IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 28 August 2023, accepted 22 October 2023, date of publication 26 October 2023, date of current version 7 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3327921

RESEARCH ARTICLE

A Deep Reinforcement Learning-Based Optimal Computation Offloading Scheme for VR Video Transmission in Mobile Edge Networks

XIANGYANG XU[1] AND YU SONG[2]
[1]Department of Cybersecurity, Henan Police College, Zhengzhou 450046, China
[2]School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China

Corresponding author: Xiangyang Xu (xyxu@hnp.edu.cn)

This work was supported by the Henan Province Science and Technology Research Project "Research on Collaborative Innovation and the Construction of High Quality Online Teaching System in Digital Campus."

ABSTRACT Large bandwidth, Low latency and intensive computing are the main challenge in high-performance virtual reality (VR) video transmission. As mobile edge computing (MEC) can provide computation and storage resources closer to terminals, it has been a promising mode in VR video transmission to substantially improve communication quality. This work focuses on the autonomous perception ability in MEC-supported VR video transmission, and introduces deep reinforcement learning to investigate optimal task offloading solutions. Therefore, this paper proposes a deep reinforcement learning-based optimal computation offloading scheme for VR video transmission in mobile edge networks. Specifically, a Deep Deterministic Policy Gradient-based computation offloading algorithm in designed as the main technical framework. The optimal planning of computation offloading strategies is viewed as a Markov decision problem, and a deep Q-Network is employed to deal with it. Finally, the setting of MEC-supported VR video transmission scenes is simulated, in which the proposed scheme is implemented for evaluation. The results are displayed in visualization format and show that the proposed task computation scheme can possess proper performance results in MEC-supported VR video transmission scenes.

INDEX TERMS Virtual reality, video transmission, computation offloading, mobile edge networks, reinforcement learning.

I. INTRODUCTION

Since 2016, the virtual reality industry has been growing rapidly. But due to the high demand for local computing and rendering equipment, users are still mainly a few enthusiasts and VR business is difficult to serve ordinary users [1]. Cloud virtual reality (VR) is a cloud-based real-time virtual reality technology, which uses cloud servers instead of users' local computing devices [2]. However, due to the intensive data volume of VR video, the bandwidth and latency limitation of network transmission becomes the new bottleneck of the whole system after the cloud computing and rendering [3]. For the basic 4KB resolution CloudVR service, the bandwidth requirement needs at least 40Mbit/s [4]. While the round trip time (RTT) should be less than 70 ms to provide a good experience for users [5]. In VR-based mobile network structures, currently, the transmission distance between the side of users and the side of servers usually remains in metropolitan level. Without considering the time consumption in procedures of forwarding and transmission, the RTT is as high as 20 to 40 ms for fiber optic transmission alone [6]. Such circumstances make it difficult to reach the requirement of instantaneous VR communication [7], [8]. With the development and implement of 5G networks, the bandwidth of mobile network is greatly improved. Mobile edge computing (MEC) technology deploys the server at the edge of the base station near the user, which greatly reduces the transmission delay by sinking the user-plane gateway, application edge, etc., making CloudVR possible [9]. On the other hand, the source side (edge server) and the channel

The associate editor coordinating the review of this manuscript and approving it for publication was Lei Wei.

side (base station) are more closely connected in the MEC scenario, and the bandwidth is sufficient to support baseband data at the server [10].

Each frame of VR video contains full view information, but the user can only see a small portion of the image within its FOV (field of view) when watching VR video, which means that there is a large amount of redundancy in each frame of VR image [11]. Ideally, only the valid image information within the user's FOV can be pushed based on the user's viewpoint information [12]. However, due to the limitations of network latency and bandwidth and the special nature of VR video viewing, this approach will lead to severe lag and image switching lag (switching images only when a new frame arrives) [13]. Therefore, the current mainstream program is based on the user's point-of-view information transmission quality uneven code stream program [14]. It is expected to ensure the quality of the image within the FOV at the same time, as far as possible to reduce the quality of redundant images [15]. When the user's viewing direction changes slightly, the user does not need to wait for the arrival of new-frame data, and can complete the screen switch locally, solving the problem of lag and lag [16]. The server side dynamically adjusts the FOV position of the transmitted video based on the user's uploaded viewpoint information to match the user's FOV as much as possible and realize the dynamic pushing of the user's viewpoint perception [17].

For the construction of non-uniform quality panoramic images, the most common solution is to split the panoramic video into different tiles and transmit different quality tiles according to the user's FOV, which largely saves network bandwidth [18]. These solutions are based on HTTP adaptive streaming protocols such as DASH, and the extend temporal segmentation to spatial segmentation [19]. Firstly, the panoramic video is divided into multiple blocks in time and space, and then multiple quality versions are generated for each block separately [20]. According to the user's point-of-view information, the appropriate quality version is selected for each block for transmission, and the closer the quality is to FOV, the higher the quality is, realizing dynamic adaptive pushing flow [21]. In essence, these methods are still based on ERP (equirectangular projection) panoramic map for block coding, and the quality of the image is relatively rigid between blocks, which affects the user's viewing experience. Another type of methods are based on projection transformation, such as tetrahedral and cube projection. They use the classical map projection idea to divide the sphere into many spherical trapezoids and project them onto some kind of polyhedra, with the characteristics of small distortion and high compression efficiency [22].

While common polyhedral projection methods have equal size on each side, a pyramidal projection scheme proposed by Facebook projects the spherical surface onto a positive quadrilateral and uses the difference in projected areas of different surfaces to generate a non-uniform mass image with a clear bottom surface and blurred sides [23]. This method integrates the generation of non-uniform quality images into the projection transformation, and the image quality changes more naturally. The above studies mainly focus on the source perspective and reduce the redundant data by coding or projection methods. An VR transmission mechanism based on joint source-channel coding is proposed by Cheng et al. [24], which maximizes the viewing quality within the user's FOV by using different levels of error protection after chunking the VR video with the user's FOV information. In the study of [25], in order to measure the user's viewing experience, a QoE (quality of experience) metric is defined and an efficient algorithm is presented for controlling the code rate and to maximize the QoE [26].

