cached the corresponding video content. Finally, Eq. 9 and Eq. 10 restrict the optimization variables as binary value.

Lemma 1. *The edge caching decision problem Ω is NP-complete.*

Proof. Our problem Ω tries to minimize both the transmission latency and traffic cost for each time period. We first consider a simplified case Ω' that removes Eq. 4 and only focus on one time slot. Given a deployment solution, it is obvious that our problem can be verified in polynomial time. Besides, this problem is actually equivalent to the *Helper Decision Problem* (HDP) described in [7], where the edge in our problem plays the same role as *helper*. Hence, the problem HDP that has been proven to be NP-complete [7] can be directly reduced in polynomial time to Ω'. Since problem Ω' is a simplified case of Ω, we then prove that our problem Ω is NP-complete. ■

From the analysis of the offline scenario, we know the complexity of solving the cooperative edge caching problem. Solving the online form is even more challenging in the following two aspects. First, the *a prior* information of viewer request pattern is seldom available in practice so that model-based solutions may not well adapt to the request dynamics and make intelligent caching decisions. Second, the caching strategies of each edge are mutually influential, making it difficult to achieve global optimality, especially considering a long term optimization.

The recent success in both deep reinforcement learning and multi-agent learning provides an alternative perspective for this problem. The rich historical viewer request patterns offer invaluable data resources that could be utilized for a data-driven caching solution. Specifically, the learning-based approach can not only well capture the hidden dynamics of local and related edges but also enable an end-to-end solution from request prediction to cache decision. Given these unique advantages, we propose a multi-agent deep reinforcement learning (MADRL) based approach to solve this problem, which is described in the following section.

IV. Multi-Agent Deep Reinforcement Learning for Cooperative Edge Caching

In this section, we present MacoCache, a multi-agent deep reinforcement learning (MADRL) based cooperative edge caching framework that adaptively makes intelligent caching decisions to minimize both content access latency and traffic cost. We start from introducing the background of single-agent deep reinforcement learning. We then present the multi-agent advantage actor-critic (MAA2C) training policy used in our MADRL model and how we stabilize the training process. At last, we describe how we implement our MADRL model to solve the edge caching problem.

A. Background of Single-Agent Deep Reinforcement Learning

In traditional single-agent deep reinforcement learning, there is an agent trying to make proper decisions based on the past experience and the given rewards. Specifically, at each time slot t, the agent observes a state s_t and takes an action a_t based on its policy π. The agent then transits to state s_{t+1} with a reward r_t. The target of the agent is to find an optimal policy π to select actions so as to maximize the discounted accumulated rewards $R^t = \sum_0^\infty \gamma^t r_t$, where γ is a discounted factor to reduce the importance of future rewards. Many advanced learning algorithms are proposed in recent years to enable the wide application of DRL, such as deep Q-learning (DQN) [16], double DQN [17], deep deterministic policy gradient (DDPG) [18], A3C [19] and so on. In our edge caching context, however, single-agent DRL is not applicable for two reasons. First, as an independent caching unit, each edge should have its own caching strategy based on the corresponding user request and content access patterns, which could be significantly different among different edges. Second, using one central agent for caching decision is not scalable, which restricts the practical usage. We therefore expand it to MADRL to solve this problem.

B. Multi-Agent Advantage Actor-Critic Learning Model

We consider applying multi-agent advantage actor-critic (MAA2C) learning model to solve this reinforcement learning (RL) problem. In our multi-agent settings, each edge e serves as a learning agent with an actor network (as θ) and a critic network (as ω). The actor network is trained to learn a policy π_{θ_e} for caching decision making, while the critic network is trained to learn a value function V_{ω_e} as an estimate of the expected total reward. These agents work concurrently to explore the optimal edge caching strategy. Note that the multi-agent environment for edge caching is partially observable, i.e., each agent e is able to observe and communicate with its neighbor agents $\mathcal{N}_e$. Then the input state of an agent will also include its neighbors' states, denoted as

$$s_{t,\mathcal{Z}_e} = \{s_{t,e}, \{s_{t,j}\}_{\forall j \in \mathcal{N}_e}\} \tag{11}$$

where $\mathcal{Z}_e$ includes both itself and its neighbors.

