Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 6684293, 16 pages
https://doi.org/10.1155/2020/6684293

*Research Article*

# Deep Reinforcement Learning-Based Collaborative Video Caching and Transcoding in Clustered and Intelligent Edge B5G Networks

**Zheng Wan**[1] **and Yan Li**[1,2]

[1]*School of Information Management, Jiangxi University of Finance and Economics, No. 665, West Yuping Road, Nanchang, Jiangxi 330032, China*
[2]*School of Information Engineering, Nanchang Institute of Technology, No. 289, Tianxiang Road, Nanchang, Jiangxi 330099, China*

Correspondence should be addressed to Yan Li; yli@nit.edu.cn

Received 14 October 2020; Revised 7 November 2020; Accepted 25 November 2020; Published 12 December 2020

Academic Editor: Lisheng Fan

In the next-generation wireless communications system of Beyond 5G networks, video streaming services have held a surprising proportion of the whole network traffic. Furthermore, the user preference and demand towards a specific video might be different because of the heterogeneity of users' processing capabilities and the variation of network condition. Thus, it is a complicated decision problem with high-dimensional state spaces to choose appropriate quality videos according to users' actual network condition. To address this issue, in this paper, a Content Distribution Network and Cluster-based Mobile Edge Computing framework has been proposed to enhance the ability of caching and computing and promote the collaboration among edge severs. Then, we develop a novel deep reinforcement learning-based framework to automatically obtain the intracluster collaborative caching and transcoding decisions, which are executed based on video popularity, user requirement prediction, and abilities of edge servers. Simulation results demonstrate that the quality of video streaming service can be significantly improved by using the designed deep reinforcement learning-based algorithm with less backhaul consumption and processing costs.

## 1. Introduction

Beyond fifth-generation (B5G) networks is the next-generation wireless communications systems. They are desired to provide rather reliable services with super high transmission rate, ultralow latency, very little energy loss, excellent quality of experience (QoE), and much enhanced security [1]. Due to providing mobile edge computing and edge caching capabilities together with machine learning, edge intelligence is emerging as a new concept and has extremely high potential in addressing the new challenges in B5G networks [2, 3]. In wireless communication networks, video streaming services have hold a surprising proportion of the whole network traffic. In particular, because of the impact of the epidemic at 2019-nCoV in this year, it has greater dependence and demand on online video streaming services, such as online meeting, online teaching, and online shopping.

In recent years, the number of smart devices has been explosively grown, which led to unprecedented increase in the demand on video streaming service. In video streaming service, it generally requires higher data rates and bigger system capacity. The overall mobile data traffic has experienced 17-fold growth from 2012-2017 as summarized in Cisco Visual Networking Index [4]. Mobile videos account for more than half of this data traffic and are predicted to further grow by 2022, accounting for 79% of the total data traffic. Due to the immense demands of mobile videos, mobile network operators can not be enough to satisfy the users' demands on high-quality video streaming services.

To address this issue, firstly, edge video caching has been recognized as a promising solution to reduce the data traffic, because edge video caching can bring videos closer to the users, which will reduce data traffic going through the backhaul links and the time required for video delivery [5]. Motivated by serving the users better, different edge caching strategies have been studied recently. Secondly, a good video QoE is very important to users. In a full range of user mobile devices, the source video streams are needed to be transcoded into multiple representations. But the video transcoding is also an extremely computation intensive and time-consuming work [6].

Recently, mobile edge computing (MEC) has been introduced as an emerging paradigm in the edge of the cellular Radio Access Network (C-RAN) [7–12]. The MEC servers are implemented particularly at the BSs in the mobile edge computing platforms, enabling video streaming services in close-proximity to the mobile users. Due to this position, MEC presents a unique opportunity to not only perform edge caching but also implement edge processing.

Due to the heterogeneity of users' processing capabilities and the variation of network condition, the user preference and demand towards a specific video might be different. For example, users with better network condition usually prefer high-resolution videos while users with poor network condition may desire for appropriate quality videos according to their actual network condition. Based on this phenomenon, adaptive bitrate (ABR) streaming [13, 16] has been widely used to improve the quality of delivered video. In ABR streaming, the bitrate of the streaming video will be chosen according to the users' specific request and actual network condition. A video content is encoded into multiple layers with different bitrates, satisfying different users' requirement. Then, each video layer will be further segmented into many small video chunks, which contains several seconds of the video content. Thus, users can dynamically adjust video layer for different video chunks, depending on their actual network conditions. So, it is a complicated decision problem with high-dimensional state spaces to choose appropriate quality videos according to users' actual network condition. There are obvious advantages in deploying ABR streaming locally at multi-MEC servers in RAN, such as avoiding the long latency and reducing the prestorage pressure at RAN [14–18]. Then, the required video layer of mobile users can be transcoded in an on-demand fashion, which can improve ABR streaming performance over mobile edge computing networks when it is directly served from a local MEC server.

Deep learning has a strong perception ability. It is mainly used to solve classification and regression problems by capturing and analyzing data features [19–22], but it does not have the ability to make decisions. Reinforcement learning [23] has the ability to make decisions, but it is helpless to perceive problems and cannot handle high-dimensional data. Reinforcement learning is actually an agent that learns the best decision sequence during the interaction with the environment. In order to deal with the complicated control and decision problems with high-dimensional state spaces, a promising solution has been given in recent development of deep reinforcement learning (DRL) [24]. DRL consists of two modules: deep learning and reinforcement learning. It uses deep learning to extract features from complex high-dimensional data and transform it into a low-dimensional feature space. Then the low-dimensional feature state space inputs into reinforcement learning to make decisions for seeking more rewards. The goal of DRL is to enable an agent to take the best action in the current state to maximize long-term gains in the environment [25, 26]. And the interaction between the agent's action and state is learned by leveraging the deep neural network (DNN). Due to these characteristics, DRL becomes a powerful tool in robotics, wireless communication, etc. [27–29]. Since the advent of deep Q network (DQN) [30–32] in 2013, a large number of algorithms and papers to solve practical application problems have appeared in the field of deep reinforcement learning. The basic idea behind many reinforcement learning algorithms is to estimate the $Q$ value function by using the Bellman equation as an iterative update. Such value iteration algorithms converge to the optimal $Q$ value function.

This paper intends to propose a video transmission model combining MEC and Content Distribution Network (CDN) technology, which interconnects the CDN network with the MEC network through the CDN tips. Also, we focus on exploiting MEC storage and processing capabilities to improve the performance of high-quality streaming services. We aim to solve the collaborative caching and transcoding for multi-MEC servers by using the DRL algorithm in mobile edge computing system. Specifically, the main contributions of this paper are as follows:

(i) A CDN and Cluster-based Mobile Edge Computing (2C-MEC) system model has been proposed, which promotes cooperation among MEC servers and reduces unnecessary backhaul consumption and processing costs. We design a MEC-enabled collaborative caching and transcoding for multi-MEC servers in the 2C-MEC system by leveraging video caching and transcoding in the vicinity of RAN at multi-MEC servers

(ii) The optimization problem of collaborative caching and transcoding for multi-MEC servers can be formulated as a stochastic Markov decision process to maximize the time-averaged Deep Q-Network (DQN) reward. The reward is defined as the weighted sum of the cache hit rate, user perceived QoE, the cost of performing transcoding, and transmission at multi-MEC servers. Then, we develop a DRL-based algorithm to automatically obtain the intracluster collaborative caching and transcoding decisions, which are executed based on video popularity, user requirement prediction, and abilities of MEC servers

(iii) Simulation results demonstrate that video streaming service can be significantly improved by using the proposed DRL-based algorithm compared with the scheme that video transcoding is not implemented at the MEC servers, with less backhaul consumption and processing costs

The remainder of this paper is organized as follows. Section 2 presents a related work. Section 3 describes the framework design of system and Section 4 formulates the optimization problem. The DRL-based algorithm is presented in

Section 5. Section 6 presents the simulation results and analysis, followed by conclusions in Section 7.

## 2. Related Work

The research on the application of DRL to wireless network transmission optimization in the MEC environment is extensive studied recently. It can be seen that the research in this area mainly began in 2018, increasing quickly year by year after 2018. Furthermore, the application of DRL in video transmission optimization under MEC environment is less. The current research in this area includes the following categories: DRL-based caching strategy, DRL-based real-time transcoding scheduling decision, DRL-based wireless network communication resource allocation [33–37], and DRL-based offloading and service migration of computing tasks [38–43]. In this paper, we mainly focus on the first two topics, trying to satisfy the requests of quality for user's streaming service.