The research [27] investigates how to use SDN-based microcellular network architecture in 5G for multi-path collaborative VR video transmission, hoping to enhance the quality of VR video transmission using microcellular and edge data center (EDC) to improve the user experience. The main idea is that based on the pre-cached content in the edge data center, millimeter wave transmission of cached content is used between microcell sites to satisfy the requests of users within the coverage area microcell sites, thus satisfying the demand for low latency. The research [28] explores the transmission of VR video multicast in 5G networks and discusses the challenges of VR video multicast over 5G microcellular networks, while they propose a single frequency network (SFN)-based implementation and a millimeter wave band-based scheme, respectively [29]. In addition, a mechanism for VR video multicast transmission using D2D assistance in 5G is proposed in the research [30], which aims to efficiently utilize spectrum resources for VR video transmission in a multicast manner.

The research [31] proposes a multi-path layered transmission mechanism in 5G-based heterogeneous networks, where the base and augmented layer VR video content is mainly transmitted via WIFI, while the 5G network is mainly used for data retransmission as well as correction. The research [32] utilizes the mode of 5G MEC to investigate an adaptive VR video transmission scheme, where the transmission rate is minimized by optimizing the computation offloading and caching strategies with limited latency and energy consumption. The research [33] focuses on user viewpoint-based caching strategies in MEC and a viewpoint-aware caching strategy is presented to increase the hit rate of cached content. In the scenario of multiple micro-base stations with multi-user multicast, research [34] proposes a deep learning assisted scheduling and content quality adaptive ground transmission mechanism to solve the problem of VR video for micro-base station multicast transmission. The research [35] discusses the advantages of millimeter wave in VR video transmission and proposes a solution using mobile edge computing as well as pre-caching, aiming to provide reliable VR video transmission, taking the more interactive VR games as the research object. In the research [36], the problem of fusing millimeter wave communication,

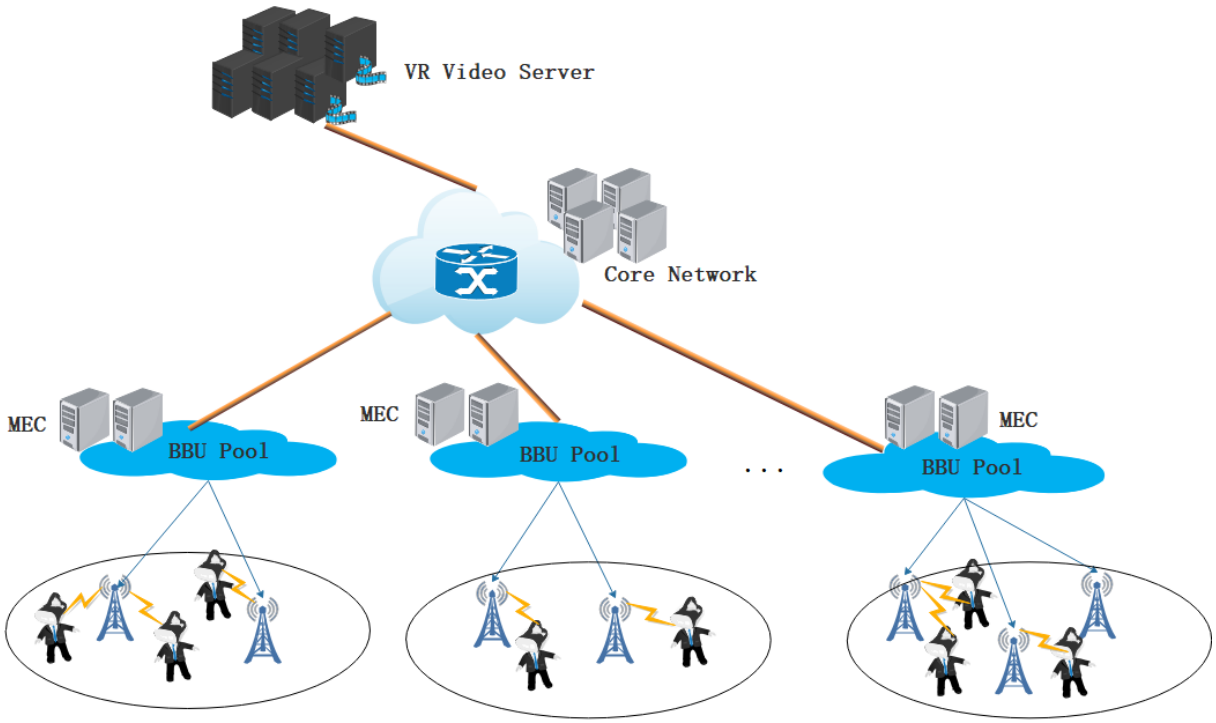

FIGURE 1. System model.

mobile edge computing, and pre-caching solution is further modeled, the corresponding solution algorithm is given, and the solution is verified by simulation to have lower latency compared to the conventional solution. Based on the MEC architecture, this paper proposes a resource optimization model based on the 5G smart edge, which can guarantee the service demand of mobile VR by collaboratively optimizing communication, computation and storage.

Given above analysis and description, we summarize main contributions of this paper as follows:

- The latency model of mobile VR services based on 5G edge computing are formulated in order to adapt the reinforcement learning model.
- We adopt reinforcement learning to tackle the offloading decision problem of mobile VR in edge computing. The proposed offloading algorithm is designed for improving the efficiency of edge resource.
- We conduct considerable amount of simulation experiments to verify performance of the proposed computation offloading scheme.

II. SYSTEM MODEL

Figure 1 shows the mobile VR service architecture with 5G smart edge converged communication, computation and storage. The architecture mainly includes mobile VR terminal, 5G access network, and MEC. this paper focuses on lightweight VR device and mobile terminal-based VR headset. 5G access network provides access services for users, and this paper adopts 5G typical network structure, where users transmit services through a remote radio head unit (RRH). The edge computing server is deployed near by the baseband pool (BBU), which provides storage and computing resources.