Independent Q-learning (IQL) is one of the most straightforward and popular approaches to solve multi-agent RL problems. In IQL, the Q-function of each agent only depends on its own observation and action, treating the other agents as part of the environment. Based on this idea, we can easily extend IQL to independent A2C (IA2C) with the actor-critic method. However, simply using IA2C for multi-agent RL problems is problematic as the environment will become non-stationary in the experience replay from the view of each agent [20]. Actually, the behaviors of all agents are implicitly included in the environment dynamics, while the learning policies are continuously updating, making the learning process hard to converge.

We adopt two methods to stabilize the training process of each agent and the enhanced approach is referred to as MAA2C. The first one is a fingerprint-based method that includes the behavior policies of neighboring agents to make the environment stationary. Since the whole policy network that uses deep network is too large to be directly included [20], we turn to include a low-dimensional fingerprint, i.e., the probability simplex of neighborhood agents as $\pi_{t,\mathcal{N}_e} = \{\pi_{t,j}\}_{\forall j \in \mathcal{N}_e}$. This inclusion is reasonable since each agent is aware of the policy change of other agents and is able to adapt to the environment dynamics. Therefore, the behavior policy of each agent can be represented as

$$\pi_{t,e} = \pi_{\theta_e}(a_{t,e}|s_{t,\mathcal{Z}_e}, \pi_{t-1,\mathcal{N}_e}) \tag{12}$$

Besides the fingerprint, we also adjust the impact of neighborhood policies on each agent. Since viewers belonging to neighboring agents can also obtain videos from the current agent, so we need to jointly consider the rewards of both current agent and its neighboring agents when updating policy. In our edge caching context, we try to make each agent focus more on its local requests, which can not only make the policy updating more locally correlated but also increase the probability of local-hit. For this reason, we introduce *neighbor reward weight* $\delta_{i,j} \in [0, 1]$ to relax neighborhood rewards. The weighted reward for agent e is then calculated as

$$\tilde{r}_{t,e} = r_{t,e} + \sum_{j \in \mathcal{N}_e} \delta_{e,j} r_{t,j} \tag{13}$$

The value of $\delta_{e,j}$ is expected to be inversely proportional to the latency between agent e and j, e.g., $\delta_{e,j} = \frac{l_{max} - l_{e,j}}{l_{max}}$, where l_{max} is the largest edge to edge latency.

With the two proposed stabilization methods, the MAA2C model updates both the policy and value function of each agent during each time period in an experience-replay way. We train each agent using a policy gradient algorithm that estimates the parameter gradient direction towards the maximized total weighted rewards. For each training step, we can first calculate the estimate of the advantage function as

$$A_{t,e} = \tilde{r}_{t,e} + \gamma V_{\omega_e}(s_{t+1,\mathcal{Z}_e}, \pi_{t,\mathcal{N}_e}) - V_{\omega_e}(s_{t,\mathcal{Z}_e}, \pi_{t-1,\mathcal{N}_e}) \tag{14}$$

With the estimated advantage, we update the actor network policy π_{θ_e} through increasing the likelihood of selecting optimal actions. To encourage exploration and prevent premature convergence to suboptimal policies, we also introduce an entropy term as stated in [19]. Then the parameterized network θ_e of agent e can be updated as

$$\begin{aligned} \theta_e \leftarrow & \theta_e + \eta_\theta \sum_t \nabla_{\theta_e} log \pi_{\theta_e}(a_{t,e}|s_{t,\mathcal{Z}_e}, \pi_{t-1,\mathcal{N}_e}) A_{t,e} \\ & + \beta' \nabla_{\theta_e} H(\pi_{\theta_e}(\cdot|s_{t,\mathcal{Z}_e}, \pi_{t-1,\mathcal{N}_e})) \end{aligned} \tag{15}$$

where the term $H(\cdot)$ is the entropy, η_θ is the learning rate of the actor network, and β' is a hyperparameter to control the entropy term. We use temporal difference method to update the parameterized critic network as

$$\begin{aligned} \omega_e \leftarrow & \omega_e - \eta_\omega \sum_t \nabla_{\omega_e} [\tilde{r}_{t,e} + \gamma V_{\omega_e}(s_{t+1,\mathcal{Z}_e}, \pi_{t,\mathcal{N}_e}) \\ & - V_{\omega_e}(s_{t,\mathcal{Z}_e}, \pi_{t-1,\mathcal{N}_e})]^2 \end{aligned} \tag{16}$$

where η_ω is the learning rate of the critic network. After the A2C based training, the actor network can be used for caching selection for every agent.

