Learning-Based Prediction, Rendering and Association Optimization for MEC-Enabled Wireless Virtual Reality (VR) Networks

Xiaonan Liu, *Student Member, IEEE*, and Yansha Deng, *Member, IEEE*

Abstract—Wireless-connected Virtual Reality (VR) provides immersive experience for VR users from anywhere at anytime. However, providing wireless VR users with seamless connectivity and real-time VR video with high quality is challenging due to its requirements in high Quality of Experience (QoE) and low VR interaction latency under limited computation capability of VR device. To address these issues, we propose a MEC-enabled wireless VR network, where the field of view (FoV) of each VR user can be real-time predicted using Recurrent Neural Network (RNN), and the rendering of VR content is moved from VR device to MEC server with rendering model migration capability. Taking into account the geographical and FoV request correlation, we propose centralized and distributed decoupled Deep Reinforcement Learning (DRL) strategies to maximize the long-term QoE of VR users under the VR interaction latency constraint. Simulation results show that our proposed MEC rendering schemes and DRL algorithms substantially improve the long-term QoE of VR users and reduce the VR interaction latency compared to rendering at VR devices.

Index Terms—Field of view (FoV) prediction, rendering, downlink transmission, mobile edge computing (MEC), deep reinforcement learning (DRL), and virtual reality (VR).

I. INTRODUCTION

WITH the development of virtual reality (VR) technology, the interactions between VR users and their world will be revolutionized. VR can connect users across global communities within highly immersive virtual worlds that breaks geographical boundaries. This vision has inspired the commercial release of various hardware devices, such as Facebook Oculus Rift [1]. Indeed, it is anticipated that 99 million VR devices are needed in 2021 [2], and that the market will reach 108 billion dollars by then [3]. However, the poor user experience provided by traditional computer-supported VR devices constrains the type of activities and experience of the VR user. One of the main barriers of the wired connected VR devices is the limited mobility of VR users. To overcome this disadvantage, wireless connected VR devices can be

potential solution in providing ubiquitous user experiences from anywhere at anytime, and also can unleash plenty of novel VR applications [4]. Nevertheless, there are some unique challenges in wireless VR system that does not exist in wired VR system and traditional wireless video transmission system as identified in [4]. This includes how to provide seamless and real-time VR video with high quality through unstable wireless channels, solve handover issues when VR users are in mobility, and support the asymmetric and coupled traffic in the uplink and downlink transmission [4]–[7].

There are growing research interests in wireless VR networks. In [8], the echo state network (ESN) was proposed to solve the resource block allocation problem in both uplink and downlink wireless VR transmission to maximize the average quality of service (QoS) of VR users. Extending from [8], the authors in [9] optimized the resource block allocation using ESN to maximize the success transmission probability accounting for the VR user data correlation. In [10] and [11], joint caching and computing optimization problem was formulated to minimize the average required transmission rate and maximize the average tolerant delay, respectively. In [12] and [13], a Recurrent Neural Network (RNN) based on long short-term memory (LSTM) was applied for viewpoint prediction. In [14]–[16], and [17], the authors studied optimal multicast of tiled 360 VR video from one base station to multiple VR users in OFDMA and TDMA system, and focused on downlink transmission via optimizing transmission power.

For real-time interactive VR applications, the latest 2D video content needs to be first delivered to the VR device via wired/wireless communication, and then rendered to 3D VR video locally. In reality, human wearing VR device only watches a portion of observable visual world at any given time, which is so called Field of View (FoV). Rendering the full 360 degree video in real-time can be costly both for downlink transmission and computation. One potential solution is to only render the requested FoV each time based on the uplink tracking information of VR users' motion, including head and eye movements. According to [18], the data size of the rendered FoV is 75% of that of the stitched 2D image, which means that the size of data to be delivered via downlink transmission can be reduced by 25% compared to delivering the stitched 2D images.

Rendering real-time VR videos with high quality demands the computing unit with high processing ability, so that the

rendering latency can be reduced, unfortunately, the computation ability and battery capacity of wireless VR devices are limited. Recently, mobile edge computing (MEC) has emerged to push mobile computing and network control to network edge, so as to enable computation-intensive and latency-critical applications at the resource-limited mobile devices, which promise dramatic reduction in latency and energy consumption [19].

Shifting the FoV rendering task from VR device to MEC server can not only alleviate the computation requirement at the VR device, but also potentially decrease the VR interaction latency especially for those VR users in the same virtual VR environment requesting the same FoV. With MEC multicast to a group of VR users selecting the same FoV, and unicast to single VR user selecting unique FoV, the downlink transmission cost of network can be further decreased. In practice, each MEC has different computation capability, it would be interesting to explore if we can obtain further gain for multiple VR user groups selecting the same FoV but different MECs by performing rendering at only one MEC, and then migrate the rendered FoV wired to other MECs, namely, rendered FoV migration.

As pointed out by [4], the Quality of Experience (QoE) of VR transmission is substantially different from that of conventional video transmission, due to its unique requirements in the VR interaction latency, and asymmetric uplink and downlink data rates. Motivated by above, in this paper, we focus on optimizing the QoE of VR users with interactive VR applications in a MEC-enabled wireless VR networks, and we develop a decoupled learning strategy to efficiently optimize the QoE in wireless VR system, which can improve the training efficiency [20], [21]. The main contributions can be summarized as follows:

- We propose a MEC-enabled wireless VR networks, where the field of view (FoV) of each VR user can be real-time predicted, and the rendering of VR content is moved from VR device to MEC server with rendering model migration capability.
- With the aim of optimizing the long-term QoE of VR users, we propose a decoupled learning strategy. This strategy decouples the optimization by separately resolving two-sub tasks, which are FoV prediction and rendering MEC association with the help of Recurrent Neural Network (RNN) predictor and Deep Reinforcement Learning (DRL) algorithms, respectively.
- In order to capture the complex dynamics of the FoV request from each VR device, we propose the RNN model based on Gated Recurrent Unit (GRU) architecture at the central controller to predict the requested FoV in the current time slot based on those in previous time slots sent via uplink. Our results shown that our proposed FoV prediction based on GRU achieves 96% in prediction accuracy.
- Accounting for the geographical and predicted FoV request correlation, we propose centralized and distributed decoupled Deep Reinforcement Learning (DRL) strategies based on Deep Q-Network (DQN) and Actor Critic (AC) [22]–[28] to maximize the long-term QoE of
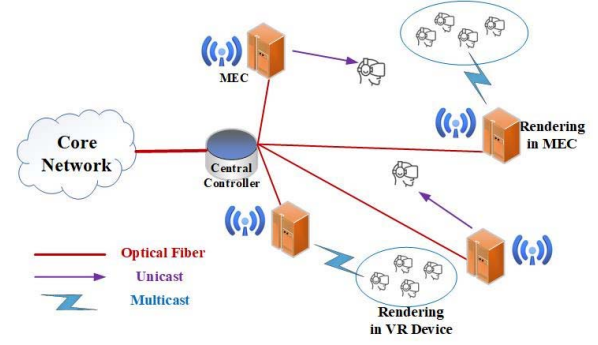

Fig. 1. Wireless VR system in cellular network.