The latency of mobile VR services based on 5G edge computing mainly consists of two parts: content transmission latency and computation offloading latency, as shown in Figure 2, which illustrates the composition of computation offloading latency for offloading computation tasks to edge nodes. It is assumed that there are U mobile VR user, N VR video, R 5G base station and K edge computing node. The bandwidth and average signal-to-noise ratio of the user between associated base stations are denoted as $B_{r,u}$ and $\beta_{r,u}$, respectively, and the data transfer rates of the forward haul from the base stations to the BBU and the backhual from the BBU to the cloud service are denoted as d_f, d_b respectively. Therefore, the transmission delay of 1-bit data from the cloud server to the BBU, from the BBU to the base station, and from the base station to the VR user terminal is written as follows:

$$t_b = \frac{1}{d_b/U} \tag{1}$$

$$t_f = \frac{1}{d_f/U} \tag{2}$$

$$t_w = \frac{1}{B_{r,u} \cdot \log_2(1+\beta_{r,u})} \tag{3}$$

Through multi-level caching based on interest and popularity prediction, the VR contents requested by users are then all at the C-RAN edge nodes. Therefore, the content transmission delay from the C-RAN edge node to the user terminal is $D_u^{n,\tau}(t_f + t_w)$. In this paper, two types of intensive computation tasks, 3D scene reconstruction and high realism rendering, are computationally offloaded. The

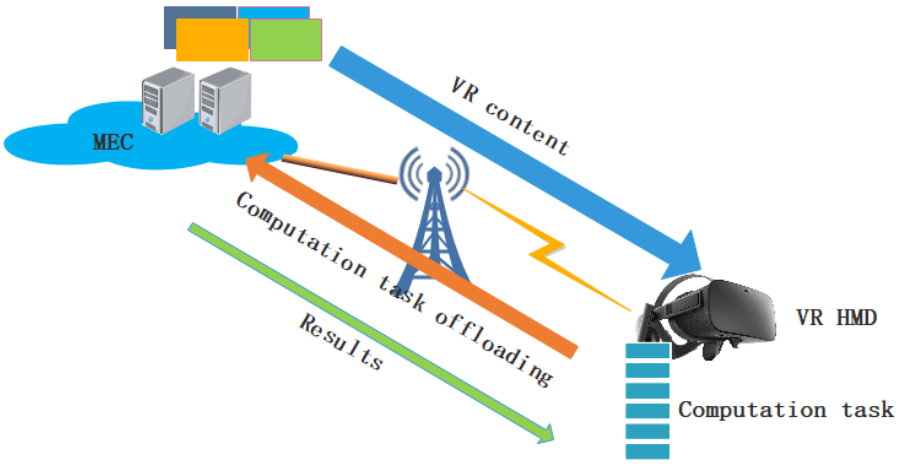

FIGURE 2. Latency model for mobile VR.

computationally offloaded delay mainly consists of two parts, one is the computation delay of the computation task calculation; the other part is the transmission delay of the computation task offloading and return the computation result data.

The computational task of Mobile VR user u is denoted as $C_u = \{C_u^{n,\tau}, D_u^{n,\tau}\}$,where $C_u^{n,\tau}$ denotes the CPU cycles required for the computational task of 3D scene reconstruction and rendering, $D_u^{n,\tau}$ is the amount of data currently requested by the user for VR content as shown in Equation (3), where $q_{n,\tau,m}^k$ is the bit rate of the first Tile of the first video segment of the first VR video. The data volume of the 3D scene reconstruction and rendering computation tasks, and the data volume of the computation task return results are defined as $\varsigma_u^{n,\tau}$, $\xi_u^{n,\tau}$. In addition, for user terminals and edge nodes, computing power of them are denoted as H_u and H_k, respectively.

$$D_u^{n,\tau} = \sum_{k}^{K} \sum_{m}^{M} q_{n,\tau,m}^k \tag{4}$$

In this paper, the following three computation modes are used: 1) the mobile terminal completes the computation task locally, 2) the computation task is offloaded to the edge node, and 3) a hybrid mode of local computation and edge node computation offloading. Therefore, the computational delay for local 3D scene reconstruction and rendering t_l, the computational delay for projection and rendering at the edge node t_k, and the computational delay for rendering the computational tasks locally and at the edge node t_h, respectively, are expressed as Equations (5), (6), and (7), respectively:

$$t_l = \frac{C_u^{n,\tau}}{H_u} \tag{5}$$

$$t_k = \frac{C_u^{n,\tau}}{H_{u,k}} \tag{6}$$

$$t_h = \frac{\delta C_u^{n,\tau}}{H_u} + \frac{(1-\delta) C_u^{n,\tau}}{H_{u,k}} \tag{7}$$

III. DEEP REINFORCEMENT LEARNING-BASED TASK OFFLOADING SCHEME

A. PROBLEM FORMULATION

The massive amount of information in mobile VR leads to high processing complexity, which puts a strong demand on the intensive computing capability of the system. VR information processing requires real-time completion of intensive computing tasks to ensure that users can obtain a natural and smooth experience. Due to the limitation of computing power, mobile terminals are difficult to support mobile VR intensive computing tasks in mobile VR computing tasks, mobile terminals. There are still obvious gaps in terms of computing power to meet the needs of mobile VR intensive computing. If the calculation is migrated to a remote cloud server, its latency performance is difficult to guarantee. Therefore, it is necessary to effectively make full use of computing ability of 5G mobile terminals from comprehensive views, so as to fulfill the intensive computing requirements of VR. Through the analysis of the intensive computing tasks of mobile VR, it is found that 3D scene reconstruction and rendering require the largest amount of computation in the whole process, which cannot be satisfied by the current computing power of mobile terminals, while such computing tasks as video/audio decoding can meet the computational requirements. Therefore, this paper will focus on two types of intensive computation tasks, namely 3D scene reconstruction and high realism rendering, to study the computation offloading strategy.

The time-varying characteristics of the wireless spectrum lead to fluctuations in the communication links of mobile VR users. In this paper, the time-varying characteristics of the wireless channel need to be considered in the computational offloading, and the computational offloading allocation algorithm for the channel quality of mobile terminals will be designed. The computing capacity of edge computing nodes is limited to a certain extent, especially in the current situation that edge computing nodes cannot meet the offloading of intensive computing tasks for multiple users under the nodes,