C. MADRL Design for Edge Caching

We next describe the detailed design of MADRL for our edge caching context, including the state, action, reward design, and the learning network architecture.

State and observation design. The state of each agent should include both the current caching situations and the request demand situations. Thus, we define the state for an agent e as $s_{t,e} = \{\mathbf{y}^t_{\mathbf{f},e}, \mathbf{d}^t_{e,\mathbf{f}}\}$, where $\mathbf{d}^t_{e,\mathbf{f}} = \{d^t_{e,1}, ..., d^t_{e,F}\}$ indicates the request demand and $\mathbf{y}^t_{\mathbf{f},e} = \{y^t_{1,e}, ..., y^t_{F,e}\} \in \{0, 1\}$ records the caching state. As described, each agent also need to include the neighborhood states and their corresponding behavior policies. Therefore, the observation of an agent e to be included to the input network is

$$\mathcal{O}(t, e) = \{s_{t,\mathcal{Z}_e}, \pi_{t-1,\mathcal{N}_e}\} \tag{17}$$

Action design. At the end of each time period, an agent will observe the input from the environment and make the next

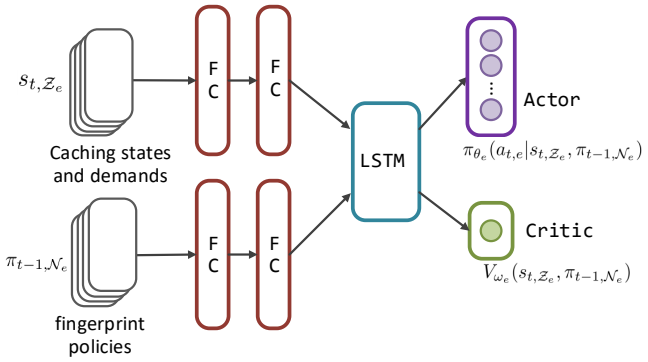


Fig. 8. The learning network architecture.

action based on its policy. We define an action of agent e as $a_{t,e} = \{a^f_{t,e}\}_{\forall f \in \mathbf{F}}$, where $a^f_{t,e} = 1$ means we cache the content f and $a^f_{t,e} = 0$ otherwise. The total cached contents should not exceed C_e so that $\sum_{f \in \mathbf{F}} a^f_{t,e} \leq C_e$. Many existing works model the action space as selecting one cached content for replacement using one-hot encoding. This is however not practically feasible in our edge caching context since we need to select a set of video content to cache for each decision. To address this problem, we define the output space of each actor network as the caching probability of each content and the top C_e contents rather than only one with the highest probability will be selected for caching.

Reward design. Each learning agent receive a reward $r_{t,e}$ after taking their own actions. We define the reward as the negative value of the weighted sum of the transmission latency and traffic cost. Recall the definition in section III-B, we denote the reward of agent e at time period t as

$$r_{t,e} = -\sum_{f \in \mathbf{F}} (\alpha d^t_{f,e} l_{e,j'_f} + \beta d^t_{f,e} p_{e,j'_f} + \beta max\{y^t_{f,e} - y^{t-1}_{f,e}, 0\} p_{e,j'_f}) \tag{18}$$

where $j'_f = argmin_{j \in \{\mathcal{N}_e, c\}, y^t_{f,j}=1}(\alpha l_{e,j} + \beta p_{e,j})$ indicates the neighboring edge (or the CDN server) that has the requested content and leads to the minimal sum of transmission latency and traffic cost. Note that if one request achieves local hit, the latency and traffic cost are zero.

Network architecture. Our MAA2C model uses deep neural networks to represent the learning policy and the learning network architecture is illustrated in Fig. 8. Traditional learning models usually use deep neural network (DNN) as the architecture for the actor and critic network. The past request information is also included to learn the historical patterns. This simple architecture however may not be able to fully explore the hidden sequential patterns of the video requests. Besides, including the historical requests into the observation will also greatly increase the input space, making it difficult for training. To address this problem, we use long short term memory (LSTM), which is an advanced learning model that is widely used for time series processing, to capture the hidden content request patterns. Specifically, the LSTM layer is deployed at the last layer before the final output layer so that the feature representations of the past time periods can be integrated together for the action decision at the current time period. Before the LSTM layer, the input observation will first be fed to fully connected layers for processing.