VR users, via determining optimal association between MEC and VR user group, and optimal rendering MEC for model migration. By comparing with non-learning based nearest MEC association, our results on centralized and distributed DQN shown substantial gain in both QoE and VR interaction latency. Interestingly, the rendering model migration further improve these gains.

The rest of this paper is organized as follows. The system model and problem formulation are proposed in Section II. RNN-based FoV prediction and DRL-based MEC rendering, migration and association scheme are presented in Section III. The simulation results and conclusions are described in Section IV and Section V, respectively.

II. SYSTEM MODEL AND PROBLEM FORMULATION

We consider a wireless VR system where multiple MECs are connected to a central controller through a fiber link and serves K_{VR} VR users via wireless links, as shown in Fig. 1. The central controller is connected to the core network via fiber, and can fetch real-time 2D videos without distortion from the core network. According to [29], 2D images can be captured by multiple unmanned aerial vehicles (UAVs) and transmitted to the core network to render into VR videos. Therefore, we assume that the required 2D images captured by UAVs or other equipment are available at the core network in real-time. In addition, we assume that the MEC can only perform the rendering for the VR users choosing the same FoV at each time slot.[1]

A. System Model

Our MEC-enabled wireless VR system is consist of four main parts, including FoV selection and prediction, uplink transmission, FoV rendering, and multi-group multicast and unicast downlink transmission.

1) FoV Selection and Prediction: When VR users enjoy the VR video, because of the restricted area of vision in human eye, they usually watch only a portion of the VR video, namely, the Field of View (FoV) [30], [31], which specifies a $150° \times 135°$ (i.e., diagonal $200°$) FoV requirement. As the FoV is part of the actual rendered 3D VR video, the MEC

---

[1]In the real-time VR system, it is possible that the 2D pictures for the required VR video frames do not exist in the MEC, and the MEC needs to obtain the required 2D pictures from the core network.
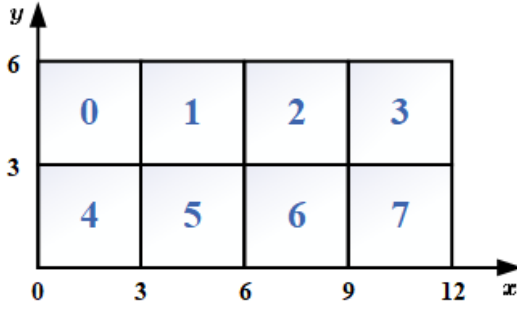
Fig. 2. Brownian motion for FoV selection.

benefits from obtaining the tracking information related to the viewport of the VR user, and uses the video characteristics such as projection and mapping formats to generate FoV. If the VR system could know the required FoV before transmission, it could render and deliver the FoV to VR users in advance, which can decrease the latency.

Let us first denote the total number of FoVs of a VR virtual environment as N_{FoV}. When VR users enjoy the 360-degree video, they may randomly select the same or different FoVs in the continuous time slots. When the VR user enjoys the VR video, it usually focuses on the centre of the virtual environment, and slowly moves its eyes towards the object where it wants to look at in the VR environment, and this is the so-called fixational eye movement [32], [33]. The fixational eye movements have been widely assumed to be a random uncorrelated process following Brownian motion [34]. Therefore, the Brownian motion can be used to model the eye movement of each VR user corresponding to different FoVs over time slots. To capture the FoV selection in the 3D VR scenario close to reality, we map the 3D VR view into a large 2D view with N_{FoV} FoVs in each time. When the VR user's eyes move inside the cubic in Fig. 2, the corresponding FoV will be selected. According to Brownian motion [35], [36], the eye movement of the VR user at the tth time slot can be modeled by an independent Gaussian distribution with variance $2D(t)$ and zero mean $\mathcal{N}(0, 2D(t))$. Thus, the eye movement of the kth VR user in a mapped 2D VR view at the tth time slot can be expressed as

$$\triangle S_k(t) = \{\mathcal{N}(0, 2D_k(t)), \mathcal{N}(0, 2D_k(t))\}. \qquad (1)$$

At one time slot, the selected FoV of the kth VR user is observed, which can be used for FoV prediction in the next time slot. For example, we assume that there are 8 2D FoVs in a VR virtual environment, as shown in Fig. 2. If the VR user selects the 2th FoV at a certain time slot, in the next time slot, it will select one of FoVs among the $1, 2, 3, 5, 6, 7$th FoVs. If the VR user selects a boundary FoV, such as the 0th FoV at a certain time slot, it can choose one FoV from the $0, 1, 4, 5$th FoVs in the next time slot.

Based on the historical FoV selection in the previous time slots, the wireless VR system can predict the requested FoV of the kth VR user in the next time slot.

2) Uplink Transmission: For conventional wireless VR system without FoV prediction, each VR user needs to deliver its actual request FoV to the MEC through uplink broadcast transmission. To focus on the rendering and downlink transmission, at the tth time slot, we assume that the received FoV $\widehat{F_t^{\mathrm{oV}}}$ is equal to the actual requested FoV F_t^{oV} following

$$\widehat{F_t^{\mathrm{oV}}} = F_t^{\mathrm{oV}}. \qquad (2)$$

For our proposed system with FoV prediction, the uplink received FoV F_t^{oV} after prediction will be used to check the correctness of predicted FoV $\widetilde{F_t^{\mathrm{oV}}}$. We define the FoV prediction accuracy over T time slots as

$$P_{\mathrm{FoV}} = \frac{1}{T}\sum_{t=1}^{T}\left[\frac{F_t^{\mathrm{oV}} - \widetilde{F_t^{\mathrm{oV}}}}{F_t^{\mathrm{oV}}}\right] \times 100\%. \qquad (3)$$

3) FoV Rendering: When VR user requests to watch VR video frames, based on the predicted or uplink received FoV, the corresponding portion of the sphere can be rendered at MEC or VR device.

- As shown in Fig. 3 (a) and (b), when the rendering function is executed at the MEC, a stitched 2D image, whose color model is RGB, will be rendered into the required FoV through equirectangular projection (ERP) mapping [31] for downlink multicast or unicast transmission.
- When the rendering function is processed at the VR device, the stitched 2D picture frames will be first multicast or unicast to the VR users. Then, it will be rendered into the required FoV via ERP mapping, which is shown in Fig. 3 (c).

Here, the number of pixels in the stitched 2D image is used to quantify the size of the executed data during FoV rendering. To evaluate the best rendering strategy, we propose three FoV rendering schemes as detailed below:

(a) *MEC Rendering Without Migration Scheme:* In this scheme, the rendering from stitched 2D image to 3D FoV occurs at each MEC with associated VR users.

(b) *MEC Rendering With Migration Scheme:* With different computational capabilities at each MEC, VR users selecting the same FoV but associated with different MECs only performs FoV rendering at only one MEC, and this selected MEC can migrate the rendered FoV to other MECs via fiber links to save the computational resources.