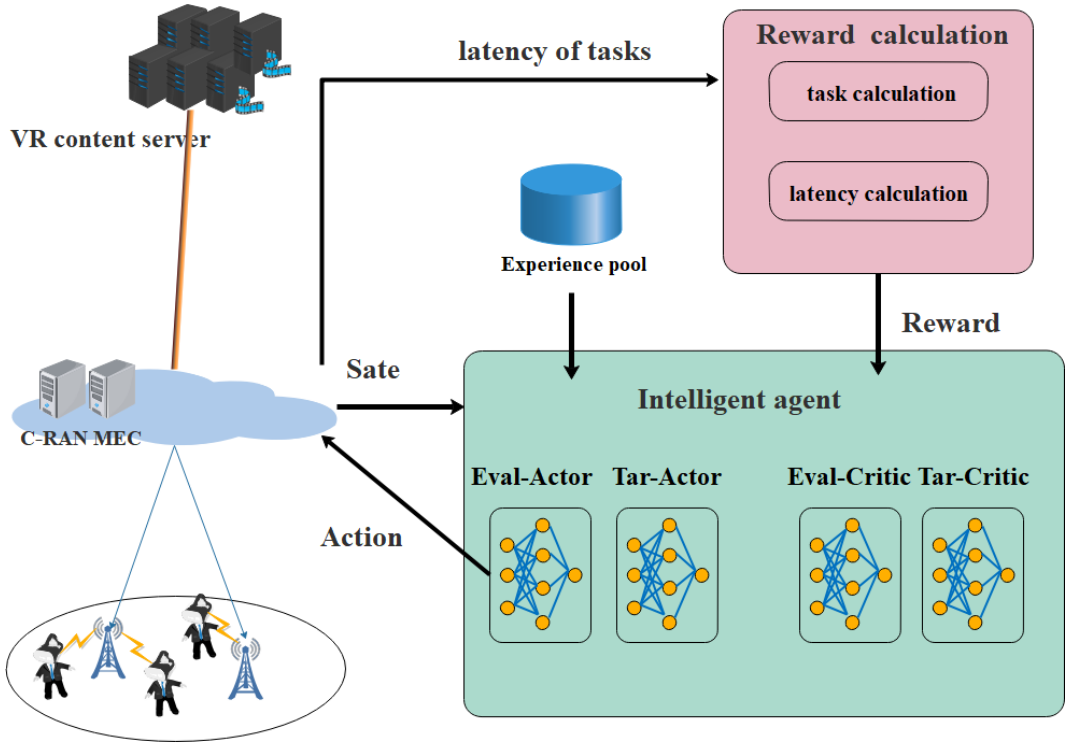

FIGURE 3. DDPG based intelligent decision algorithm for VR task offloading.

and it is necessary to offload some computing tasks to the core network edge computing nodes to meet the offloading demand of intensive computing. Based on the above analysis, this paper models the offloading optimization problem of VR computing tasks as the following problem:

$$\begin{aligned} &\min_{A,B,\Delta} \sum_{u}^{U} \sum_{\tau}^{\Gamma} t_{de} + (\alpha t_l + \lambda t_k + (1-\alpha-\lambda) t_h) \\ &s.t.\ t_{\max} C_u^{n,\tau} \le H_u \\ &\quad t_{\max}(1-\delta) \sum_{U} C_u^{n,\tau} \le \sum_{U} H_{u,k} \\ &\quad a \in \{0,1\} \\ &\quad \lambda \in \{0,1\} \end{aligned} \tag{8}$$

where t_{de} is the content transmission delay from the C-RAN edge node to the user terminal.

$$t_{de} = D_u^{n,\tau}(t_f + t_w) \tag{9}$$

B. PROPOSED DQN ALGORITHM

Markov Decision Process (MDP) is the most classic in reinforcement learning A modeling approach, it is a mathematical model of sequential decision making used to de-model in a system the stochasticity of policies achievable by an intelligence with the reward values returned by the environment. When MDP is used to deal with the ofloading decision problem of edge computing, the main objects can be represented by a triple $\{S, A, R\}$. Among, S denotes the whole state space of scenes, A denotes the whole action space of agents, and R denotes the rewarding function for actions taken by agents. And we consider the algorithmic process within the control node as an intelligent body, which will automatically obtain from each interaction with the environment $\{S_1, A_1, R_1, \ldots .S_i, A_i, R_i\}$ sequence. As more experience is gained, the intelligence performs better in decision making. The definition of $\{S, A, R\}$ is as follows:

S: Network state. The computing entities in the proposed network model in this paper are endpoints, edge servers and cloud servers. The cloud server is approximated as an infinite resource, so there is no task queue duration in calculating the computational latency of the cloud server. The network state of edge computing is N terminals $q_1, \ldots .q_n$ with E edge servers $q_1, \ldots .q_e$ the task queue.

A: Offloading action. At each instant, taking into account the task's individual delay threshold, an action may be taken by the agent, so as to assign tasks for a server to make processing. We designate the computational offloading action as $A(A^{d\to\tilde{e}}, A^{\tilde{e}\to\overleftarrow{e}})$.

R: Reward function definition. For each action made by the intelligence, the environment will provide a reward automatically. Then, we can define the cumulative reward as:

$$R_i = \begin{cases} -t_i \\ \ln(1 - \dfrac{1}{e^{\sqrt{t_i}}}) \end{cases} \tag{10}$$

$$G = \sum_{t=1}^{T} R(s_t', a_t', s_{t+1}) \tag{11}$$

The final objective is:

$$\underset{A}{Max}\, E[\sum_{t=1}^{T} R(s_t', a_t', s_{t+1})] \tag{12}$$

The MDP problem can be modeled to a quadratic $< S, A, P, R >$. In order to make the resource allocation decisions, the agent has interactions between the environment to acquire samples, and estimates the reward by the resulting samples. The final goal is to solve the optimal policy π_*. To fit for the above analyzed problem scenario, the DDPG-based approach is chosen to construct the decision-making strategy. DDPG is an enhanced gradient algorithm for Policy, and is driven from the dual network of deep Q-network (DQN). The deterministic policy, as opposed to the random policy, has only one action selection, simplifying the sample size during training.

As shown in Figure 3, the deployment of the intelligent decision algorithm comprised of two components: the environment and the intelligent agent. The intelligent agent responsible for offloading decisions for edge computing is deployed on the core router, and the task information generated on the device is handed over to the intelligent agent for decision making. The intelligent agent is deployed with a DDPG-based offloading algorithm for edge computing. DDPG employs a policy $\pi_\theta(s)$ for resource allocation decision-making, which deterministically converts a state into a specific action, and chooses only one action compared to a random policy, which significantly enhances the training convergence. Within the Actor-Critic network, the agent is boosted by the Actor network using the Policy Gradient method, and the action with the highest probability is selected directly via the policy function $\pi_\theta(s)$ at the current state. Correspondingly, the action of Actor's evaluation is determined by the Critic network. The formula for DQN-based policy gradient is:

$$\nabla_\theta J(\pi_\theta) = E_{s,\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_\pi Q_\pi(s, a)|a = \pi_\theta(s)] \tag{13}$$