V. Performance Evaluation

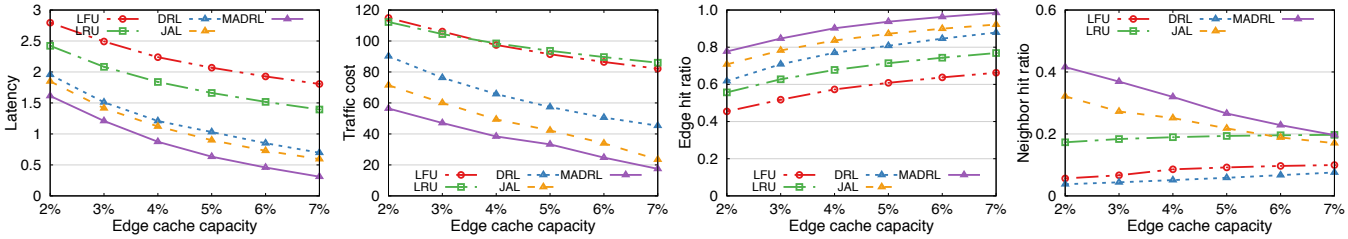
In this section, we conduct extensive experiments to evaluate the performance of our MacoCache edge caching strategy. Specifically, using real trace-driven evaluations, we demonstrate the superiority of MacoCache over classic rule-based solutions and state-of-the-art learning-based solutions.

A. Setup and Methodology

Evaluation setup. We collect a real data trace of viewers' requesting records from iQiYi for two weeks and extract the request and content access patterns for evaluation. Without loss of generality, we randomly select 30 edge areas from our target 10km*10km region mentioned in section II. We assume a BS (or agent) located at the center of each edge area will serve the requests belonging to each corresponding area. Since we cannot directly obtain the real transmission latency between neighboring edges, we set the latency proportional to the Euclidean distance between them [21]. The transmission latency between an edge and the CDN server is set as 5 times of the average latency between any two neighbors in our environment. The traffic cost is set as 1 for neighbor hit and 5 for CDN hit. To balance the transmission latency and the traffic cost in our target, we set α as 1 and β as 2 to make these two components comparable. We define the neighbor edges of one particular edge as the nearest $\mathcal{N}$ surround edges, where the default neighbor number is set as 8. Besides, we set the default time slot granularity as every one hour. Based on our analysis of the real viewing dataset, the requested contents are highly skewed where only a small portion of contents are frequently requested by viewers. Given this situation, we only consider caching those video contents that are requested no less than 10 times in our target region. And the default cache capacity for every edge is set as 4% of our target video contents.

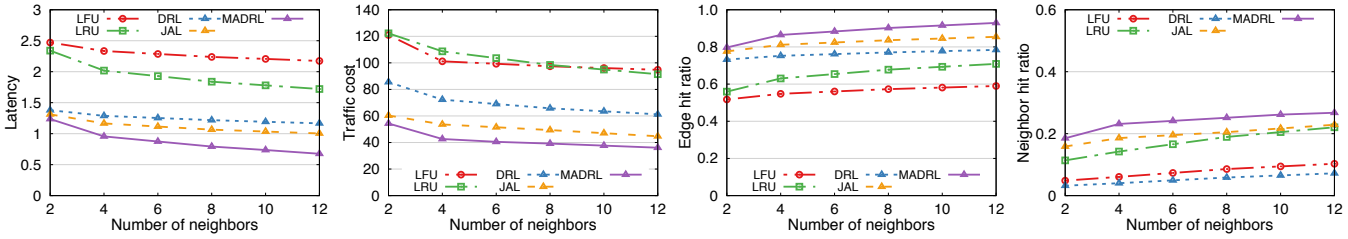
We implement the MacoCache learning model using pytorch [22], which runs on a server with dual GTX 1080 Ti GPU cards, dual Intel I7 3.6 GHz CPU cards and 32GB memory. The learning rate for the actor network η_θ and the critic network η_ω is set as $5e-4$ and $2e-4$, respectively. We set γ as 0.99 by default. We consider the caching states of the past 24 time slots as the input for LSTM. The size of the hidden layers is both 2048. We use the previous 80% data for training and the rest of data is used for evaluation.