(c) *VR Device Rendering Scheme:* This scheme is a conventional scheme for comparison, where the FoV rendering occurs only at the VR device, such that the stitched 2D picture frames need to be transmitted to the VR device for rendering locally using ERP mapping. Due to the fact that the computation ability of the VR device is much smaller than that of the MEC, we expect that it may cost much more time for the VR device to render the required FoV.

4) Multi-Group Multicast and Unicast: Based on the received FoV in the uplink or the predicted FoV of each VR user, the VR users with the same received/predicted FoVs can be grouped together. After FoV rendering, the MECs will multicast the required FoVs to VR users selecting the same FoV, or unicast a single VR user selecting unique FoV, respectively. Let us consider a set of $\mathcal{B} = \{1, 2, \ldots, B\}$ MECs, and each MEC is equipped with N transmit antennas. These B MECs serve the downlink transmission for B
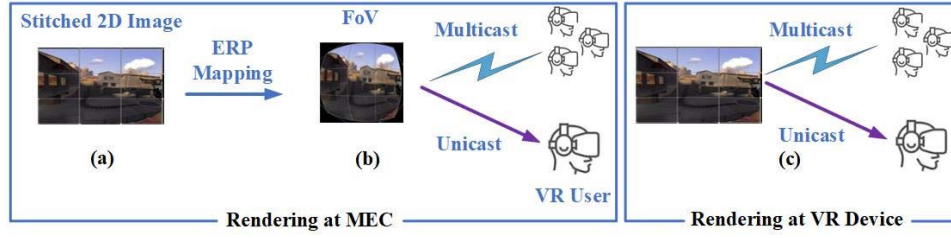
Fig. 3. FoV rendering.

VR user groups $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_B\}$ with single antenna. The B VR user groups can be multicast group, unicast group, or inactive group with no VR users. Assuming that there are M multicasting groups and U unicasting groups ($M+U \leq B$), these M multicast groups and U unicast groups can be denoted as the sets $\mathcal{V}^{\mathrm{mul}} = \{\mathcal{V}_1^{\mathrm{mul}}, \mathcal{V}_2^{\mathrm{mul}}, \ldots, \mathcal{V}_M^{\mathrm{mul}}\}$ and $\mathcal{V}^{\mathrm{uni}} = \{\mathcal{V}_1^{\mathrm{uni}}, \mathcal{V}_2^{\mathrm{uni}}, \ldots, \mathcal{V}_U^{\mathrm{uni}}\}$, respectively. With the number of VR users in the kth multicasting group denoted as $|\mathcal{V}_k^{\mathrm{mul}}|$, the total number of VR users can be calculated as $K_{\mathrm{VR}} = \sum_{k=1}^{M} |\mathcal{V}_k^{\mathrm{mul}}| + U$. Note that each VR user can only be assigned to just one group.

B. Mathematical Model

1) FoV Rendering Model: We denote the number of pixels as N_p, and the size of each pixel is 8 bits. For the MEC rendering schemes, at each time slot, the size of the FoV to be transmitted in the downlink can be calculated as

$$C = 3 \times 8 \times N_p \times N_p \times V = 48N_p^2, \qquad (4)$$

where 3 presents the red, green and blue color in RGB model, the single-eye resolution is $N_p \times N_p$, and V is the number of viewpoints with $V = 2$ for two eyes. According to [4], the resolution of the FoV is at least 1080p, and C can be very large when N_p is high. Usually, the FoV has to be compressed before downlink multicast or unicast. By assuming the compression ratio as $\mathcal{C}_{\mathcal{R}}$, the size of the compressed data for downlink transmission can be calculated as $\frac{C}{\mathcal{C}_{\mathcal{R}}}$. In addition, through ERP mapping, 25 percent pixels of the stitched 2D image can be reduced [18], which means that the data size of the FoV is 75 percent of that of the stitched 2D image. Thus, the data size of the stitched 2D image can be calculated as $\mathcal{M} = \frac{2}{3}C = 32N_p^2$, and $32N_p^2$ bits data are required to be executed in the ERP mapping step.

2) Downlink Transmission Model: For VR users in the multicast groups, the multicast signal between the bth MEC and the kth VR user in the jth multicast group at the tth time slot can be written as

$$
\begin{aligned}
\mathbf{y}_{j_k,b}^{\mathrm{mul}}(t) = {}& \mathbf{h}_{j_k,b}^{H}(t)\mathbf{v}_{j,b}^{\mathrm{mul}}(t)x_b^{\mathrm{mul}}(t) \\
& + \sum_{\mathcal{V}_i^{\mathrm{mul}} \in \mathcal{V}^{\mathrm{mul}}/\mathcal{V}_b^{\mathrm{mul}}, m \in \mathcal{V}_i^{\mathrm{mul}}} \mathbf{h}_{j_k,i}^{H}(t)\mathbf{v}_{m,i}^{\mathrm{mul}}(t)x_i^{\mathrm{mul}}(t) \\
& + \sum_{\mathcal{V}_l^{\mathrm{uni}} \in \mathcal{V}^{\mathrm{uni}}, u \in \mathcal{V}_l^{\mathrm{uni}}} \mathbf{h}_{j_k,l}^{H}(t)\mathbf{v}_{u,l}^{\mathrm{uni}}(t)x_u^{\mathrm{uni}}(t) + \mathbf{n}_{j_k}(t),
\end{aligned}
$$
$$(5)$$

where $\mathbf{h}_{j_k,b}(t) \in \mathbb{C}^{M \times 1} \sim \mathcal{CN}(\mathbf{0}, \alpha\mathbf{I}_M)$ is the uncorrelated Rayleigh fading channel vector between the bth MEC

and the kth VR user in the jth multicast group, α is the large-scale fading coefficient of the multicast VR users. $\mathbf{v}_{j,b}^{\mathrm{mul}}(t) \in \mathbb{C}^{M \times 1}$ and $\mathbf{v}_{u,l}^{\mathrm{uni}}(t) \in \mathbb{C}^{M \times 1}$ are the multicast and unicast vectors from the bth and lth MECs connected to the VR users in the jth multicast group and the uth VR user in the unicast group, respectively. In (5), $x_b^{\mathrm{mul}}(t)$ and $x_u^{\mathrm{uni}}(t)$ are the multicast and unicast messages intended for the VR users in the jth multicast group and the uth VR user in the unicast group, respectively. We assume that $x_b^{\mathrm{mul}}(t)$ and $x_u^{\mathrm{uni}}(t)$ are independent from each other. Meanwhile, $\sum_{\mathcal{V}_i^{\mathrm{mul}} \in \mathcal{V}^{\mathrm{mul}}/\mathcal{V}_b^{\mathrm{mul}}, m \in \mathcal{V}_i^{\mathrm{mul}}} \mathbf{h}_{j_k,i}^{H}(t)\mathbf{v}_{m,i}^{\mathrm{mul}}(t)x_i^{\mathrm{mul}}(t)$ and $\sum_{\mathcal{V}_l^{\mathrm{uni}} \in \mathcal{V}^{\mathrm{uni}}, u \in \mathcal{V}_l^{\mathrm{uni}}} \mathbf{h}_{j_k,l}^{H}(t)\mathbf{v}_{u,l}^{\mathrm{uni}}(t)x_u^{\mathrm{uni}}(t)$ are the interference from the other MECs that provide FoVs for the VR users in other multicast and unicast groups, respectively. In addition, $\mathbf{n}_{j_k}(t) \sim \mathcal{CN}(0, \sigma_{j_k}^2 \mathbf{I}_M)$ is the additive white Gaussian noise at the kth VR user in the jth multicast group.