As depicted in Figure 3 the DRL-based algorithm for distributed dynamic resource optimization consists of four components. They all have their own identical structures. First, the agent gets the current network state through having interaction with the environment. And it records the present state vector, action vector, reward and next state vector as $\phi(S)$, A, R, $\phi(S')$ respectively. It stores them within the experience pool. With the purpose of gaining deeper insights into the action space, a fixed amount of noise is introduced to the actions chosen by Actor. Then, subsequent to the accumulation of a specific data volume in experience pool, a mini-batch size data block, with a specific size, is extracted from it and fed into the Critic network to calculate the following actions. Correspondingly, the loss function is calculated as:

$$loss = \frac{1}{m}\sum_{j=1}^{m}(y_j - \hat{y}_j)^2 \tag{14}$$

where $\hat{y}_j$ is denoted as:

$$\hat{y}_j = Q(\phi(S_j, a_j, w) \tag{15}$$

Once the loss function has been obtained, the agent then updates all the parameters of Critic network by passing the gradient backwards through w of the neural network. Leveraging the loss function, Eval-Net update the parameters of the network uses with gradient descent scheme, after a specific episode, the latest parameters are copied from Eval-Net to Target-Net. DDQN uses the TD difference method to achieve the update in each step. After multiple episodes of parameter updating, the loss function of Eval-Net will converge to the optimal policy π after applying the trained network parameters w. The update of the parameters for the Actor-Critic neural network is defined as follows:

$$w' \leftarrow \tau w + (1 - \tau)w' \tag{16}$$

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta' \tag{17}$$

In contrast to DQN, the proposed algorithm utilizes a step-wise update method, where only a part of the parameters are updated each time. At the same time, aiming at the achieving better exploration of the whole solution space, the learning process incorporates the introduction of noise η as well, which increases the randomness when selecting actions, and the action can be selected via the following formula:

$$A = \pi_\theta(S) + \eta \tag{18}$$

Subsequently, the loss functions for both the Critic network and the Actor network are represented as:

$$J(w) = \frac{1}{m}\sum_{j=1}^{m}(y_j - \hat{y}_j)^2 \tag{19}$$

And the gradient of the loss function pertaining to the Actor network is articulated in the subsequent manner.

$$\nabla_\theta J(\pi_\theta) = E_{s,\rho^\pi}\,(\Omega_1 \cdot \Omega_2) \tag{20}$$

where

$$\Omega_1 = \nabla_\pi Q_\pi(s, a)|_{s=s_i, a=\pi_\theta(s)} \tag{21}$$

$$\Omega_2 = \nabla_\theta \pi_\theta(s)|s = s_i \tag{22}$$

The DDPG-based computation offloading algorithm encompasses two components. At first the decisions executed by the intelligent agent resemble those of the stochastic algorithm, and after the DDPG-based computation offloading algorithm learns from the interaction, the offloading policy becomes closer to the optimal. The parameters of the trained DDPG neural network will be updated at the end of the learning iteration.

IV. SIMULATION AND ANALYSIS

A. SETUP

In this paper, the simulation of edge environment is conducted using the Networkx library in python, while the construction of the proposed algorithm in this paper is achieved through

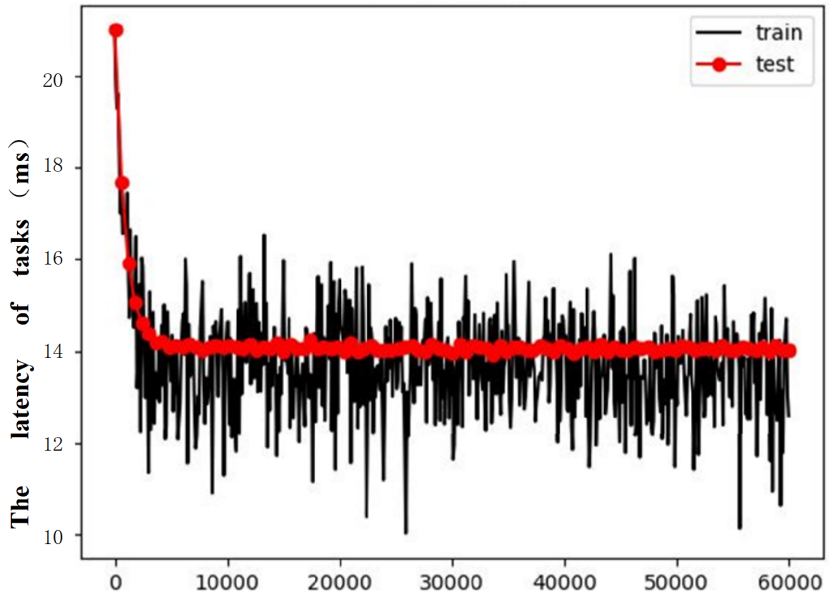

FIGURE 4. Performance of latency of tasks with DDPG.

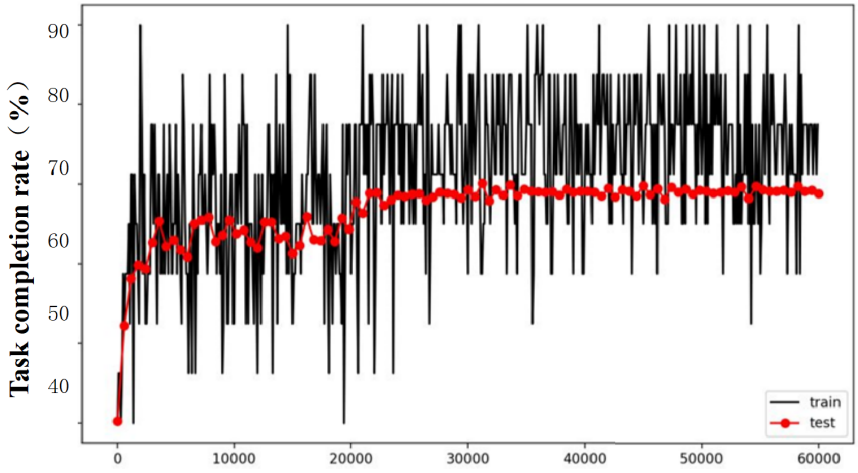

FIGURE 5. Performance of tasks completion rate with DDPG.