*2.1. DRL-Based Caching Strategy.* For edge video caching at MEC servers, video caching policy is driven by video popularity. Therefore, knowing the video popularity is key to solve the video caching problem. To avoid such drawbacks, combining DRL methods are introduced to implement video cache strategies, which is an important research direction [44–47]. In order to reduce the traffic load of backhaul and transmission latency, Wei et al. [48] proposed the Q-Learning-based collaborative cache algorithm to solve the intelligent baseband unit pool cache problem. Yang et al. [49] considered the task offloading decision, cache allocation, and computation allocation problems in single MEC sever; a DRL algorithm was proposed to solve this optimization problem with low complexity. Zhong et al. [50, 51] presented a DRL-based framework with Wolpertinger architecture for content caching at the single MEC. They proposed deep actor-critic reinforcement learning-based policies for both centralized and decentralized content caching, aiming at maximizing the cache hit rate in centralized edge caching and the cache hit rate and transmission delay as performance metrics in decentralized edge caching. Gursoy et al. [52] designed a deep actor-critic RL-based multiagent framework for the edge caching problem in both a multicell network and a single-cell network with D2D communication.

Applying DRL to cache technology mainly solves the problem of cache content location decision, cache update strategy, and cache content delivery. It implements resource allocation and cache scheduling by using deep learning to analyze and learn network information. Then corresponding video content and bitrate versions are cached to improve cache hit radio and utilization of cache resources. However, the lack of transcoding on the network edge will reduce the video cache hit rate.

*2.2. DRL-Based Transcoding Scheduling Strategy.* The user's demand towards a specific video might be different because of the heterogeneity of their actual network condition. To address this issue, transcoding in network edge has been widely used to improve the quality of delivered video on the wireless networks. To achieve accurate QoE, Liu et al. [53] and Zhang et al. [54] presented deep learning-based QoE prediction called DeepQoE. Then in [53], the authors designed a content-aware bitrate adaptation policy with the objective to prefetch a higher resolution version for video clips that is in line with viewers' interests. Zhang et al. [54] also developed a DeepQoE-based ABR system to verify that their framework can be easily applied to multimedia communication service. To address the challenge of how to allocate bitrate budgets for different parts of the video with different users' interest, Gao et al. [55] proposed a content-of-interest-based rate adaptation scheme for ABR. They designed a deep learning approach for recognizing the interestingness of the video content and a DQN approach for rate adaptation according to incorporating video interestingness information. Considering joint computation and communication for ABR streaming, Guo et al. [56] presented a joint video transcoding and quality adaptation framework for ABR streaming. Inspired by recent advances of blockchain technology, Liu et al. [57] proposed a novel DRL-based transcoder selection framework for blockchain-enabled D2D transcoding systems where video transcoding has been widely adopted in live streaming services, to bridge the resolution and format gap between content producers and consumers. To accommodate personalized QoE with minimized system cost, Wang et al. [58] proposed DeepCast, which is an edge-assisted crowdcast framework. It makes intelligent decisions at edges based on the massive amount of real-time information from the network and viewers. In [59], using DRL to train a neural network model for resource provisioning, Pang et al. designed a joint resource provisioning and task scheduling approach for transcoding live streams in the cloud.

The application of DRL in transcoding scheduling decisions mainly focuses on making intelligent real-time transcoding decisions at the network edge based on a large amount of real-time information from the network and customers. In order to meet the high-quality video service experience of requirements of different users, DRL-based transcoding scheduling strategy will aim at achieving personalized QoE with minimized system cost.

*2.3. Our Vision and Motivation.* Inspired by the success of DRL in solving complicated control problems, DRL-based methods are commonly used in caching and transcoding strategy for MEC system. But there are still some issues which are needed to be resolved. (i) At present, there are many systems mainly studying single-MEC server. However, single-MEC server does not have enough storage and computing ability to satisfy the needs of different users. (ii) There are few researches on the cooperation mode and efficiency of multi-MEC servers. The completion of intensive tasks requires efficient collaboration among multi-MEC servers. (iii) In multi-MEC servers' system, the load balance among MEC servers and the resource utilization of the MEC server are basically not considered. (iv) According to users' network conditions, adaptively collaborative caching and transcoding methods in ABR streaming are needed further explored.

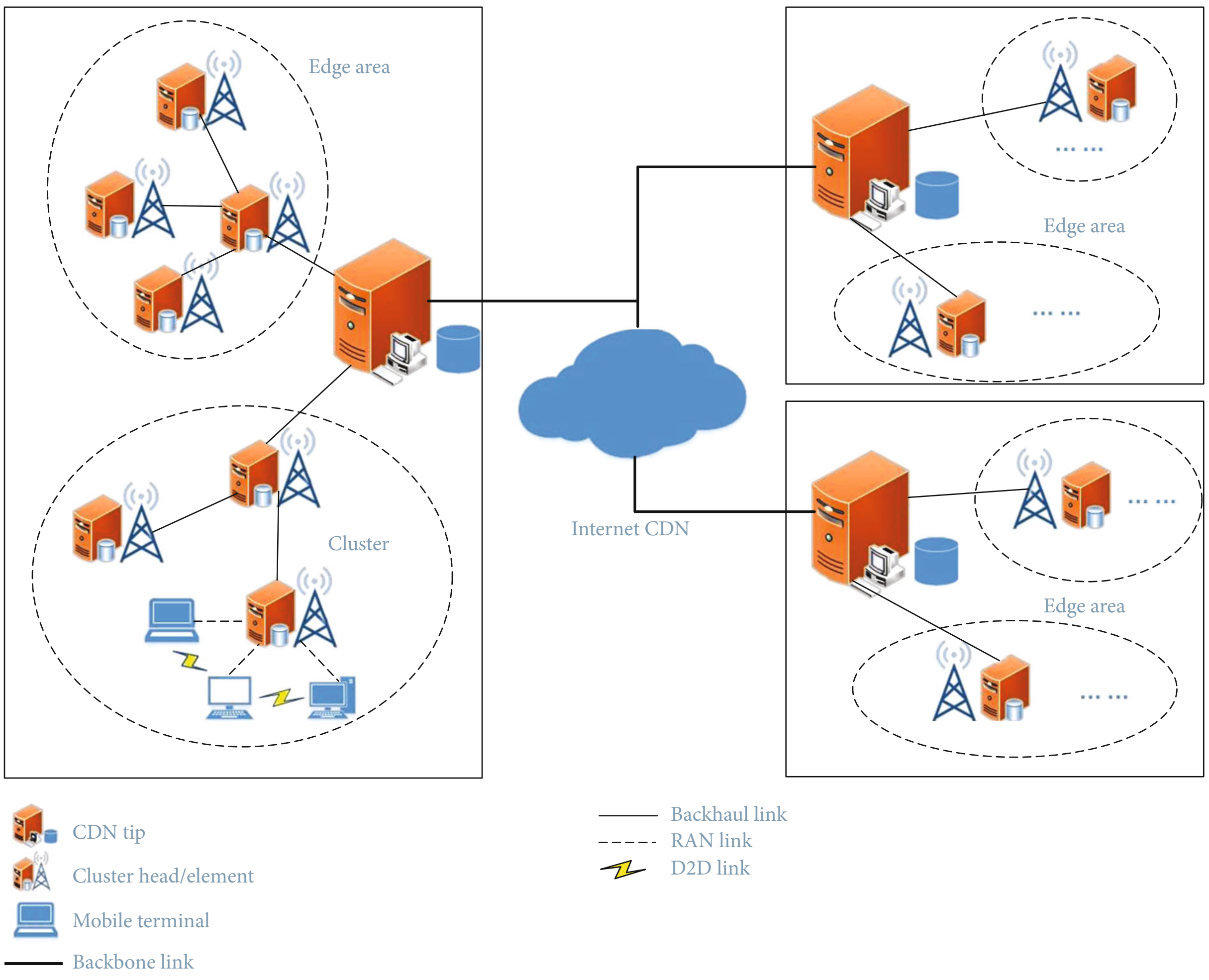

Figure 1: A CDN and Cluster-based Mobile Edge Computing (2C-MEC) system.

To address these issues, in this paper, a CDN and Cluster-based Mobile Edge Computing (2C-MEC) system model has been presented, which promotes cooperation among MEC servers and reduces unnecessary backhaul consumption and processing costs. Then, aiming to exploit MEC storage and processing capabilities to improve performance of high-quality streaming services, we focus on solving the collaborative caching and transcoding for multi-MEC servers by using the DRL algorithm in the 2C-MEC system model.

## 3. Framework Design of System

*3.1. 2C-MEC System Model.* In order to meet the transmission requirements of real video services in the internet, the video transmission strategy based on mobile edge computing must consider a heterogeneous wireless access network environment and popular video transmission technology. As shown in Figure 1, this paper intends to propose a video transmission model combining Cluster-based MEC and CDN technology, which is called as a CDN and Cluster-based Mobile Edge Computing system.

The video transmission model-based mobile edge computing is seamlessly connected with the current popular video transmission CDN technology. In this model, the edge area consists of the CDN tips (that is, the “edge node” in the CDN, in order to distinguish it from the edge computing node, called “CDN tip” in this paper) and many edge computing nodes in the local area (may be deployed at small base stations, macro base stations, and locations higher than the macro base stations). Thereby, the computing, storage, and communication capabilities of edge computing nodes are used to assist in the deployment of sparse CDN tips to optimize wireless video transmission across the entire network.