Based on (5), the multicast transmission rate between the kth VR user in the jth multicast group and the bth MEC at the tth time slot can be expressed as

$$R_{j_k,b}^{\mathrm{mul}}(t) = \log_2\left(1 + \frac{|\mathbf{h}_{j_k,b}^{H}(t)\mathbf{v}_{j,b}^{\mathrm{mul}}(t)|^2}{\mathbf{I}_{j_k,b}^{\mathrm{mul}}(t) + \sigma_{j_k}^2}\right), \qquad (6)$$

where

$$\mathbf{I}_{j_k,b}^{\mathrm{mul}}(t) = \sum_{\substack{\mathcal{V}_i^{\mathrm{mul}} \in \mathcal{V}^{\mathrm{mul}}/\mathcal{V}_b^{\mathrm{mul}} \\ m \in \mathcal{V}_i^{\mathrm{mul}}}} |\mathbf{h}_{j_k,i}^{H}(t)\mathbf{v}_{m,i}^{\mathrm{mul}}(t)|^2 + \sum_{\substack{\mathcal{V}_l^{\mathrm{uni}} \in \mathcal{V}^{\mathrm{uni}} \\ u \in \mathcal{V}_l^{\mathrm{uni}}}} |\mathbf{h}_{j_k,l}^{H}(t)\mathbf{v}_{u,l}^{\mathrm{uni}}(t)|^2. \quad (7)$$

For the VR users in the unicast groups, the unicast signal between the bth MEC and the kth VR user at the tth time slot can be expressed as

$$
\begin{aligned}
\mathbf{y}_{k,b}^{\mathrm{uni}}(t) = {}& \mathbf{g}_{k,b}^{H}(t)\mathbf{v}_{k,b}^{\mathrm{uni}}(t)x_b^{\mathrm{uni}}(t) \\
& + \sum_{\mathcal{V}_i^{\mathrm{uni}} \in \mathcal{V}^{\mathrm{uni}}/\mathcal{V}_b^{\mathrm{uni}}, u \in \mathcal{V}_i^{\mathrm{uni}}} \mathbf{g}_{k,i}^{H}(t)\mathbf{v}_{u,i}^{\mathrm{uni}}(t)x_i^{\mathrm{uni}}(t) \\
& + \sum_{\mathcal{V}_l^{\mathrm{mul}} \in \mathcal{V}^{\mathrm{mul}}, m \in \mathcal{V}_l^{\mathrm{mul}}} \mathbf{g}_{k,l}^{H}(t)\mathbf{v}_{m,l}^{\mathrm{mul}}(t)x_l^{\mathrm{mul}}(t) + \mathbf{n}_k(t),
\end{aligned}
$$
$$(8)$$

where $\mathbf{g}_{k,b}(t) \in \mathbb{C}^{M \times 1} \sim \mathcal{CN}(\mathbf{0}, \beta\mathbf{I}_M)$ is the uncorrelated Rayleigh fading channel vector between the bth MEC and the kth VR user in the unicast group, and β is the large-scale fading coefficient for the unicast VR users. Meanwhile, $\sum_{\mathcal{V}_i^{\mathrm{uni}} \in \mathcal{V}^{\mathrm{uni}}/\mathcal{V}_b^{\mathrm{uni}}, u \in \mathcal{V}_i^{\mathrm{uni}}} \mathbf{g}_{k,i}^{H}(t)\mathbf{v}_{u,i}^{\mathrm{uni}}(t)x_i^{\mathrm{uni}}(t)$ and $\sum_{\mathcal{V}_l^{\mathrm{mul}} \in \mathcal{V}^{\mathrm{mul}}, m \in \mathcal{V}_l^{\mathrm{mul}}} \mathbf{g}_{k,l}^{H}(t)\mathbf{v}_{m,l}^{\mathrm{mul}}(t)x_l^{\mathrm{mul}}(t)$ are the interference from the other MECs that provide service for VR users in the unicast and multicast groups, respectively. In (8),

$\mathbf{n}_k(t) \sim \mathcal{CN}(0, \sigma_k^2 \mathbf{I}_M)$ is the additive white Gaussian noise at the kth VR user in the unicast group.

Based on (8), the unicast transmission rate between the kth VR user and the bth MEC at the tth time slot can be written as

$$R_{k,b}^{\text{uni}}(t) = \log_2 \left(1 + \frac{|\mathbf{g}_{k,b}^H(t)\mathbf{v}_{k,b}^{\text{uni}}(t)|^2}{\mathbf{I}_{k,b}^{\text{uni}}(t) + \sigma_k^2} \right), \qquad (9)$$

where

$$\mathbf{I}_{k,b}^{\text{uni}}(t) = \sum_{\substack{\mathcal{V}_i^{\text{uni}} \in \mathcal{V}^{\text{uni}}/\mathcal{V}_b^{\text{uni}} \\ u \in \mathcal{V}_i^{\text{uni}}}} |\mathbf{g}_{k,i}^H(t)\mathbf{v}_{u,i}^{\text{uni}}(t)|^2 + \sum_{\substack{\mathcal{V}_l^{\text{mul}} \in \mathcal{V}^{\text{mul}} \\ m \in \mathcal{V}_l^{\text{mul}}}} |\mathbf{g}_{k,l}^H(t)\mathbf{v}_{m,l}^{\text{mul}}(t)|^2. \qquad (10)$$

3) VR Interaction Latency: As defined in [4], [37], the VR interaction latency T^{loop} is the time starting from the VR user's movement to the time where the virtual environment responds to its movements. It is consisted of four parts: 1) the time of the VR user to uplink its FoV request, and other tracking information (T^{uplink}); 2) the time of the FoV rendering at MECs or VR devices to generate the predicted or uplink requested FoV (T^{render}); 3) the time to migrate the rendered FoV from one optimal MEC to the other MECs ($T^{\text{migration}}$) with the VR users selecting the same FoV via fiber; and 4) the time to transmit the rendered FoV or the stitched 2D picture frames from the MEC to the VR user (T^{downlink}) depending on rendering at the MEC or the VR device, respectively. Thus, the VR interaction latency T^{loop} can be calculated as

$$T^{\text{loop}} = T^{\text{uplink}} + T^{\text{render}} + T^{\text{migration}} + T^{\text{downlink}}, \qquad (11)$$

as shown in Fig. 4 (a).