the application of tensorflow. This experiment divides the edge servers into three layers, which includes the terminal layer, the edge layer and the cloud layer. The network model assumes that there are 3-7 wireless edge servers and in the edge layer,there are 2-5 wired edge servers, and the wireless edge servers are served in a certain range. Initially, for each AP,there exist N=3 edge devices within the service range, and each edge in each time slot follows the Poisson distribution for the probability of offload requests. The computing capacity of each edge server is set from 4GHZ to 6GHZ, the computation capacity of the cloud is 10GHZ, and the computation capacity of each edge is set as 0.5GHZ. The bandwidth between the edge server and terminal is set as 100MB/S. The bandwidth between edge server and the wireless terminal is set as 50MB/S. The bandwidth among different edge servers is set as 300MB/S. The transmission delay among edge servers is set as 10ms. And for link between the edge server and the cloud server, transmission delay is set as 30ms. It is assumed that both of the Actor network and Critic network are set as a three-layer structure. And the second fully connected layer is assumed to consist of 200 neurons. The neural network for this experiment is implemented by Tensorflow, and the hyper-parameters of networks are given in Table 1. The specific parameters of the compared DQN are presented in Table 2.

B. RESULTS

In this paper, firstly, we compared the convergence of this algorithm throughout the training process, followed by a

TABLE 1. Parameters of simulation.

Parameters	Value
episode	5000
a_A	1×10^{-3}
a_c	2×10^{-3}
Mem	1000G
Batch	64
γ	0.9
TAU	0.01

TABLE 2. Parameters of DQN.

Parameters	Value
learning rate	0.001
discount factor	0.96
exploration rate	0.05
batch size	512
replay buffer size	$1e^5$

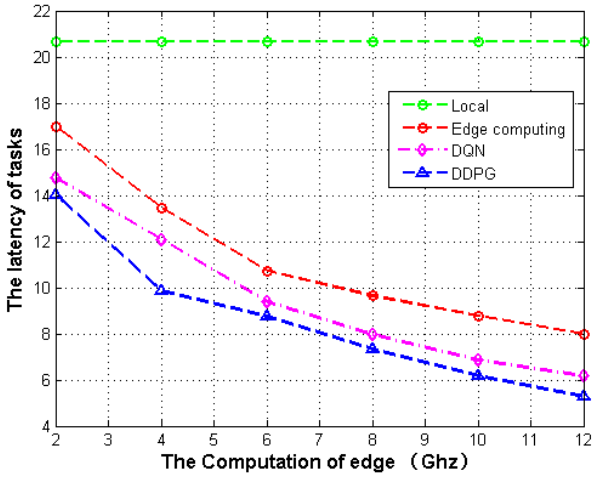

FIGURE 6. Comparison of latency performance VS. computation of mobile edge.

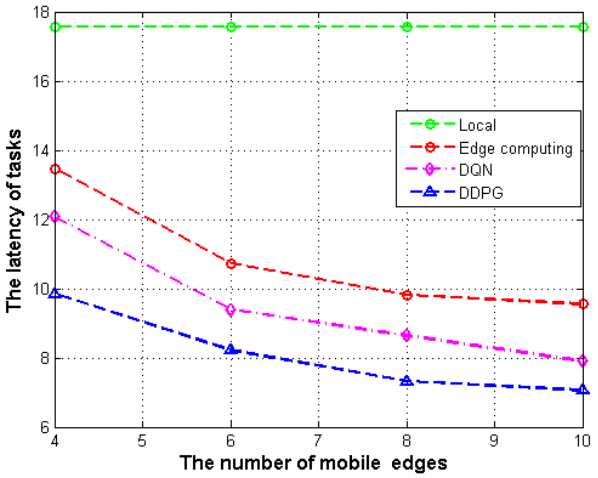

FIGURE 7. Comparison of latency performance VS. number of mobile edge.

comparison with the performance of other algorithms in the identical scenario. As illustrated in Figure 4, the average task latency drops rapidly in the first 30000 training rounds. However, after 30000 of training rounds, the task latency stabilizes at 13.8 ms or less. As shown in Figure 5, the task completion rate increases significantly in the first 20000 rounds of training, and the training converges to 85%-90% after the 20000th iteration. The above results show that the training effect slows down after 20000 iterations of the algorithm, and the training convergence is achieved.

In the simulation, we are trying to examine the performance of the proposed method with different situation. As shown in Figure 6, the number of users and mobile edge is fixed, and with the increase of computation of mobile edge, the latency of tasks are decreased. As shown in Figure 6, the DDPG algorithm is significantly better than the local computation and server computation mode, and the DDPG is also better than the DQN since the action space of the DQN algorithm is discrete values. when the server computation capacity improves, the average task latency of offload method decreases except for the terminal local computation, and the local computation is limited by bandwidth and the computation capacity of the terminal's processor. Then, the relationship between the number of servers and the task latency expectation is given in Figure 7, the capacity of edge servers and the number of users (i.e. the computation tasks) are fixed, with the increase of the number of edge servers, the individual task latency expectation is gradually decreasing, limited by the bandwidth between servers. From the figure we can see that with the increase of servers count to 8, it is not obvious to increase to improve the effect. The effect of the number of mobile terminals on the expected task delay is shown as Figure 8. We observe that the DDPG-based algorithm outperforms better the rest of the algorithms, and the expected task delay experiences a rapid surge with the growing number of mobile terminals. In the context of mobile edge computing, the growing number of mobile terminals results in a corresponding rise in task offloading requests during each time slot, consequently amplifying the computational burden on the edge server.

Figure 9 shows the task success rate for each algorithm. It is evident that the DDPG-based task offloading decision

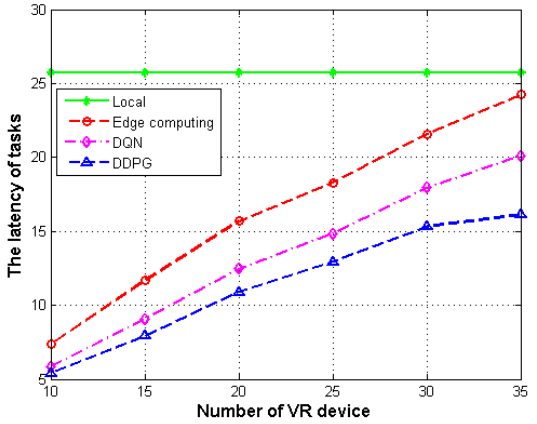

FIGURE 8. Comparison of latency performance VS. number of mobile VR device.