Due to the large number of edge nodes and the large difference in capabilities among them, a hierarchical management model is proposed to cluster edge nodes. The communication protocols within and among clusters can draw on the related research of sensor networks and P2P networks. The influencing factors of edge node clustering strategy include edge node capabilities, geographic location distribution, number and activity of users. The 2C-MEC system can promote mutual cooperation among MEC servers and reduces unnecessary backhaul consumption and processing costs.

Based on the proposed Cluster-based Mobile Edge Computing framework, on the one hand, the storage and computing capabilities of the MEC servers have been improved. The 2C-MEC system enables the MEC servers’ collaboration within the cluster to have sufficient storage and computing power to meet users' needs. On the other hand, the collaboration among MEC servers is promoted. Under this framework, it is possible to pursue the multi-MEC collaboration method within the cluster, which focuses on exploring the effective ways of multi-MEC servers' collaboration of caching and transcoding. On the contrary, existed studies focused on "cloud-edge" collaboration or "edge-edge" collaboration.

In this paper, we plan to design the edge node clustering algorithm based on the following ideas: (i) firstly, the cluster division is based on the principle of proximity to geographic location. (ii) Secondly, the overall service capabilities of the nodes in the cluster should match their users' needs, and the edge service capabilities among different clusters should be balanced to a certain extent. (iii) Thirdly, if the edge node is located in the intersection area of two clusters, the appropriate cluster is selected based on the similarity of the video access preferences of the users managed by this node and the video access preferences of the users managed by other nodes in one cluster. (iv) Finally, after the clustering is completed, we can comprehensively consider the computing, storage, communication capabilities of the edge node, and its communication delay with other nodes in the cluster to elect this cluster's head.

In a word, the 2C-MEC system model proposed in this paper is compatible with popular CDN technology, resulting in conveniently utilization of its research results in cache replacement, content prefetching and load balancing. Furthermore, the ability of MEC to utilize heterogeneous edge nodes with different capabilities and deployments further improves the quality of video transmission.

*3.2. Rebuffer Model.* In order to keep continuous playback in video streaming service, a playback buffer is usually deployed at the user device, in which the video chunks are downloaded into. The rebuffer model used in this paper comes from the reference [60]. Let $B(t)$ denote the bitrate of the chunk at time stage $t$ for the user. And $W(t)$ denotes the wireless transmission rate (bits/second) of user experienced during time stage $t$. Then, the buffer occupancy rate $L(t)$ is defined as follows:

$$L(t) = \frac{\text{Buffer occupancy}}{\text{Buffer size}}. \quad (1)$$

When $B(t)/W(t) < 1$, the new video chunk is put into the buffer at rate of less than 1; then, the buffer decreases. In another way, if more than one chunk is played before the next chunk arrives, then, the buffer is depleted and the rebuffering is happened. So, in the rebuffer model, the term of rebuffering time and buffered video time are usually introduced, which are used in Reference [56]. A video has some chunks; each chunk also contains a fixed duration of video, such as $D$ seconds of video. Let $T(t)$ denote the buffered video time at playback buffer at the beginning of time stage $t$. In the rebuffer model, we assume that one chunk will be downloaded into the buffer at one time. The total downloading time of one chunk during time stage $t$, denoted by $d(t)$, can be expressed as

$$d(t) = \frac{B(t) * D}{W(t)}. \quad (2)$$

Furthermore, the video rebuffering time of playback buffer during time stage $t$ is denoted as $R(t)$. Then, we can get

$$\begin{aligned} R(t) &= \max\left(d(t) - T(t), 0\right), \\ T(t+1) &= D + \max\left(T(t) - d(t), 0\right). \end{aligned} \quad (3)$$

*3.3. Video Quality Rate Model.* In video processing, Peak Signal to Noise Ratio metric (PSNR) is the de facto standard criterion to provide objective quality evaluation between the original frame and the compressed one. In the video quality evaluation, the video quality rate $q(t)$ of a video coded at rate $B(t)$ can be approximated by a logarithmic function [61] as follows:

$$q(t) = \beta \log\left(B(\text{t})\right), \quad (4)$$

where the $\beta$ value can be obtained from the video encoder during the encoding in video source. Generally, the mentioned quality rate $q(t)$ is a nondecreasing function, which means a higher bitrate may be a high definition video while a lower bitrate may be a standard definition video.

Then, let $B_u(i, t) \in \{B_1, B_2, \cdots, B_{\max}\}$ and $B_{\max}$ be the set of all video layers after video transcoding and the highest video level at the MEC servers, respectively. And $B_u(i, t)$ denotes the bitrate assigned to user $i$ at timeslot $t$.

*3.4. Cache Hit Rate Model.* In our setting, requests by all users are served by the MEC severs; all video have the same size, and there are no priorities for different users, while there are popularities for different videos. Videos popularity distribution is always the key to solve the video caching problem. Considering the changing popularities, the probability that the requests of video $v$ is defined as $Z_v$, which follows the Zipf distribution [16] as follows:

$$Z_v = \frac{v^{-\alpha}}{\sum_{v=1}^{V} v^{-\alpha}}, \quad (5)$$

where $\alpha > 0$ is the parameter of Zipf distribution which indicates the skewness degree. According to our setting, the video streaming service quality of content caching can be evaluated in terms of the cache hit rate. The cache hit rate $\text{CRH}(t)$ in $T$ requests during time stage $t$ is defined [40] as

$$\text{CRH}(t) = \frac{\sum_{i=1}^{T} 1(H_i)}{T}, \quad (6)$$

where indicator function $1(H_i)$ is defined as

$$1(H_i) = \begin{cases} 1, & H_i \in C_T, \\ 0, & H_i \notin C_T, \end{cases} \quad (7)$$

where $C_T$ represents the cache state during this period; if there is the cache of video, the request $H_i$ can hit in the cache.

*3.5. System Cost Model.* In the system cost model, most of the operational cost consists of bandwidth cost and transcoding cost in video streaming service. The fraction of other service cost is negligible comparing with the above two kinds of cost.

Then, the bandwidth cost $C_b(t)$ [62] of all MEC servers in the cluster can be obtained by the following formula:

$$\begin{aligned} C_b(t) &= \sum_{n=1}^{M} P(n,t) \cdot W(n,t), \\ W(n,t) &= \sum_{i \in U^t} B_u(i,t) \cdot I^t(i,n), \quad n \in \{0, \cdots, M\text{-}1\}, \end{aligned} \tag{8}$$

where $U^t$ and $M$ are the user group and the numbers of severs in the cluster at time stage $t$. And $I^t(i,n)$ is an indicator that represents whether user $i$ is connected to MEC server $n$ at the time stage $t$. Respectively, $P(n,t)$ and $W(n,t)$ be the unit bandwidth price and the amount of bandwidth usage in the MEC server $n$.

Beside the bandwidth cost, the video streaming service also needs to consider the transcoding cost. Based on the definition and description of video transcoding in [56, 62], the transcoding cost is closely related to the input bit-rate, target bit-rate, the video length, and the number of CPU cores needed for transcoding according to the video pricing model. Then, we define the transcoding cost incurred at time stage $t$ as

$$O(t) = \sigma * (L_{\max} - l) * T_v * N_{\text{cpu}}, l \in \{L_1, L_2, \cdots, L_{\max}\}, \tag{9}$$

where $\sigma$ is an adjustable parameter and symbols of $l$, $T_v$, and $N_{cpu}$ represent the level of input video, the video length, and the number of CPU cores required for transcoding, respectively.

In order to simplify the problem formulation, in our system cost model, the operational cost mainly consisted of bandwidth cost and transcoding cost. Since bandwidth cost and transcoding cost have different measurement units, bandwidth cost reflects the network transmission capacity, while transcoding cost reflects the computing power of the MEC node; it is not easy to unify the corresponding dimensional units. However, in the comparison of simulation experiments, only the cost of comparing different environments is required. Therefore, like the design in Reference [62], the bandwidth cost and transcoding cost can be regarded as values without a unit of measurement, and there is no need to consider the details of the unit of measurement. The operational cost can be expressed as

$$C(t) = C_b(t) + O(t). \tag{10}$$

## 4. Optimization Problem Formulation

Based on using the DRL algorithm for resource optimization in the 2C-MEC system, we describe the three basic elements of reinforcement learning. They are the state, action, and reward of the collaborative video caching and transcoding optimization problem.

*4.1. State Space.* The state at time stage $t$ is jointly determined by the four tuples, the current bandwidth cost $C_b(t)$, the current buffer occupancy rate $L(t)$, the current rebuffer time $R(t)$, and the current video quality $q(t)$. Then, the state space $S(t)$ at time stage $t$ can be defined as follows:

$$S(t) = \{C_b(t), L(t), R(t), q(t)\}, \tag{11}$$

where the state space is denoted as $S$.

*4.2. Action Space.* The control action for the agent is to select the video caching strategy and video transcoding strategy for the next requested video chunk according to the current system state. In this network, the action at each time stage $t$ is the joint video cache updating, cache$(M(t), U(t))$, and video transcoding layer adaption decision, $B_u(i,t)$.