Baseline methods. We compare our MacoCache method with several baseline methods as follows. 1) Least recently used (LRU): This method always caches the most recently used content in the cache and evicts those that are least recently used. 2) Least frequently used (LFU): This method counts the frequency of contents and always caches the most frequently used ones. 3) Deep reinforcement learning (DRL) [23]: DRL uses the same learning architecture as our proposed method except that it only considers the caching situations of the local edge. 4) Joint action learners (JAL) [24]: JAL utilizes



(a) Average latency (b) Average traffic cost (c) Edge hit ratio (d) Neighbor hit ratio

Fig. 9. The average latency, traffic cost, edge hit ratio and neighbor hit ratio of different approaches for different cache size of each edge.



(a) Average latency (b) Average traffic cost (c) Edge hit ratio (d) Neighbor hit ratio

Fig. 10. The average latency, traffic cost, edge hit ratio and neighbor hit ratio of different approaches for different number of neighbors for each edge.

a stateless multi-agent Q-learning method to conduct caching decisions. Compared to MacoCache, it does not maintain the historical request patterns and does not use a deep neural network as policy representations. Unless otherwise specified, the result of our model is represented as MADRL in the experiments.

Evaluation metrics. We mainly evaluate the following performance metrics. 1) Average latency: It is calculated as the average latency value for all requests in every edge in a given time period. 2) Average traffic cost: This metric includes two parts of cost. The first part is the average total traffic cost of requests in each edge for each time period. And the second part comes from the cost of content prefetching for each edge at the end of each time period. 3) Edge hit ratio: This metric is calculated as the sum of requests with local hit and neighbor hit divided by the total number of requests. It reflects the ratio of content requests that are served by edge servers instead of the remote CDN server. 4) Neighbor hit ratio: It is calculated as the sum of requests with neighbor hit divided by the total number of requests.

B. Evaluation Results

We first consider the impact of different edge cache capacity on our evaluation metrics. We change the edge cache capacity from 2% to 7% of the total active content number and show the results in Fig. 9. From Fig. 9(a) and Fig. 9(b) we can find that MADRL outperforms all other baseline approaches in minimizing both content access latency and traffic cost with different percentage of cache capacity. Even with quite small cache space (2%), MADRL can reduce 73%, 50%, 21%, 14% latency and 103%, 98%, 59%, 26% cost compared with LFU, LRU, DRL and JAL, respectively. Besides, these two metrics have a decreasing trend for all approaches when we increase the edge capacity since larger storage space enables edge nodes to cache more content simultaneously so that more video requests can be satisfied by local edge or neighbor edges.

From Fig. 9(c) we can find that our approach has about 13% and 7% higher edge hit ratio than the state-of-the-art DRL and JAL based approach respectively, not to mention the simple rule-based approach LFU and LRU. This is because DRL only considers maximizing the individual performance for every single edge without well-planned cooperation and JAL may not well capture the historical request patterns using a stateless architecture. As a comparison, MADRL is able to fully mine the time-sequential request features from the historical states and jointly consider both its local situation and the neighbors' situations, achieving a higher edge hit ratio. As to Fig. 9(d), we find that DRL has the lowest neighbor hit ratio which is below 10%. Comparing the edge hit situation and neighbor hit situation of MADRL and DRL we can find that MADRL sacrifices some local hit and spares a portion of cache space to serve its neighbors. This tradeoff actually benefits more edges since CDN fetching can be largely avoided, leading to an average lower latency and traffic cost. Besides, note that the neighbor hit ratio of MADRL and JAL decreases as the capacity increase. This is because when cache space is large enough, most requests can be served locally and the portion of neighbor hit is naturally reduced.

We next change the number of neighbors for each edge to evaluate the performance of different approaches. The neighbors are selected from near to far for each edge (if there are multiple neighbors with the same distance, we randomly select the specific number from them). From Fig. 10(a) and Fig. 10(b) we can find that MADRL outperforms other approaches even when there are very few neighbors. When the number of neighbors increases, the reduction of average latency and cost are marginally declined. For example, the latency reduction is 500% when increasing the number of

neighbors from 2 to 4 than from 10 to 12. This means the cache cooperation among several nearby neighbors is usually enough and considering too many neighbors is not so cost-efficient. Similarly, the edge hit ratio and neighbor hit ratio also reveal the same characteristics from Fig. 10(c) and Fig. 10(d). In average, our MADRL based approach is able to achieve about 14% and 8% higher edge hit compared with DRL and JAL, respectively.