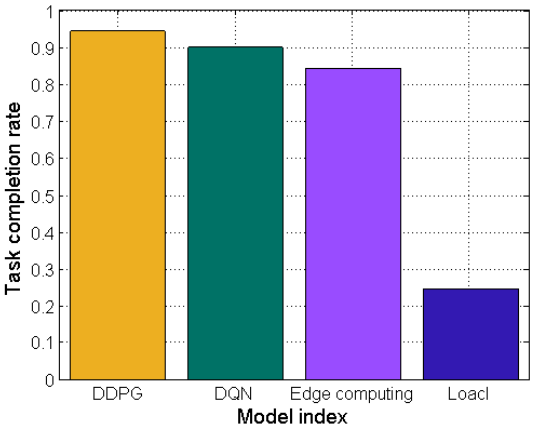

FIGURE 9. Comparison of task completion performance with different methods.

mechanism outperforms other mechanisms when considering the same task queue, followed by the probability of completing the task within the specified time frame for DQN and edge server offloading computation, and the local computing scheme without task computation offloading has difficulty in meeting the computation time frame requirements of the application. The reason for this result is that the action space of DDPG is continuous. Therefore, the action granularity of computation offloading is more delicate and precise than that of the DQN-based computation offloading mechanism. The computation offloading method in which the tasks are all executed on the edge server will waste the computation resources of the end device and the cloud server, so although the edge server computation offloading method can better meet the requirements of the device application, the service effect is still inferior to the two methods based on DRL. End devices with weak processors will have difficulty in meeting the computationally intensive tasks in the task queue without using compute offloading services.

V. CONCLUSION

This paper utilizes reinforcement learning to address the offloading decision issue concerning mobile VR in edge computing. First, The latency model of mobile VR services in 5G edge computing network are formulated in order to adapt the reinforcement learning model. Then, a DDPG-based offloading algorithm is designed for improving the efficiency of edge resource. Finally, Plenty simulations are conducted to verify that the proposed DDPG-based resource offloading algoirthm can improve the performance of edge computing offloading service.

REFERENCES

[1] Q. Li, D. Wang, and H. Lu, "A cooperative caching and computing-offloading method for 3C trade-off in VR video services," *IEEE Access*, vol. 9, pp. 124010–124022, 2021, doi: 10.1109/ACCESS.2021.3110741.

[2] Z. Guo, K. Yu, N. Kumar, W. Wei, S. Mumtaz, and M. Guizani, "Deep-distributed-learning-based POI recommendation under mobile-edge networks," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 303–317, Jan. 2023.

[3] Z. Guo, K. Yu, A. K. Bashir, D. Zhang, Y. D. Al-Otaibi, and M. Guizani, "Deep information fusion-driven POI scheduling for mobile social networks," *IEEE Netw.*, vol. 36, no. 4, pp. 210–216, Jul. 2022.

[4] L. Yang, Y. Li, S. X. Yang, Y. Lu, T. Guo, and K. Yu, "Generative adversarial learning for intelligent trust management in 6G wireless networks," *IEEE Netw.*, vol. 36, no. 4, pp. 134–140, Jul. 2022, doi: 10.1109/MNET.003.2100672.

[5] L. Zhao, Z. Yin, K. Yu, X. Tang, L. Xu, Z. Guo, and P. Nehra, "A fuzzy logic-based intelligent multiattribute routing scheme for two-layered SDVNs," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4189–4200, Dec. 2022.

[6] J. Zhang, L. Zhao, K. Yu, G. Min, A. Y. Al-Dubai, and A. Y. Zomaya, "A novel federated learning scheme for generative adversarial networks," *IEEE Trans. Mobile Comput.*, early access, May 22, 2023, doi: 10.1109/TMC.2023.3278668.

[7] P. Lin, Q. Song, F. R. Yu, D. Wang, and L. Guo, "Task offloading for wireless VR-enabled medical treatment with blockchain security using collective reinforcement learning," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15749–15761, Nov. 2021, doi: 10.1109/JIOT.2021.3051419.

[8] L. Hu, Y. Tian, J. Yang, T. Taleb, L. Xiang, and Y. Hao, "Ready player one: UAV-clustering-based multi-task offloading for vehicular VR/AR gaming," *IEEE Netw.*, vol. 33, no. 3, pp. 42–48, May 2019, doi: 10.1109/MNET.2019.1800357.

[9] Z. Guo, K. Yu, K. Konstantin, S. Mumtaz, W. Wei, P. Shi, and J. J. P. C. Rodrigues, "Deep collaborative intelligence-driven traffic forecasting in green Internet of Vehicles," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 1023–1035, 2023.

[10] J. Yang, J. Luo, J. Wang, and S. Guo, "CMU-VP: Cooperative multicast and unicast with viewport prediction for VR video streaming in 5G H-CRAN," *IEEE Access*, vol. 7, pp. 134187–134197, 2019, doi: 10.1109/ACCESS.2019.2941646.

[11] Q. Li, L. Liu, Z. Guo, P. Vijayakumar, F. Taghizadeh-Hesary, and K. Yu, "Smart assessment and forecasting framework for healthy development index in urban cities," *Cities*, vol. 131, Dec. 2022, Art. no. 103971.

[12] C. Liu, K. Wang, H. Zhang, X. Li, and H. Ji, "Rendered tile reuse scheme based on FoV prediction for MEC-assisted wireless VR service," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1709–1721, May 2023, doi: 10.1109/TNSE.2023.3234029.

[13] Z. Guo, D. Meng, C. Chakraborty, X.-R. Fan, A. Bhardwaj, and K. Yu, "Autonomous behavioral decision for vehicular agents based on cyber-physical social intelligence," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 4, pp. 2111–2122, Aug. 2023.

[14] L. Zhao, Z. Bi, A. Hawbani, K. Yu, Y. Zhang, and M. Guizani, "ELITE: An intelligent digital twin-based hierarchical routing scheme for softwarized vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 9, pp. 5231–5247, Sep. 2023.

[15] Y. Jin, J. Liu, and F. Wang, "Ebublio: Edge assisted multi-user 360-degree video streaming," in *Proc. IEEE Conf. Virtual Reality 3D User Inter. Abstr. Workshops (VRW)*, Christchurch, New Zealand, Mar. 2022, pp. 600–601, doi: 10.1109/VRW55335.2022.00151.