So, the action is selected from the action set $A(t)$, in which $M(t)$, $U(t)$, and $B_u(i,t)$ represent the number of MEC severs selected in the cluster, the decision of video cache updating, and the target video layer, respectively. Then, the action space can be described as

$$A(t) = \{M(t), U(t), B_u(i,t)\}, \tag{12}$$

where the action space is denoted as $A$.

In practice, since the numbers of MEC severs in a cluster and the set of all video layers are not large; also, the decision of video cache updating is only yes or no; the number of possible actions in the state space set for the collaborative video caching and transcoding problem can be not very large.

*4.3. Reward.* The reward should reflect the objective of the framework, which, in our case, is to reduce the operational cost and desire best QoE for users by solving the collaborative caching and transcoding for multi-MEC servers. In our paper, we define the reward function during time stage $t$, denoted by $r(t)$, as follows:

$$\begin{aligned} r(t) = \omega_1 \text{CRH}_{sl}(t) + \lambda q(t) - \omega_2 \|q(t) - q(t-1)\| - \omega_3 R(t) \\ - \omega_4 C_b(t) - \omega_5 O(t). \end{aligned} \tag{13}$$

The first term on the right-hand side of (13) is the weighted sum of the short and long-term cache hit rate. Considering the number of requests for local video in the next epoch, the short-term cache hit rate $\text{CRH}_s(t)$ can be either 0 or 1. Thus, let the total normalized number of requests for local video within the last 20 requests as the long-term cache hit rate $\text{CRH}_l(t) \in [0,1]$. The total cache hit rate $\text{CRH}_{sl}(t)$ for each step is defined as the weighted sum of the short and long-term cache hit rate, which is defined as

$$\text{CRH}_{sl}(t) = \text{CRH}_s(t) + \mu * \text{CRH}_l(t), \tag{14}$$

where $\mu$ is the weight to balance the short and long-term cache hit rate.

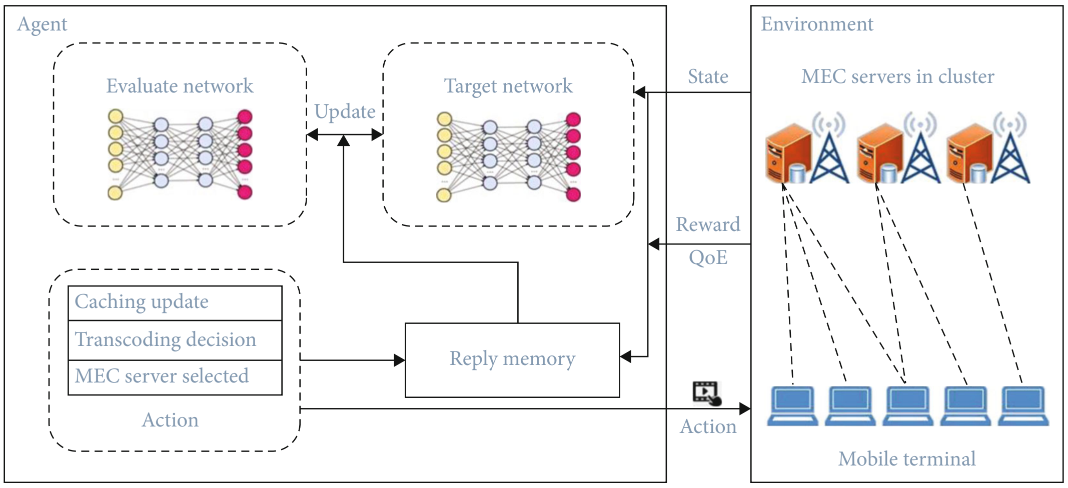

Figure 2: The design of DRL-based intracluster collaborative caching and transcoding framework.

The second, third, and fourth terms on the right-hand side of (13) are video quality, video quality variation, and video rebuffering time, respectively. The fifth and sixth terms on the right-hand side of (13) are two penalty terms for the bandwidth cost and transcoding cost in each step. The total cache hit rate, video quality, video quality variation, and video playback rebuffering time are directly associated with user perceived QoE in the video streaming service. And the weights $\omega_1, \lambda, \omega_2, \omega_3, \omega_4, \omega_5$ are the weighting parameters.

*4.4. Problem Formulation.* In this paper, our objective is to derive the jointly optimal video caching policy and video transcoding policy for maximizing the rewards in video streaming service. Future rewards and present rewards have different importance and weights because of the uncertainly of system dynamics. The objective of the joint video caching policy and video transcoding policy is to maximize the expected average reward. Then, we can formulate the dynamic optimization problem as a Markov decision process (MDP) as follows:

$$\begin{aligned} \max_{M(t),U(t),B_u(i,t)} \quad & J(t) = E\left[\sum_{t=0}^{T-1} \gamma^t r(t)\right] \\ \text{s.t.} \quad & C1: M(t) \in \{0, 1, \cdots, M\}, \quad \forall t \\ & C2: U(t) \in \{0, 1\}, \quad \forall t \\ & C3: B_u(i,t) \in \{B_1, B_2, \cdots, B_{\max}\}, \quad \forall t, \end{aligned} \tag{15}$$

where $\gamma \in (0, 1]$ is the discount factor.

It is impractical for the optimization problem with a large number of states in state space. But the DRL algorithm has been proved a useful mathematical tool for large-scale optimization problem which does not need any prior knowledge of state transition probability. Based on this, we propose a DRL-based algorithm to solve the optimization problem in formulation (15). Thus, the design of DRL-based intracluster collaborative caching and transcoding framework is shown in Figure 2.

## 5. DRL-Based Intracluster Collaborative Caching and Transcoding Algorithm

*5.1. Deep Reinforcement Learning-Based Collaborative Video Caching and Transcoding for Multi-MEC Servers.* Based on DQN's excellent performance when dealing with discrete state space and action space, we adopt DQN for learning the intracluster collaborative caching and transcoding policy. Specifically, as illustrated in Figure 2, the inputs of the deep neural network are the video service system states listed in Equation (11), and the outputs of the network are the $Q$ value function, $Q(s, a; \theta)$, for each action listed in Equation (12).

We illustrate the details of the DRL-based learning algorithm for collaborative caching and transcoding for multi-MEC servers in Algorithm 1.

## 6. Simulation Results and Analysis

In this section, firstly, we illustrate the experiment settings. Then, the computer simulations are carried out to demonstrate the performance of the proposed DRL algorithm of collaborative caching and transcoding for multi-MEC servers in mobile edge computing wireless networks.

### *6.1. Experimental Settings*

*6.1.1. Data Generation.* In our experiments, the user data of requests is generated randomly, while the video data of users' requests is generated according to the Zipf distribution. We have collected different numbers of requests in one episode as the testing data, such as 30, 40, and 50. In order to make the experiment more comprehensive, we generate two types of data sets. Firstly, the video data set in users' different-number requests was generated with unchanged popularity distribution with Zipf parameter set as 1.3. Then, the video data set in users' same-number requests was generated with a varying Zipf parameter.

```
1: Initialization:
2:    Initialize replay memory D to capacity $N$
3:    Initialize Q network and target Q network with random weights
4:    Initialize MEC service matrix $V$ of requests
5: **for** episode =1, $M$ **do**
6:    Generate the user requests data
7:    Observe the initial state $s_1$ as illustrated in Eq. (11)
8:    **for** t =1, $T$ **do**
9:       Give a random probability $\varsigma \in [0, 1]$
10:        Choose action $A(t)$ which listed in Eq. (12) as $A(t) = \begin{cases} a^*(t) = \arg \max_a Q(s, a; \theta), \varsigma > \varepsilon \\ a(t) \neq a^*(t), \textit{randomly select } a(t), \textit{others} \end{cases}$
11:        Based the action $A(t)$, execute the transcoding policy and the caching updated
12:        Observe the reward $r(t)$, state $s(t+1)$
13:        Store the transition $(s(t), A(t), r(t), s(t+1))$ in D
14:        Update MEC service matrix $V$ of requests
15:        Sample random minibatch of transitions
             $(s(t), A(t), r(t), s(t+1))$ from D
16:        Set $y_j = \begin{cases} r_j & \text{for terminal } s' \\ r_j + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a) & \text{for non-terminal } s' \end{cases}$
17:        Perform a gradient descent step according to equation: $\begin{matrix} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot)}[(y_i - Q(s, a; \theta_i))^2] \\ y_i = r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a) \end{matrix}.$
18:        Update the parameters in the Q network
19:        Reset the parameters in the target Q network every G time stages
20:    end for
21: end for
```

Algorithm 1: Deep reinforcement learning algorithm for collaborative video caching and transcoding (DRL-CCT).