VI. Related Work

A. Caching at Network Edge

Mobile edge caching [2] emerges in recent years as a new caching paradigm that enables the content fetching at the much closer network edge, particularly the base stations or APs, so as to reduce the traffic cost as well as content access latency [3], [25]. Many existing approaches propose specific models according to different caching situations. They either rely on dedicated models with the content popularity assumed to be known [6], [7], [26] or propose forecasting models to predict the general content popularity based on such features as historical patterns [27] and social patterns [28], [29]. These approaches however are not adaptive enough to well fit the highly dynamic and heterogeneous environment in edge caching. Recently, many learning-based approaches [23], [30], [31] have also been proposed to improve the caching performance in edge, while they are mostly confined to centralized learning within one node that is not scalable enough in edge context. Jiang et al. [24] models the D2D caching problem as a multi-agent multi-armed bandit problem and relies on Q-learning to learn a coordinated caching scheme among multiple agents, while the stateless architecture may not fully capture the high dynamics among different edges towards a well adaptive strategy. Different from existing approaches, our MacoCache framework leverages multi-agent deep reinforcement learning to well adapt the massively distributed dynamics and diversities and achieve collaborative intelligence for edge caching.

B. Deep Reinforcement Learning

In recent years, deep reinforcement learning has been widely explored in many research fields given its powerful learning capacity from the past experience. For example, Mnih et al. [16] applies deep Q-learning in the popular Atari games, which achieves excellent performance even beyond the average human level. Mao et al. [14] develops a DRL based system that is able to automatically select proper bitrate purely based on the historical experience without any predefined rules. Pang et al. [32] propose a multimodal DRL based approach to optimize video quality selection in 360-degree video streaming. Wang et al. [33] applied DRL for viewer scheduling and transcoding selection in crowdsourced livecast streaming. The integration of DRL and multi-agent learning, referred to as multi-agent deep reinforcement learning (MADRL), further expands the learning capacity to more complicated situations that need cooperation or competition [34], such as spectrum sharing in vehicular network [35] and traffic signal control [36]. In this paper, we propose a MADRL based framework that is carefully designed to afford the massively vibrant, diversified and distributed caching environment to minimize both the video access latency and the redundant traffic cost.

VII. Further Discussion

Active content set selection. Our MacoCache framework does not conduct caching selection through the whole video set since there are numerous video numbers in practical scenarios and more than 80% videos are only accessed less than 5 times from our trace. We therefore focus more on those active videos, which is also a common processing method in practical caching service. Besides, how to forecast the popularity of video content, especially for the near future time period, has been widely explored with good performance in recent researches [27], [29], which is not our focus in this paper.

Online updating. Our MacoCache framework supports online updating and the selected set of popular content can also be updated over time. According to the practical video watching situation, we can define a content updating period, e.g. several hours or one day, where the active contents for caching are selected at each updating period. Based on it, the learning model can be updated periodically with the latest content request data. It is worth noting that the updating process and the current prediction process are independent and the well-trained model can be applied to the next period iteratively.

Scalability. Though we select a certain number of edges for evaluation, our MacoCache framework is scalable to support large-scale edge node for collaborative caching. In practical deployment, each edge runs its own learning agent and shares information among its neighbors, which renders much better scalability compared to those caching approaches with a centralized scheduler.

VIII. Conclusion

In this paper, we propose MacoCache, a novel edge caching framework that leverages multi-agent deep reinforcement learning (DRL) to better boost the edge caching performance so as to minimize both the content access latency and the traffic cost. Through our trace analysis on a real-world video watching dataset, we demonstrate that both the content access patterns and the content similarities among different edge areas are highly heterogeneous and dynamic across both temporal and geographical dimensions. Inspired by the recent advances in deep reinforcement learning and multi-agent learning, we therefore propose a MADRL based approach to well adapt the massively distributed dynamics in the edge caching context and achieve collaborative intelligence for edge caching. The trace-driven evaluation further demonstrates the superiority of our approach compared with the state-of-the-art solutions.

Acknowledgement

This work is supported by a Canada Technology Demonstration Program (TDP) grant and a Canada NSERC Discovery Grant. The work of Lifeng Sun is supported by NSFC 61936011, Beijing Key Laboratory of Networked Multimedia. The corresponding author is Jiangchuan Liu.

REFERENCES

[1] "Cisco visual networking index: Forecast and trends, 20172022 white paper." https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[3] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[4] S. Podlipnig and L. Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, 2003.