[16] C. Xu, J. Qin, P. Zhang, K. Gao, and L. A. Grieco, "Reinforcement learning-based mobile AR/VR multipath transmission with streaming power spectrum density analysis," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4529–4540, Dec. 2022, doi: 10.1109/TMC.2021.3082912.

[17] K. Liu, Y. Liu, J. Liu, and A. Argyriou, "Tile caching for scalable VR video streaming over 5G mobile networks," *J. Vis. Commun. Image Represent.*, vol. 79, Aug. 2021, Art. no. 103210, doi: 10.1016/j.jvcir.2021.103210.
[18] Q. He, Z. Feng, H. Fang, X. Wang, L. Zhao, Y. Yao, and K. Yu, "A blockchain-based scheme for secure data offloading in healthcare with deep reinforcement learning," *IEEE/ACM Trans. Netw.*, early access, Jun. 5, 2023, doi: 10.1109/TNET.2023.3274631.
[19] Z. Zhou, X. Dong, R. Meng, M. Wang, H. Yan, K. Yu, and K. R. Choo, "Generative steganography via auto-generation of semantic object contours," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2751–2765, 2023.
[20] K. Long, Y. Cui, C. Ye, and Z. Liu, "Optimal wireless streaming of multi-quality 360 VR video by exploiting natural, relative smoothness-enabled, and transcoding-enabled multicast opportunities," *IEEE Trans. Multimedia*, vol. 23, pp. 3670–3683, 2021, doi: 10.1109/TMM.2020.3029880.
[21] W. Wei, J. Han, Y. Xing, K. Xue, J. Liu, and R. Zhuang, "MP-VR: An MPTCP-based adaptive streaming framework for 360-degree virtual reality videos," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Montreal, QC, Canada, Jun. 2021, pp. 1–6, doi: 10.1109/ICC42927.2021.9500817.
[22] X. Chen, B. Cao, and I. Ahmad, "Lightweight neural network-based viewport prediction for live VR streaming in wireless video sensor network," *Mobile Inf. Syst.*, vol. 2021, pp. 8501990:1–8501990:12, Nov. 2021, doi: 10.1155/2021/8501990.
[23] Z. Zhou, Y. Su, J. Li, K. Yu, Q. M. J. Wu, Z. Fu, and Y. Shi, "Secret-to-image reversible transformation for generative steganography," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 4118–4134, Sep. 2023.
[24] Q. Cheng, H. Shan, W. Zhuang, L. Yu, Z. Zhang, and T. Q. S. Quek, "Design and analysis of MEC- and proactive caching-based 360° mobile VR video streaming," *IEEE Trans. Multimedia*, vol. 24, pp. 1529–1544, 2022, doi: 10.1109/TMM.2021.3067205.
[25] Y. Liu, J. Liu, A. Argyriou, L. Wang, and Z. Xu, "Rendering-aware VR video caching over multi-cell MEC networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2728–2742, Mar. 2021, doi: 10.1109/TVT.2021.3057684.
[26] X. Yuan, Z. Zhang, C. Feng, Y. Cui, S. Garg, G. Kaddoum, and K. Yu, "A DQN-based frame aggregation and task offloading approach for edge-enabled IoMT," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1339–1351, May 2023.
[27] J. Yang, J. Luo, D. Meng, and J.-N. Hwang, "QoE-driven resource allocation optimized for uplink delivery of delay-sensitive VR video over cellular network," *IEEE Access*, vol. 7, pp. 60672–60683, 2019, doi: 10.1109/ACCESS.2019.2915370.
[28] L. Zhao, Y. Cui, C. Guo, and Z. Liu, "Optimal streaming of 360 VR videos with perfect, imperfect and unknown FoV viewing probabilities," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Taiwan, Dec. 2020, pp. 1–6, doi: 10.1109/GLOBECOM42002.2020.9322648.
[29] Q. Zhang, Z. Guo, Y. Zhu, P. Vijayakumar, A. Castiglione, and B. B. Gupta, "A deep learning-based fast fake news detection model for cyber-physical social services," *Pattern Recognit. Lett.*, vol. 168, pp. 31–38, Apr. 2023.
[30] J. Du, F. R. Yu, G. Lu, J. Wang, J. Jiang, and X. Chu, "MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9517–9529, Oct. 2020, doi: 10.1109/JIOT.2020.3003449.
[31] J. Dai, Z. Zhang, S. Mao, and D. Liu, "A view synthesis-based 360° VR caching system over MEC-enabled C-RAN," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3843–3855, Oct. 2020, doi: 10.1109/TCSVT.2019.2946755.
[32] C. Zheng, S. Liu, Y. Huang, and L. Yang, "MEC-enabled wireless VR video service: A learning-based mixed strategy for energy-latency tradeoff," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Seoul, South Korea, May 2020, pp. 1–6, doi: 10.1109/WCNC45663.2020.9120529.
[33] J. Dai and D. Liu, "An MEC-enabled wireless VR transmission system with view synthesis-based caching," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshop (WCNCW)*, Marrakech, Morocco, Apr. 2019, pp. 1–7, doi: 10.1109/WCNCW.2019.8902723.
[34] X. Feng, Y. Liu, and S. Wei, "LiveDeep: Online viewport prediction for live virtual reality streaming using lifelong deep learning," in *Proc. IEEE Conf. Virtual Reality 3D User Interfaces (VR)*, Atlanta, GA, USA, Mar. 2020, pp. 800–808, doi: 10.1109/VR46266.2020.1584727730619.
[35] W.-K. Seo and C. E. Rhee, "Low latency streaming for path-walking VR systems," *IEEE Access*, vol. 10, pp. 50702–50714, 2022, doi: 10.1109/ACCESS.2022.3174351.
[36] T. M. C. Chu and H.-J. Zepernick, "Performance analysis of an adaptive rate scheme for QoE-assured mobile VR video streaming," *Computers*, vol. 11, no. 5, p. 69, Apr. 2022, doi: 10.3390/computers11050069.

XIANGYANG XU received the bachelor's degree in information engineering from Zhengzhou University, in 2001, and the master's degree in software engineering from PLA Information Engineering University, in 2005. He is currently an Associate Professor with the Henan Police College. His research interests include computer networks, edge computing, and multimedia technology.

YU SONG received the master's degree in information management from Zhengzhou University, in 2005. He is currently an Associate Professor with the School of Computer and Artificial Intelligence, Zhengzhou University. His research interests include information planning, data governance, the Internet of Things, and artificial intelligence.