*6.1.2. Parameter Setting.* In our experiments, we set 7 MEC severs in one cluster, which serve 30 users in this region and provide about 50 videos for users' requests. Then, we set $D = 10s$, $\beta = 6.5$, $\alpha = 1.3$, $\mu = 0.6$, $\sigma = 1.2$, the weights associated with cache hit rate and QoE in the reward function are set as $\omega_1 = 1$, $\lambda = 0.9$, $\omega_2 = 0.9$, $\omega_3 = 0.1$, and the weights associated with cost penalty in the reward function are set as $\omega_4 = 0.1$, $\omega_5 = 0.1$.

In the experiment, there are four video layers of the video, with $B_{\max} = 10$ Mbps as the highest layer at the MEC server. The bitrates of the three transcoded layers are $B_1 = 1$ Mbps, $B_2 = 2$ Mbps, and $B_3 = 4$ Mbps, and the set of available CPU cores at MEC is $\{2, 4, 6, 8\}$. Video transcoding from $B_{\max}$ to $B_1$, $B_2$, and $B_3$ needs 2, 4, and 6 CPU cycles, respectively. With the number of caching strategy being 2 (yes or no), the number of videos' bitrates being 4, and the number of MEC severs in one cluster being 7, the number of actions in action set $A$ is $2 \times 4 \times 7 = 56$.

*6.1.3. Deep Neural Network for DQN.* We use a fully connected neural network with 2 hidden layers, 256 and 512 in size. The loss function is the mean square error. The naive $\varepsilon$ -greedy strategy is used for exploration, and the probability of randomly choosing an action during training is $\varepsilon$. As the learning progresses, the degree of exploration continues to shrink. The learning rate is 0.01, the size of experience replay in DQN is 2000, the attenuation parameter used to update the target Q network is 0.9, and the batch size in stochastic batch gradient descent is 32. The experiments are implemented using Python and TensorFlow.

*6.2. Simulation Results.* In this section, we compare the proposed DRL algorithm (called DRL-CCT) with the latest baseline methods, such as the method (called caching only at network edge) in Reference [51] and the method (called transcoding only at network edge) in Reference [56]. In our experimental framework, we simulated the above methods according to the setting form of the reward function in the above literature. Also, we compare the proposed DRL algorithm with the algorithm of DRL-CCT without transcoding policy. Especially due to the characteristics of deep reinforcement learning, for our proposed algorithm, all reported results were obtained from average of 20 algorithm executions.

Figure 3 shows the convergence performance of the DRL-CCT algorithm under the set of full weight in the different learning rates. With continuous learning, the average reward gradually stabilizes. Compared with the balanced method of the algorithm in Reference [56], the average reward of the algorithm we proposed converges faster, and the subsequent fluctuations are slightly larger. But in contrast, the deep network used in our DRL-CCT algorithm is more concise and efficient. The convergence performance is influenced by learning rate. The performance of the learning rate 0.01 is better than the performance of the learning rates 0.1, 0.001, and 0.0001. The convergence performance becomes worse

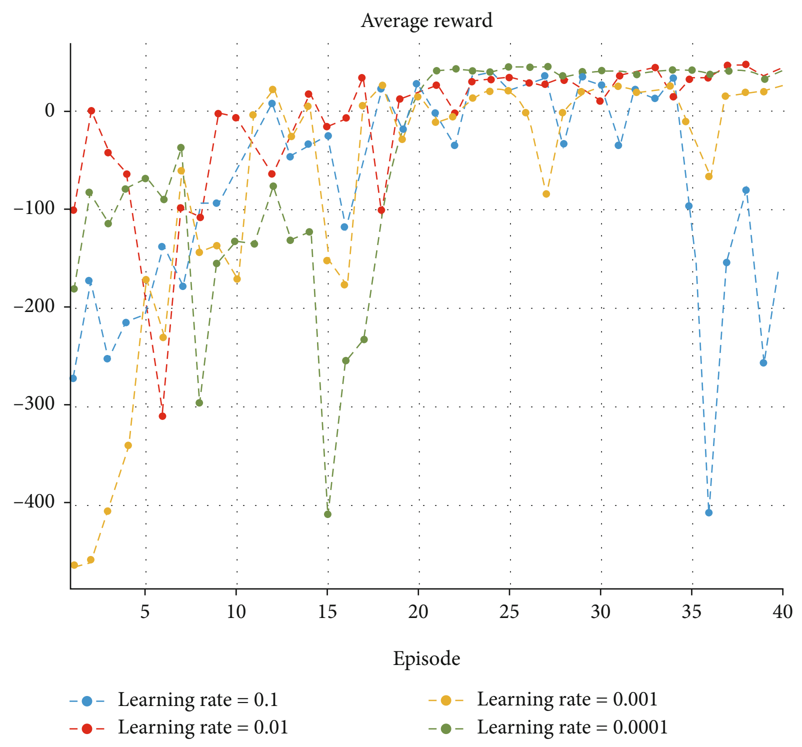

FIGURE 3: The convergence performance of DRL-CCT algorithm in the different learning rate.

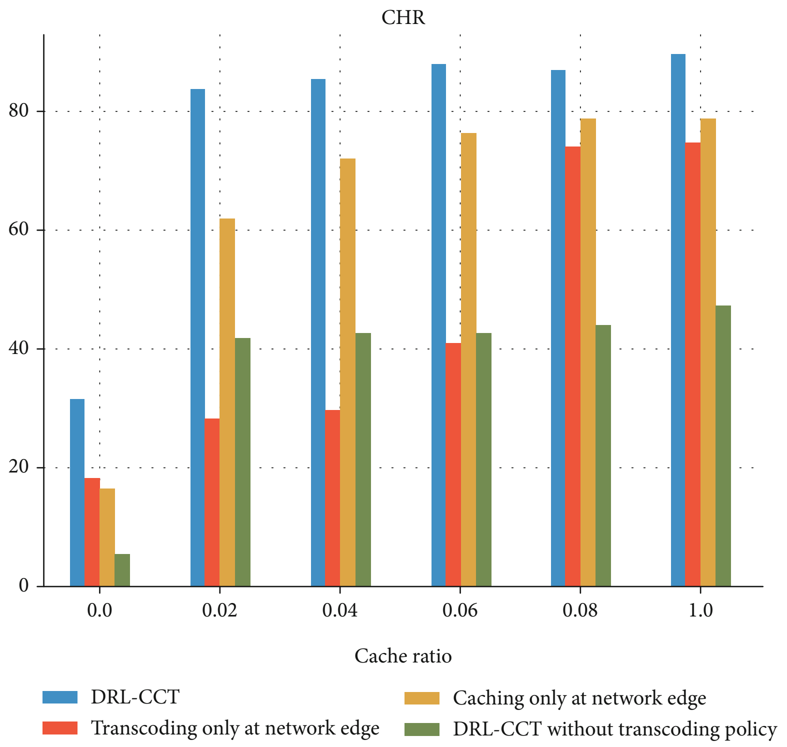

FIGURE 4: Cache hit rate vs. cache ratio.

in the learning rate 0.1, owing to a large update step such that the average reward converges to a local optimal solution. In fact, an appropriate learning rate depends on the state of the environment in the current optimization process.

Figure 4 gives the comparison of cache hit rates in different algorithms at the same cache ratios. Compared with the other algorithms, the DRL-CCT algorithm has a higher cache hit rate. Since the 2C-MEC system model has been proposed,

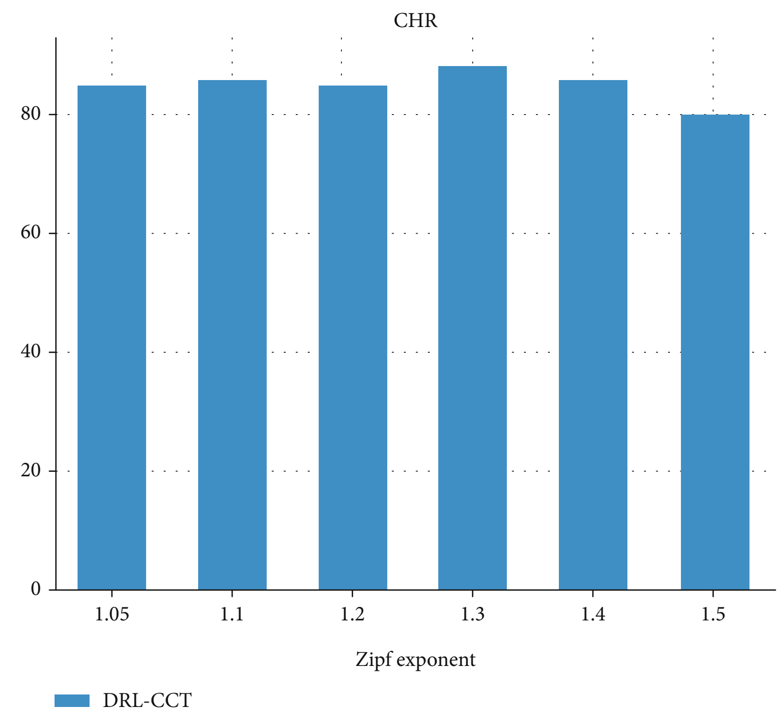

Figure 5: Cache hit rate vs. Zipf exponent.

the cluster-based video cache hit rate is definitely better than the video cache hit rate based on a single MEC server, especially when the cache ratio is relatively small. In addition, the performance in the cache hit rate of DRL-CCT without transcoding policy algorithm is the worst one because only the highest version of the video is cached in the MEC. Owing to the absence of transcoding function at network edge, the MEC server has to return to the source server for extraction when the user requests for other version of the video, which results in low cache hit rate.