Let us assume the execution ability of the GPU of the kth MEC or VR device as F_k^{MEC} and F_k^{VR}, respectively. We use f_k^{MEC} and f_k^{VR} to represent the number of cycles required for processing one bit of input data of the kth MEC or VR device, respectively. Here, the number of cycles depends on the application type and the GPU architecture of the kth MEC or VR device. For the MEC rendering scheme with migration, we assume that the bth MEC is selected to be only rendering MEC with the same FoV request, the distance between the kth MEC and the bth MEC is $\hat{L}_{k,b}$, and the transmission rate of the optical fiber is R^{fiber}.

The VR interaction latency of the proposed MEC and VR device rendering schemes are shown in Fig. 4 (b) and (c) and introduced in details as follows:

(a) *MEC Rendering Without Migration: a-1) Without prediction:* At the tth time slot, the VR user needs to deliver the actual FoV request of VR users F_t^{oV} through uplink broadcast transmission, and the rendering of received FoV $\widehat{F_t^{\text{oV}}}$ is executed at each MEC with associated VR user. As shown in Fig. 4 (b1), at the tth time slot, if the kth VR user is served by the kth MEC, the VR interaction latency can be calculated as

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{render}} + T_k^{\text{downlink}} \\ &= T_k^{\text{uplink}} + \frac{f_k^{\text{MEC}}\mathcal{M}}{F_k^{\text{MEC}}} + \frac{C}{\mathcal{C}_{\mathcal{R}} R_{k,k}^{\text{down}}}, \end{aligned} \qquad (12)$$

where $R_{k,k}^{\text{down}} \in \{R_{k,k}^{\text{mul}}, R_{k,k}^{\text{uni}}\}$, and $R_{k,k}^{\text{mul}}$ and $R_{k,k}^{\text{uni}}$ are given in (6) and (9).

a-2) With prediction: As shown in Fig. 4 (b3), based on the uplink received FoVs at the tth time slot $\widehat{F_t^{\text{oV}}}$ and several previous time slots, we predict the FoV preference of each VR user at the $(t+1)$th time slot $\widehat{F_{t+1}^{\text{oV}}}$. For the kth VR user directly served by the kth MEC, the VR interaction latency with predicted FoV can be written as (12) with $T_k^{\text{uplink}} = 0$.

(b) *MEC Rendering With Migration: b-1) Without prediction:* For the VR users requesting the same FoV $\widehat{F_t^{\text{oV}}}$ through uplink transmission only select one MEC for rendering, and the rendered FoV can be migrated to other MECs. As shown in Fig. 4 (b2), at the tth time slot, if the kth VR user directly served by the bth MEC performs rendering itself, the VR interaction latency of the required FoV of the kth VR user can be presented as

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{render}} + T_k^{\text{downlink}} \\ &= T_k^{\text{uplink}} + \frac{f_b^{\text{MEC}}\mathcal{M}}{F_b^{\text{MEC}}} + \frac{C}{\mathcal{C}_{\mathcal{R}} R_{k,b}^{\text{down}}}, \end{aligned} \qquad (13)$$

where $R_{k,b}^{\text{down}} = R_{k,b}^{\text{mul}}$ and $R_{k,b}^{\text{mul}}$ is given in (6).

If the kth VR user is served by the kth MEC, where the rendering is not performed by itself, but by the bth MEC, the interaction latency of the kth VR user can be presented as

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{render}} + T_k^{\text{migration}} + T_k^{\text{downlink}} \\ &= T_k^{\text{uplink}} + \frac{f_b^{\text{MEC}}\mathcal{M}}{F_b^{\text{MEC}}} + \frac{\hat{L}_{k,b}}{R^{\text{fiber}}} + \frac{C}{\mathcal{C}_{\mathcal{R}} R_{k,k}^{\text{down}}}, \end{aligned} \qquad (14)$$

where $R_{k,k}^{\text{down}} = R_{k,k}^{\text{mul}}$ and $R_{k,k}^{\text{mul}}$ is given in (6).

b-2) With prediction: As shown in Fig. 4 (b4), at the tth time slot, for the kth VR user directly served by the bth MEC, the VR interaction latency with the predicted FoV of the kth VR user can be denoted as (13) with $T_k^{\text{uplink}} = 0$.

Otherwise, the predicted FoV will be migrated to the kth MEC from the bth MEC, and the VR interaction latency of the predicted FoV of the kth VR user can be denoted as (14) with $T_k^{\text{uplink}} = 0$.

(c) *VR Device Rendering: c-1) Without prediction:* According to Fig. 4 (c1), at the tth time slot, for the kth VR user served by the kth MEC without FoV prediction, the VR interaction latency of the kth VR user can be written as

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{downlink}} + T_k^{\text{render}} \\ &= T_k^{\text{uplink}} + \frac{\mathcal{M}}{\mathcal{C}_{\mathcal{R}} R_{k,k}^{\text{down}}} + \frac{f_k^{\text{VR}}\mathcal{M}}{F_k^{\text{VR}}}, \end{aligned} \qquad (15)$$

where $R_{k,k}^{\text{down}} \in \{R_{k,k}^{\text{mul}}, R_{k,k}^{\text{uni}}\}$, and $R_{k,k}^{\text{mul}}$ and $R_{k,k}^{\text{uni}}$ are given in (6) and (9).

c-2) With prediction: As shown in Fig. 4 (c2), at the tth time slot, the FoV preference of the VR users for the $(t+1)$th time slot will be predicted, and the VR interaction latency with the predicted FoV of the kth VR user served by the kth MEC can be presented as (15) with $T_k^{\text{uplink}} = 0$.

4) VR Quality of Experience: The quality of the FoV can be influenced by many factors, such as blockiness, blur, contrast distortion, freezing, colour depth, sharpness, etc [38], [39]. To evaluate the performance of the proposed MEC rendering
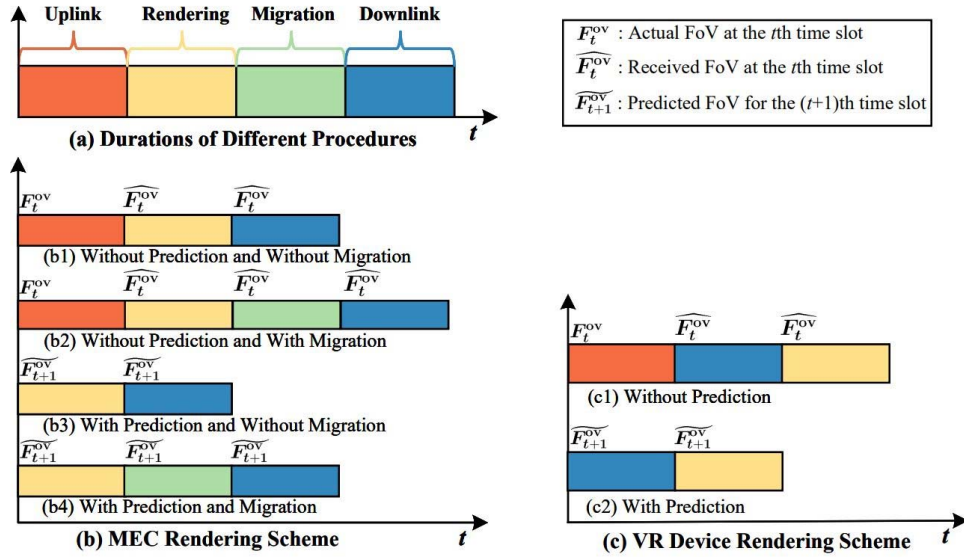
Fig. 4. VR interaction latency of the proposed MEC and VR device rendering schemes.