[5] Q. Huang, K. Birman, R. van Renesse, W. Lloyd, S. Kumar, and H. C. Li, "An analysis of facebook photo caching," in *Proceedings of SOSP*, pp. 167–181, ACM, 2013.

[6] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *Proceedings of INFOCOM*, pp. 756–764, IEEE, 2015.

[7] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[8] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proceedings of INFOCOM*, pp. 854–862, IEEE, 2013.

[9] T. X. Tran and D. Pompili, "Octopus: A cooperative hierarchical caching strategy for cloud radio access networks," in *Proceedings of MASS*, pp. 154–162, IEEE, 2016.

[10] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1863–1876, 2015.

[11] T. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, 2018.

[12] F. Wang, F. Wang, X. Ma, and J. Liu, "Demystifying the crowd intelligence in last mile parcel delivery for smart cities," *IEEE Network*, vol. 33, no. 2, pp. 23–29, 2019.

[13] X. Fan, F. Wang, F. Wang, W. Gong, and J. Liu, "When rfid meets deep learning: Exploring cognitive intelligence for activity identification," *IEEE Wireless Communications*, p. 2, 2019.

[14] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of SIGCOMM*, pp. 197–210, ACM, 2017.

[15] J. Bartelt, P. Rost, D. Wubben, J. Lessmann, B. Melis, and G. Fettweis, "Fronthaul and backhaul requirements of flexibly centralized radio access networks," *IEEE Wireless Communications*, vol. 22, no. 5, pp. 105–111, 2015.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[17] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning.," in *Proceedings of AAAI*, vol. 2, p. 5, Phoenix, AZ, 2016.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *CoRR*, vol. abs/1509.02971, 2015.

[19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of ICML*, pp. 1928–1937, 2016.

[20] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of ICML*, pp. 1146–1155, JMLR. org, 2017.

[21] O. Krajsa and L. Fojtova, "Rtt measurement and its dependence on the real geographical distance," in *Telecommunications and Signal Processing*, pp. 231–234, IEEE, 2011.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[23] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, "Caching transient data for internet of things: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2074–2083, 2018.

[24] W. Jiang, G. Feng, S. Qin, T. S. P. Yum, and G. Cao, "Multi-agent reinforcement learning for efficient content caching in mobile d2d networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 3, pp. 1610–1622, 2019.

[25] M. Ma, Z. Wang, K. Yi, J. Liu, and L. Sun, "Joint request balancing and content aggregation in crowdsourced cdn," in *Proceedings of ICDCS*, pp. 1178–1188, IEEE, 2017.

[26] A. Khreishah and J. Chakareski, "Collaborative caching for multicell-coordinated systems," in *Proceedings of INFOCOM WKSHPS*, pp. 257–262, IEEE, 2015.

[27] D. Niu, Z. Liu, B. Li, and S. Zhao, "Demand forecast and performance prediction in peer-assisted on-demand streaming systems," in *Proceedings of INFOCOM*, pp. 421–425, IEEE, 2011.

[28] H. Li, X. Ma, F. Wang, J. Liu, and K. Xu, "On popularity prediction of videos shared in online social networks," in *Proceedings of CIKM*, pp. 169–178, ACM, 2013.

[29] J. Xu, M. Van Der Schaar, J. Liu, and H. Li, "Forecasting popularity of videos using social media," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 2, pp. 330–343, 2014.

[30] C. Zhong, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *Proceedings of CISS*, pp. 1–6, IEEE, 2018.

[31] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Network*, vol. 32, no. 6, pp. 50–57, 2018.

[32] H. Pang, C. Zhang, F. Wang, J. Liu, and L. Sun, "Towards low latency multi-viewpoint 360 interactive video: A multimodal deep reinforcement learning approach," in *Proceedings of INFOCOM*, pp. 991–999, IEEE, 2019.

[33] F. Wang, C. Zhang, J. Liu, Y. Zhu, H. Pang, L. Sun, *et al.*, "Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe," in *Proceedings of INFOCOM*, pp. 910–918, IEEE, 2019.

[34] L. Zheng, J. Yang, H. Cai, M. Zhou, W. Zhang, J. Wang, and Y. Yu, "Magent: A many-agent reinforcement learning platform for artificial collective intelligence," in *Proceedings of AAAI*, 2018.

[35] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *arXiv preprint arXiv:1905.02910*, 2019.

[36] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.