In Figure 5, we study the cache hit rate as a function of the Zipf exponent. As Zipf exponent increases, cache hit rates achieved by the caching policy increase first and then decrease. This is due to the fact that with larger Zipf exponent, the video popularity distribution is more concentrated, and therefore, the popularity of the files is skewed. Consequently, caching these more popular videos leads to an increase first in the cache hit rates. Then, the cache hit rates have a fall. It is because that the DRL-CCT algorithm stores the most popular files initially when the number of popular files gets small. However, it eventually experiences diminishing returns as Zipf exponent is further increased, and the larger the Zipf exponent, the smaller the influence of less popular files is.

As for average QoE performance in Figure 6, DRL-CCT is much better than “transcoding only” and the other two algorithms. Due to the long rebuffering time, the average QoE value of the DRL-CCT without transcoding algorithm and “caching only” algorithm are below zero all the time. Compared with these methods which has no joint caching and transcoding at the edge, DRL-CCT has the highest QoE, which means users can get much better experience in video streaming services. It can be seen from Figure 7 that when there is no transcoding function at the network edge, the bandwidth cost is greater than the DRL-CCT algorithm, because the uncached video has to be extracted from the source server which leads to consume a lot of bandwidth cost. The difference of bandwidth cost performance between “transcoding only” algorithm and DRL-CCT algorithm is slight in the latter stage.

The average bandwidth cost and QoE performance in DRL-CCT algorithm with different experimental settings are shown in Figures 8–11. Figures 8 and 9 are the performance for different request numbers in an episode. It can be seen from Figure 8 that as the number of user requests in a time slot increases, the average bandwidth cost of each MEC will continue to increase. This is because the number of MEC servers is fixed. When the number of user requests has increased, the number of user requests served by each MEC must increase, which directly leads to an increase in the average bandwidth cost of each MEC. The following conclusions can be directly obtained in Figure 9 that the change in the number of requests from different users in a time slot does not have a great impact on the average QoE of the users, and the QoE value of the video streaming service is stable in a good range.

Then, Figures 10 and 11 are the performance for different MEC numbers within a cluster at network edge. According to Figure 10, on the premise that the number of user requests in a time slot is determined, when the number of MEC nodes in the edge cluster decreases, the average bandwidth cost of each MEC will increase at the beginning. However, as the deep reinforcement learning process progresses, the average bandwidth cost of each MEC will tend to stabilize. This is due to

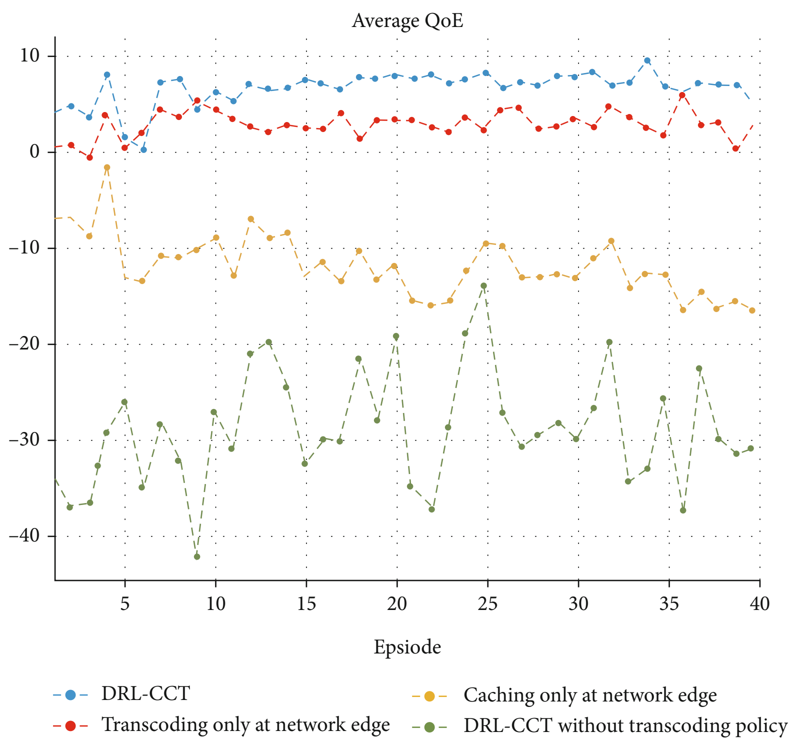

Figure 6: The QoE performance in different algorithms.

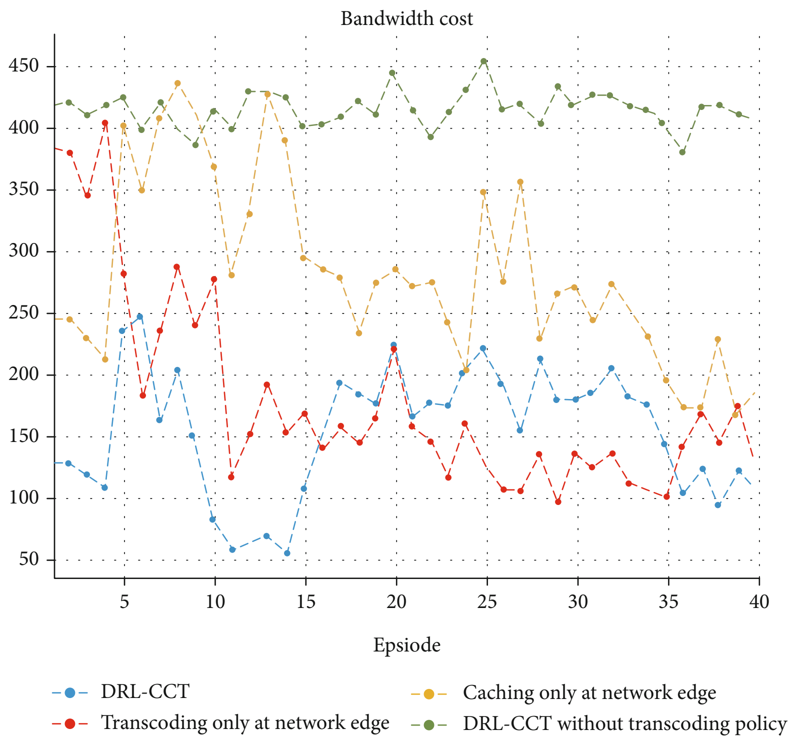

Figure 7: The bandwidth cost performance in different algorithms.

the adaptive decision-making function of deep reinforcement learning, which continuously optimizes the MEC load distribution in one edge cluster. In Figure 11, the same as in Figure 9, the average QoE performance of the system has always been relatively stable, indicating that the proposed method has excellent robustness to environmental changes.

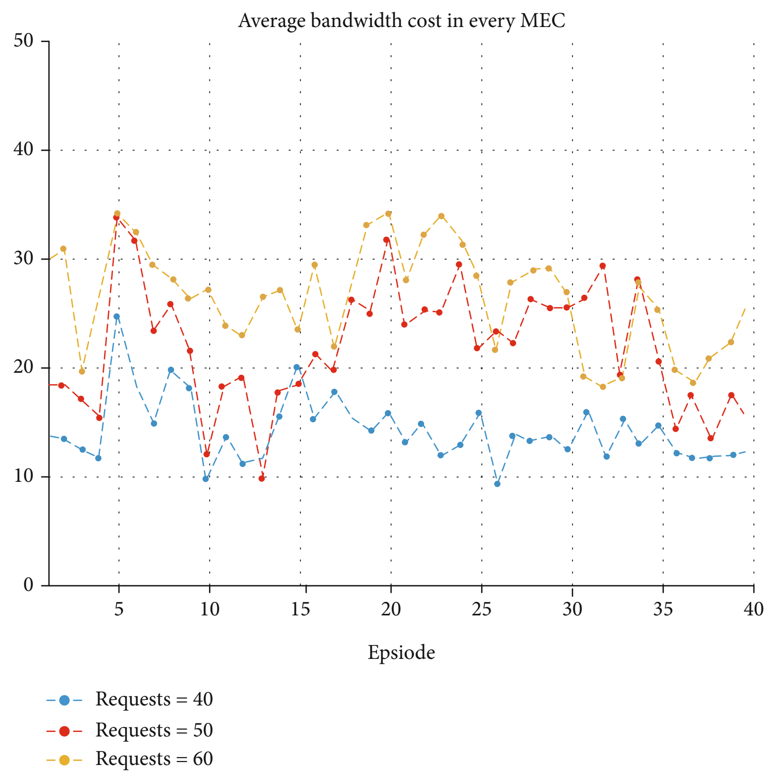

FIGURE 8: The average bandwidth cost in the DRL-CCT algorithm at different request numbers in an episode.