schemes, we focus on the objective in maximizing the Peak Signal-to-Noise Ratio (PSNR) [40], knowing that it is the most common and simple objective VR video quality assessment, and the PSNR is usually defined by the Mean Squared Error (MSE) of the kth VR user between an initial FoV $\mathcal{I}_k$ and the distorted FoV $\mathcal{D}_k$. According to [41], to measure the QoE of the kth VR user based on the MSE, we propose a binary function where $\mathcal{I}_k = 1$ and $\mathcal{D}_k \in \{0, 1\}$ to represent whether the FoV can be rendered and delivered within the threshold of VR interaction latency of the kth VR user. For real-time interactive VR applications, the delayed FoV will bring unpleasant human experience, thus, for the kth VR user, we revise basic QoE model by incorporating a maximum VR interaction latency requirement T_k^{th}. More specifically, if $T_k \leq T_k^{\text{th}}$, the rendered FoV is regarded as successfully delivered to VR device, then $\mathcal{D}_k = 1$, otherwise, $\mathcal{D}_k = 0$. The MSE of the kth VR user can be written as

$$\text{MSE}_k = (\mathcal{I}_k - \mathcal{D}_k)^2. \tag{16}$$

According to [41, Eq. (2)], the PSNR of the kth VR user is defined as

$$\text{PSNR}_k = 10 \log_{10} \frac{1}{\text{MSE}_k}. \tag{17}$$

As can be seen from (17), for $\text{MSE}_k = 0$, $\text{PSNR}_k \to \infty$. To avoid the infinite value of PSNR, we introduce a positive number $\triangle$ and modify (17) as

$$\text{PSNR}_k = 10 \log_{10} \frac{1 + \triangle}{\text{MSE}_k + \triangle}, \tag{18}$$

where $\triangle > 0$ and we set $\triangle = 1$ in this paper.

C. Problem Formulation

To ensure that each requested FoV is rendered and transmitted within the VR interaction latency, we aim to optimize the total QoE under fixed VR interaction latency constraint via determining the optimal association between

MEC and VR user group, and optimal rendering MEC for model migration.

The proposed MEC rendering schemes aim at maximizing the long-term total QoE under VR interaction latency constraint in the continuous time slots with respect to the policy π that maps the current state information S_t to the probabilities of selecting possible actions in A_t. Therefore, based on the QoE of each VR user, an optimization problem (P1) is formulated as

$$\text{(P1)} \quad \max_{\pi(A_t|S_t)} \sum_{i=t}^{\infty} \sum_{k=1}^{K_{\text{VR}}} \gamma^{i-t} \mathbb{E}_{\pi}[\text{PSNR}_k^i] \tag{19}$$

$$T_k \leq T_k^{\text{th}}, \tag{20}$$

where $\gamma \in [0, 1)$ is the discount factor which can determine the weight of the future QoE, and $\gamma = 0$ means that the agent just concerns the immediate reward. The state S_t contains the index of the requested FoV, the location of each VR user, and the computation ability of each MEC. The action A_t includes the optimal association between MEC and VR user group, and optimal rendering MEC for model migration.

Since the dynamics of the wireless VR system is Markovian in continuous time slots, this is a Partially Observable Markov Decision Process (POMDP) problem which is generally intractable. Here, the partial observation refers to that the MECs can only know the previous FoV requests and the location of each VR user in the environment, while they are unable to know all the information of the communication environment, including, but not limited to, the channel conditions, and the FoV request in the current time slot. Furthermore, the traditional optimization methods may need the global information to achieve the optimal solution, which not only increase the overhead of signal transmission, but also increase the computation complexity. Approximate solutions will be discussed in Section III.
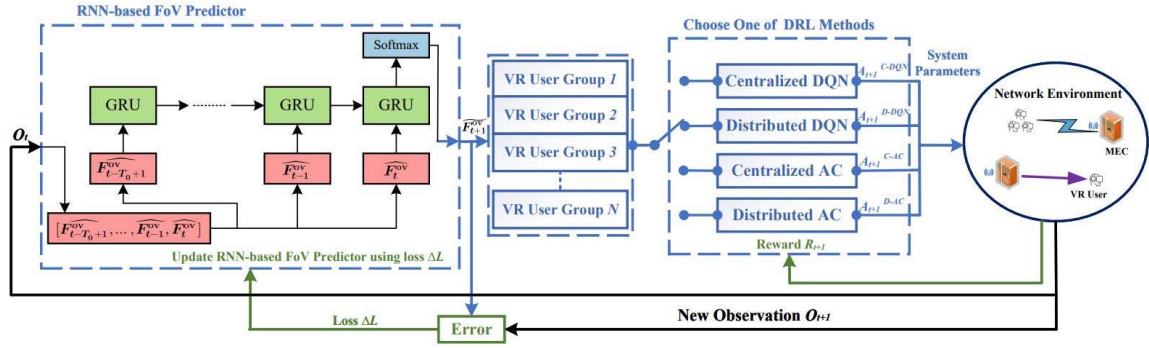
Fig. 5. Decoupled learning strategy for MEC rendering schemes in the wireless VR network.

III. DEEP REINFORCEMENT LEARNING-BASED MEC RENDERING SCHEME

Knowing the deep neural networks as one of the most impressive non-linear approximation functions, DRL is an effective method to optimally solve POMDP problems [21]. In this section, to solve (P1), a decoupled learning strategy is proposed for FoV prediction and MEC rendering association, as shown in Fig. 5. Specifically, a RNN model based on GRU is used to predict FoV preference of each user over time. Then, four DRL algorithms, including centralized DQN, distributed DQN, centralized AC, and distributed AC, are proposed to select the FoV rendering MEC and the associated MEC for downlink transmission.

A. FoV Prediction

Brownian Motion is used to simulate the eye movement of VR users over time, and assuming that the uplink received FoV preference of the kth VR user at the tth time slot is $\widehat{F_t^{\text{oV}}}^k \in \{1, 2, \ldots, N_{\text{FoV}}\}$. In order to detect dynamics in FoV preference of each VR user, the proposed learning scheme aims at utilizing not only the information presents in the most recent observation $O_t = \{O_t^1, O_t^2, \ldots, O_t^{K_{\text{VR}}}\}$, where $O_t^k = \{\widehat{F_t^{\text{oV}}}^k\}$, but also the historical information in the previous observations $H_t = \{O_{t-T_0+1}, \ldots, O_{t-2}, O_{t-1}\}$ given a memory window T_0. To recognize FoV preference over time, a RNN model with parameters $\boldsymbol{\theta}_{\text{RNN}}$, and specifically a GRU architecture, is leveraged. $\boldsymbol{\theta}_{\text{RNN}}$ is consisted of both the GRU internal parameters and the weights of the softmax layer. We choose RNN due to its ability in capturing time correlation of FoV preference over time, which can help learn the time-varying FoV preference for better prediction accuracy.

As shown in Fig. 5, the GRU layer includes multiple standard GRU units and historical observations $[O_{t-T_0+1}, \ldots, O_{t-1},]$ sequentially inputted into the RNN predictor. For the kth VR user, the GRU layer is connected to an output layer which is consisted of a softmax non-linearity with N_{FoV} output values, which represents the predicted probability $\mathcal{P}\{\widehat{F_t^{\text{oV}}}^k = \hat{f}|[O_{t-T_0+1}^k, \ldots, O_{t-1}^k], \boldsymbol{\theta}_{\text{RNN}}\}$ of the $\hat{f}$th FoV ($\hat{f} = 1, \ldots, N_{\text{FoV}}$) for the tth time slot given historical observations $[O_{t-T_0+1}^k, \ldots, O_{t-1}^k]$.

To adapt the model parameter $\boldsymbol{\theta}_{\text{RNN}}$, standard Stochastic Gradient Descent (SGD) via BackPropagation Through Time (BPTT) [42] is deployed. At the $(t+1)$th time slot, the parameters $\boldsymbol{\theta}_{\text{RNN}}$ of the RNN predictor can be updated as

$$\boldsymbol{\theta}_{\text{RNN}}^{t+1} = \boldsymbol{\theta}_{\text{RNN}}^t - \lambda_{\text{RNN}} \nabla L_{\text{RNN}}(\boldsymbol{\theta}_{\text{RNN}}^t), \quad (21)$$

where $\lambda_{\text{RNN}} \in (0, 1]$ is the learning rate, $\nabla L_{\text{RNN}}(\boldsymbol{\theta}_{\text{RNN}}^t)$ is the gradient of the loss function $L_{\text{RNN}}(\boldsymbol{\theta}_{\text{RNN}}^t)$ to train the RNN predictor. $L_{\text{RNN}}(\boldsymbol{\theta}_{\text{RNN}}^t)$ can be obtained by averaging the cross-entropy loss as

$$L_{\text{RNN}}^t(\boldsymbol{\theta}_{\text{RNN}}) = -\sum_{t'=t-T_b+1}^{t} \log\left(\mathcal{P}\{\widehat{F_{t'}^{\text{oV}}} = \widetilde{F_{t'}^{\text{oV}}}|O_{t'-T_0}^{t'}, \boldsymbol{\theta}_{\text{RNN}}\}\right), \quad (22)$$

where

$$O_{t'-T_0}^{t'} = [O_{t'-T_0+1}, \ldots, O_{t'-1}, O_{t'}], \quad (23)$$

and T_b is the randomly selected mini-batch size.

Through FoV prediction, MECs are able to know the FoV preference of each VR user in advance. The VR users with the same predicted FoVs can be grouped together. After FoV rendering, the MECs will multicast or unicast the required FoVs to VR users selecting the same FoV, or a single VR user selecting unique FoV, respectively.

B. Deep Reinforcement Learning

The main purpose of Reinforcement Learning (RL) is to select proper MECs for MEC rendering schemes. Through a series of action strategies, MECs are able to interact with the environment, and obtain rewards due to their actions, which help to improve their action strategies. After plenty of iterations, MECs can learn the optimal policy that maximizes the long-term rewards.

We define $S \in \mathcal{S}$, $A \in \mathcal{A}$, and $R \in \mathcal{R}_e$ as any state, action and reward from their corresponding sets, respectively. According to the observed environmental state S_t at the tth time slot, MECs choose specific actions A_t from the set $\mathcal{A}$ and receive rewards R_t, which are regarded as a metric to measure whether the selected actions are good. Thus, the purpose of RL algorithm is to find an optimal policy π which can maximize the long-term reward for $A = \pi(S)$. The optimization function can be formulated as $<S, A, R>$ and the detailed descriptions

of the state, action and reward of problem (P1) are introduced as follows.

- *State:* At the tth time slot, the network state can be denoted as

$$S_t = (\widetilde{\mathcal{F}_t^{\text{oV}}}, \mathcal{L}_{k,i}^t, \mathcal{F}_i^{\text{MEC}}) \in \mathcal{S},$$

$$\text{with } \widetilde{\mathcal{F}_t^{\text{oV}}} = \{\widetilde{F_t^{\text{oV}}}^1, \widetilde{F_t^{\text{oV}}}^2, \dots, \widetilde{F_t^{\text{oV}}}^{K_{\text{VR}}}\},$$

$$\mathcal{L}_{k,i}^t = \{l_{k,1}^t, l_{k,2}^t, \dots, l_{k,B}^t, \},$$

$$\mathcal{F}_i^{\text{MEC}} = \{F_1^{\text{MEC}}, F_2^{\text{MEC}}, \dots, F_B^{\text{MEC}}\}, \quad (24)$$

where $\widetilde{F_t^{\text{oV}}}^k$ is the index of the predicted FoV of the kth VR user at the tth time slot. $l_{k,i}^t$ is the distance between the kth VR user and the ith MEC at the tth time slot. F_i^{MEC} is the computation capability of the ith MEC.

- *Action:* The action space can be written as

$$A_t = \{\check{\mathcal{A}}_{k,q}^t, \acute{\mathcal{A}}_{k,i}^t\} \in \mathcal{A},$$

$$\text{with } \check{\mathcal{A}}_{k,q}^t = \{\check{A}_{k,1}, \check{A}_{k,2}, \dots, \check{A}_{k,N_{\text{FoV}}}\},$$

$$\acute{\mathcal{A}}_{k,i}^t = \{\acute{A}_{k,1}, \acute{A}_{k,2}, \dots, \acute{A}_{k,K_{\text{VR}}}\}, \quad (25)$$

where $\check{A}_{k,q}^t \in \{0,1\}$ and $\acute{A}_{k,i}^t \in \{0,1\}$ represent whether the kth MEC will render the qth FoV and serve the ith VR user at the tth time slot, respectively. For instance, if $\check{A}_{k,q}^t = 1$ and $\check{A}_{k,j}^t = 0$, the kth MEC will render and migrate the qth FoV to the jth ($j \neq k$) MEC choosing the same FoV. If $\acute{A}_{k,i}^t = 1$, the kth MEC will support the downlink transmission of the ith VR user, otherwise, not.