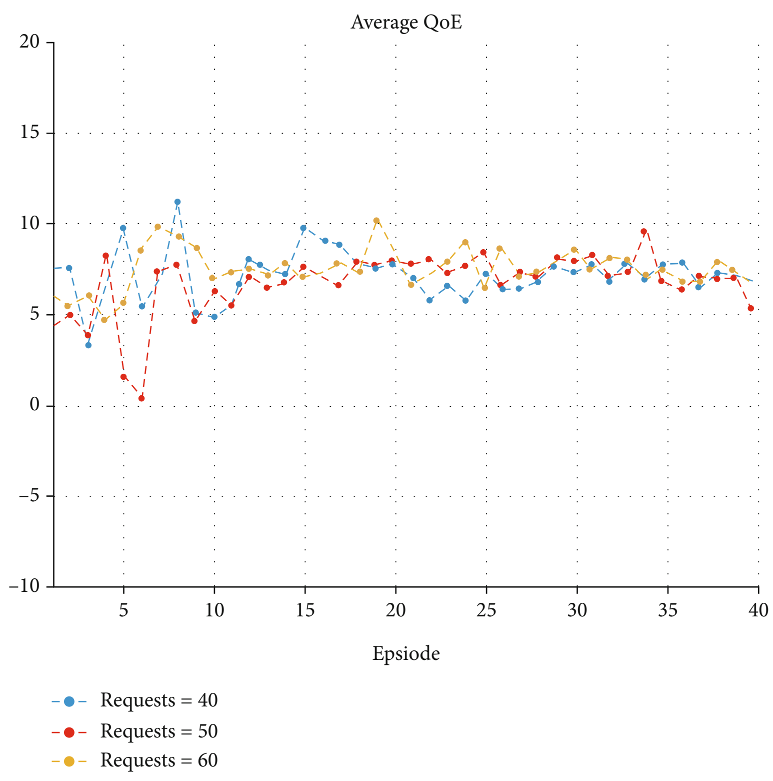

FIGURE 9: The average QoE performance in the DRL-CCT algorithm at different requests numbers in an episode.

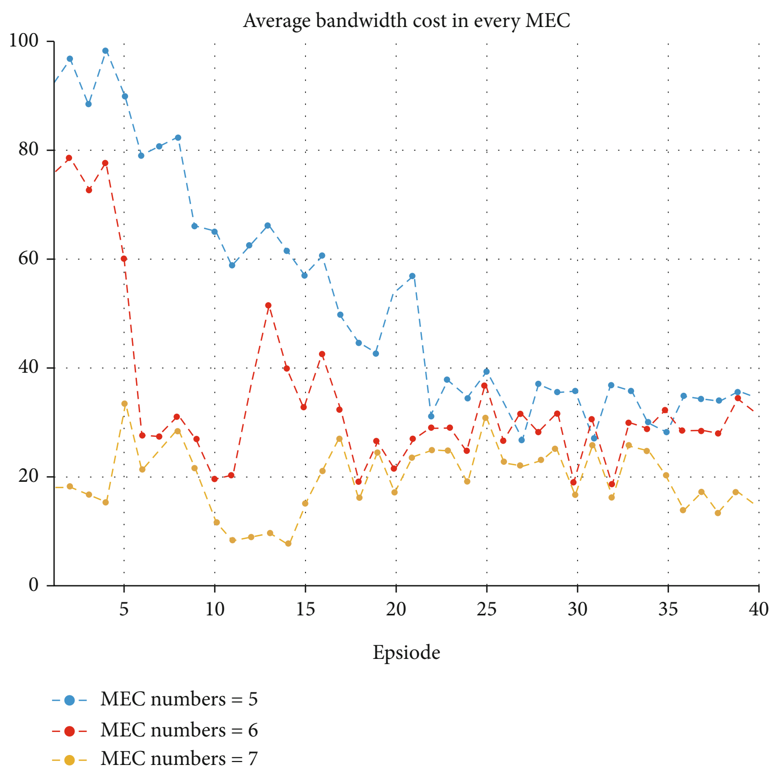

Figure 10: The average bandwidth cost in DRL-CCT algorithm at different MEC numbers with in a cluster.

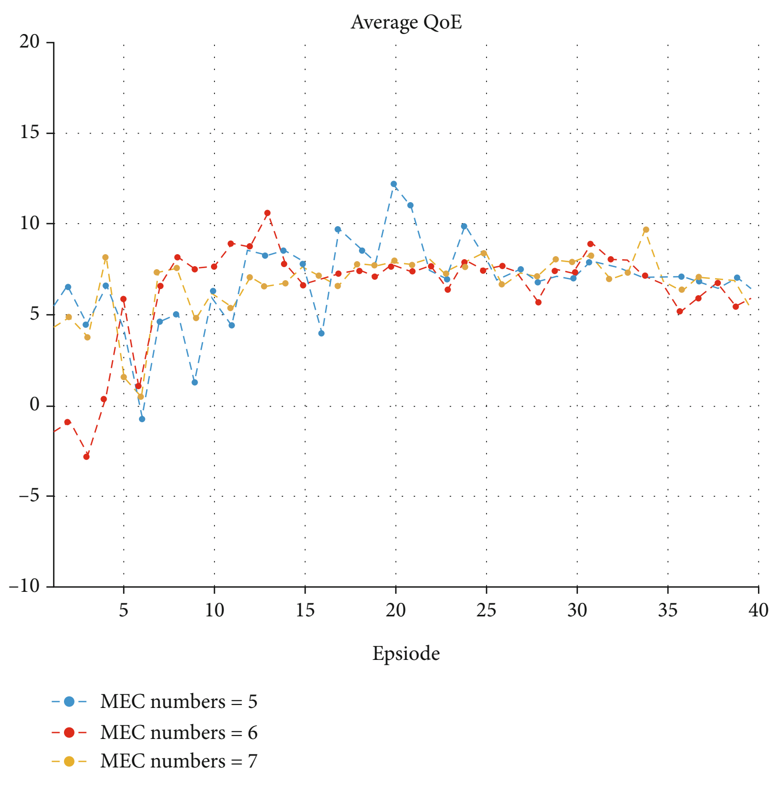

Figure 11: The average QoE performance in DRL-CCT algorithm at different MEC numbers with in a cluster.

## 7. Conclusions

In this paper, we first propose a CDN and Cluster-based Mobile Edge Computing system that can enhance the ability of caching and computing and promote the collaboration among MEC severs in one cluster. In addition, we formulate a novel deep reinforcement learning based framework to automatically obtain the intracluster collaborative caching and transcoding decisions, which are executed based on video popularity, user requirement prediction, and abilities of MEC servers. Then, numerical results are presented to validate the effectiveness of the proposed method.

Under the framework of the 2C-MEC system, this paper mainly researches on promoting the collaboration among MEC servers in the cluster. In the future work, intercluster collaboration needs to be considered when intracluster computing and storage capabilities are insufficient. If it is assumed that the terminal has caching and computing capabilities, it is also possible to consider "edge-end" collaboration, "end-end" collaboration, and other collaboration modes to implement a multidimensional collaboration model of "cloud-edge-end" among different agents. At the same time, load balancing among MEC servers in the mobile edge cluster still needs further research to explore efficient ways to solve the contradiction between the balance of MEC servers and the improvement of user QoE.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

The work was supported by the National Natural Science Foundation of China (No. 61961021), the Science and Technology Project of Jiangxi Education Department (No. GJJ180251 and No. GJJ171011), and the Innovation Special Fund for Individual Graduate Student of Jiangxi University of Finance and Economics (2020 Annual, No. 24).

## References

[1] N. Kato, B. Mao, F. Tang, Y. Kawamoto, and J. Liu, "Ten challenges in advancing machine learning technologies toward 6G," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 96–103, 2020.

[2] K. Zhang, Y. Zhu, S. Maharjan, and Y. Zhang, "Edge intelligence and blockchain empowered 5G beyond for the industrial Internet of things," *IEEE Network Magazine*, vol. 33, no. 5, pp. 12–19, 2019.

[3] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for internet of vehicles," *IEEE Wireless Communications Magazine*, vol. 26, no. 3, pp. 12–18, 2019.

[4] Cisco, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017-2022 White Paper*, 2019, https://www.cisco.com/c/dam/m/en_in/innovation/enterprise/assets/mobile-white-paper-c11-520862.pdf.

[5] S. M. Azimi, O. Simeone, A. Sengupta, and R. Tandon, "Online edge caching and wireless delivery in fog-aided networks with dynamic content popularity," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1189–1202, 2018.

[6] G. Gao, Y. Wen, and J. Cai, "vcache: supporting cost-efficient adaptive bitrate streaming," *IEEE Multimedia*, vol. 24, no. 3, pp. 19–27, 2017.

[7] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing-a key technology towards 5G," *ETSI White Paper*, vol. 11, 2015.

[8] K. Zhang, Y. Mao, S. Leng et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[9] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in *Proc. IEEE Int. Conf. On Intelligent Systems and Control (ISCO)*, Nanjing, China, 2016.

[10] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, 2016.

[11] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[12] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, "Collaborative mobile edge computing in 5g networks: new paradigms, scenarios, and challenges," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, 2017.

[13] Y. Sánchez dela Fuente, T. Schierl, C. Hellge et al., "iDASH: improved dynamic adaptive streaming over HTTP using scalable video coding," *Proceeding of ACM Multimedia Systems*, pp. 23–25, 2011.

[14] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Transactions on Mobile Computing*, vol. 18, no. 4, pp. 787–800, 2019.

[15] D. Wang, Y. Peng, X. Ma et al., "Adaptive wireless video streaming based on edge computing: opportunities and approaches," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 685–697, 2019.

[16] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, 2019.

[17] J. Yao, T. Han, and N. Ansari, "On mobile edge caching," *IEEE Communications Surveys & Tutorials*, vol. 21, 2019.

[18] S. Safavat, N. N. Sapavath Naveen, and D. B. Rawat, "Recent advances in mobile edge computing and content caching," *Digital Communications and Networks*, vol. 6, 2020.

[19] K. He, Z. Wang, W. Huang, D. Deng, J. Xia, and L. Fan, "Generic deep learning-based linear detectors for MIMO systems over correlated noise environments," *IEEE Access*, vol. 8, pp. 29922–29929, 2020.

[20] J. Xia, L. Fan, W. Xu et al., "Secure cache-aided multi-relay networks in the presence of multiple eavesdroppers," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7672–7685, 2019.

[21] H. Liu, C. Lin, J. Cui, L. Fan, X. Xie, and B. F. Spencer, "Detection and localization of rebar in concrete by deep learning

using ground penetrating radar," *Automation in Construction*, vol. 118, 2020.

[22] K. He, Z. Wang, D. Li, F. Zhu, and L. Fan, "Ultra-reliable MU-MIMO detector based on deep learning for 5G/B5G-enabled IoT," *Physical Communication*, vol. 43, p. 101181, 2020.

[23] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: an overview," in *Proceedings of SAI Intelligent Systems Conference*, pp. 426–440, Cham, 2016.

[24] S. S. Mousavi, M. Schukat, and E. Howley, *Deep Reinforcement Learning: An Overview*, Intelligent Systems Conference 2018 (IntelliSys 2018), London, United Kingdom, 2018.

[25] Y. He, Z. Zhang, F. R. Yu et al., "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10433–10445, 2017.

[26] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: a comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[27] D. Guo, L. Tang, X. Zhang, and Y. Liang, "Joint optimization of handover control and power allocation based on multi-agent deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, 2020.

[28] S. Lai, "Intelligent secure mobile edge computing for beyond 5G wireless networks," *Physical Communication*, vol. 99, pp. 1–8, 2020.

[29] R. Zhao, "Deep reinforcement learning based mobile edge computing for intelligent Internet of things," *Physical Communication*, vol. 43, article 101184, 2020.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA, 2018.

[31] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Playing atari with deep reinforcement learning," *NIPS Deep Learning Workshop*, 2013.

[32] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[33] B. Guo, X. Zhang, Y. Wang, and H. Yang, "Deep-Q-network-based multimedia multi-service QoS optimization for mobile edge computing systems," *IEEE Access*, vol. 7, pp. 160961–160972, 2019.

[34] I. AlQerm and B. Shihada, "Energy efficient power allocation in multi-tier 5g networks using enhanced online learning," *IEEE Transactions on Vehicular Technology*, vol. 66, 2017.

[35] X. He, K. Wang, H. Huang, T. Miyazaki, Y. Wang, and S. Guo, "Green resource allocation based on deep reinforcement learning in content-centric iot," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, 2020.

[36] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 37, 2019.

[37] H. TY, N. Zhao, and H. Yin, "Integrated networking, caching and computing for connected vehicles: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 99, no. 10, 2017.

[38] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," *IEEE Wireless Communications and Networking Conference (WCNC)*, 2018, pp. 1–6, Barcelona, Spain, 2018.

[39] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 99, 2018.

[40] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, 2018.

[41] S. Park, J. Kim, D. Kwon, M. Shin, and J. Kim, "Joint offloading and streaming in mobile edges: a deep reinforcement learning approach," in *2019 IEEE VTS Asia Pacific Wireless Communications Symposium (APWCS)*, Singapore, Singapore, 2019.

[42] H. Zhang, W. Wu, C. Wang, M. Li, and R. Yang, "Deep reinforcement learning-based offloading decision optimization in mobile edge computing," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, Marrakesh, Morocco, Morocco, 2019.

[43] Z. Cheng and Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning," *Future Generation Computer Systems*, vol. 96, pp. 111–118, 2019.

[44] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: convergence of computing, caching and communications," *IEEE Access*, vol. 5, no. 3, pp. 6757–6779, 2017.

[45] Z. Zhang, Y. Zheng, C. Li, Y. Huang, and L. Yang, "Cache-enabled adaptive bit rate streaming via deep self-transfer reinforcement learning," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP) IEEE*, Hangzhou, China, 2018.

[46] L. Lei, L. You, G. Dai, T. X. Vu, D. Yuan, and S. Chatzinotas, "A deep learning approach for optimizing content delivering in cache-enabled hetnet," in *Wireless Communication Systems (ISWCS)*, pp. 449–453, Bologna, Italy, 2017.

[47] L. Lei, X. Xiong, H. Lu, and K. Zheng, "Collaborative edge caching through service function chaining: architecture and challenges," *IEEE Wireless Communications*, vol. 25, no. 3, pp. 94–102, 2018.

[48] C. H. Wei, Y. W. Hung, and F. L. Chin, "Q-learning based collaborative cache allocation in mobile edge computing," *Future Generation Computer Systems*, vol. 102, pp. 603–610, 2020.

[49] Z. Yang, Y. Liu, Y. Chen, and G. Tyson, "Deep reinforcement learning in cache-aided MEC networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019.

[50] Z. Chen, M. C. Gursoy, and S. Velipasalar, "A deep reinforcement learning-based framework for content caching," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, Princeton University, NJ, USA, 2018.

[51] C. Zhong, M. Cenk Gursoy, and S. Velipasalar, "Deep reinforcement learning based edge caching in wireless networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 48–61, 2020.

[52] M. C. Gursoy, C. Zhong, and S. Velipasalar, "Deep multi-agent reinforcement learning for cooperative edge caching," in *Machine Learning for Future Wireless Communications*, pp. 439–457, Shanghai, China, China, 2020.

[53] L. Liu, H. Hu, Y. Luo, and Y. Wen, "When wireless video streaming meets AI: a deep learning approach," *IEEE Wireless Communications*, vol. 27, 2019.

[54] H. Zhang, L. Dong, G. Gao, H. Hu, Y. Wen, and K. Guan, "DeepQoE: a multimodal learning framework for video quality

of experience (QoE) prediction," *IEEE Transactions on Multimedia*, vol. 22, 2020.

[55] G. Gao, L. Dong, H. Zhang, Y. Wen, and W. Zeng, "Content-aware personalised rate adaptation for adaptive streaming via deep video analysis," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, China, 2019.

[56] Y. Guo, F. R. Yu, J. An, K. Yang, C. Yu, and V. C. M. Leung, "Adaptive bitrate streaming in wireless networks with transcoding at network edge using deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3879–3892, 2020.

[57] M. Liu, Y. Teng, F. R. Yu, V. C. M. Leung, and M. Song, "A deep reinforcement learning-based transcoder selection framework for blockchain-enabled wireless D2D transcoding," *IEEE Transactions on Communications*, vol. 68, no. 6, pp. 3426–3439, 2020.

[58] F. Wang, C. Zhang, F. Wang et al., "Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized QoE," in *IEEE INFOCOM 2019*, Paris, France, 2019.

[59] Z. Pang, L. Sun, T. Huang, Z. Wang, and S. Yang, "Towards QoS-aware cloud live transcoding: a deep reinforcement learning approach," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 670–675, Shanghai, China, China, 2019.

[60] T. Y. Huang, R. Johari, N. Mc Keown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: evidence from a large video streaming service," *acm special interest group on data communication*, vol. 44, no. 4, pp. 187–198, 2015.

[61] M. Chen, M. Ponec, S. Sengupta, J. Li, and P. A. Chou, "Utility maximization in peer-to-peer systems with applications to video conferencing," *IEEE ACM Transactions on Networking*, vol. 20, no. 6, pp. 1681–1694, 2012.

[62] Y. Zheng, D. Wu, Y. Ke, C. Yang, M. Chen, and G. Zhang, "Online cloud transcoding and distribution for crowdsourced live game video streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 8, pp. 1777–1789, 2017.