- *Reward:* The immediate reward R_t is designed as

$$R_t(S_t, A_t) = \sum_{k=1}^{K_{\text{VR}}} \text{PSNR}_k^t. \quad (26)$$

Thus, the discounted accumulation of the long-term reward can be denoted as

$$V(S, \pi) = \sum_{i=t}^{\infty} (\gamma)^{i-t} R_i(S_i, A_i), \quad (27)$$

where $\gamma \in [0, 1)$ is the discount factor.

When the number of MECs and VR users are small, RL algorithm can efficiently obtain the optimal policy. However, when a large number of MECs and VR users exist, the state and action spaces will be scaled proportionally, which will inevitably result in massive computation latency and severely affect the performance of the RL algorithm. To address this issue, deep learning is introduced to RL, namely, deep reinforcement learning (DRL), through interaction with the environment, DRL can directly control the behavior of each agent, and solve complex decision-making problem. In DRL algorithm, two methods can be used to obtain optimal policy. One is called value-based optimization, such as DQN, which indirectly optimizes the policy by optimizing value function. While the other is policy-based optimization, such as AC, which can directly optimize the policy. In the following sections, four DRL algorithms are introduced in detail.
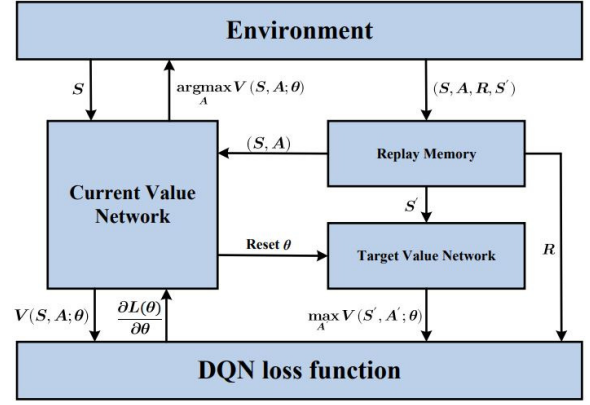

Fig. 6. The DQN diagram of the MEC rendering scheme.

1) Centralized DQN: As a value-based DRL algorithm, DQN combines a neural network with Q-learning and approximates the state-action value function via the deep neural network (DNN). Using DQN algorithm, a fraction of states are sampled and the neural network is applied to train a sufficiently accurate state-action value function, which is able to effectively solve the problem of high dimensionality in state space. Furthermore, the DQN algorithm uses the experience replay to train the learning process of RL. When updating the DQN algorithm, some experiences in the experience replay will be selected randomly to learn, so that the correlation among the training samples can be broke and the efficiency of the neural network can be improved. In addition, through averaging the selected samples, the distribution of training samples can be smoothed, which avoids the training divergence.

As shown in Fig. 6, the action-state value function $V_{\text{DQN}}(S, A)$ in the DQN agent can be parameterized by using a function $V_{\text{DQN}}(S, A; \boldsymbol{\theta}_{\text{DQN}})$, where $\boldsymbol{\theta}_{\text{DQN}}$ is the weight matrix of the DNN with multiple layers. Consider the conventional DNN, where the neurons between two adjacent layers are fully connected, which is so-called fully-connected layers. The input of the DNN is the variables in state S_t; the hidden layers are Rectifier Linear Units (ReLUs) through utilizing the function $f(x) = \max(0, x)$; the output layer is consisted of linear units, which are all available actions in A_t. The exploitation is obtained by performing propagation of $V_{\text{DQN}}(S, A; \boldsymbol{\theta}_{\text{DQN}})$ with respect to the observed state S_t. Moreover, the parameter $\boldsymbol{\theta}_{\text{DQN}}$ can be updated by using SGD as

$$\boldsymbol{\theta}_{\text{DQN}}^{t+1} = \boldsymbol{\theta}_{\text{DQN}}^t - \lambda_{\text{DQN}} \nabla L_{\text{DQN}}(\boldsymbol{\theta}_{\text{DQN}}^t), \quad (28)$$

where $\lambda_{\text{DQN}} \in (0, 1]$ is the learning rate, $\nabla L_{\text{DQN}}(\boldsymbol{\theta}_{\text{DQN}}^t)$ is the gradient of the loss function $L_{\text{DQN}}(\boldsymbol{\theta}_{\text{DQN}}^t)$ utilized to train the state-action value function. The loss function can be defined as

$$L_{\text{DQN}}(\boldsymbol{\theta}_{\text{DQN}}^t) = (\hat{V}_{\text{DQN}} - V_{\text{DQN}}(S_i, A_i; \boldsymbol{\theta}_{\text{DQN}}^t))^2, \quad (29)$$

where

$$\hat{V}_{\text{DQN}} = R_{i+1} + \gamma \max_A V_{\text{DQN}}(S_{i+1}, A; \bar{\boldsymbol{\theta}}_{\text{DQN}}^t). \quad (30)$$

$(S_i, A_i, S_{i+1}, R_{i+1})$ are randomly selected previous samples for some $i \in \{t - M_r, \dots, t\}$ with respect to a so-called minibatch. M_r is the replay memory. $\bar{\boldsymbol{\theta}}_{\text{DQN}}^t$ is the so-called

Algorithm 1 DQN to Dynamic Decision-Making and Optimization of the MEC Rendering Scheme

---

1: Initialize replay memory D to capacity $\hat{\mathcal{N}}$, learning rate $\lambda_{\text{DQN}} \in (0,1]$ and discount factor $\gamma \in [0,1)$.
2: Initialize state-action value function $V_{\text{DQN}}(S, A; \boldsymbol{\theta}_{\text{DQN}})$, the parameters of primary Q-network $\boldsymbol{\theta}_{\text{DQN}}$ and target Q-network $\bar{\boldsymbol{\theta}}_{\text{DQN}}$.
3: **for** episode $= 1, \ldots, M$ **do**
4:    Input the network state S of the MEC rendering scheme.
5:    **for** t $= 1, \ldots,$T **do**
6:       Use ϵ-greedy algorithm to select a random action A_t from action space $\mathcal{A}$.
7:       Otherwise, select $A_t = \max_{A \in \mathcal{A}} V(S_t, A; \boldsymbol{\theta}_{\text{DQN}})$.
8:       The selected MECs render the predicted or uplink received FoVs and multicast/unicast them to VR users according to the selected action A_t.
9:       MECs observe reward R_t and new state S_{t+1}.
10:     Store transition (S_t, A_t, R_t, S_{t+1}) in replay memory D.
11:     Sample random minibatch of transitions (S_j, A_j, R_j, S_{j+1}) from replay memory D.
12:     **if** $j + 1$ is terminal **then**
13:       $y_j^{target} = R_j$.
14:     **else**
15:       $y_j^{target} = R_{j+1} + \gamma \max_A V_{\text{DQN}}(S_{j+1}, A; \boldsymbol{\theta}_{\text{DQN}})$.
16:     **end if**
17:     Perform a gradient descent step and update parameters $\boldsymbol{\theta}_{\text{DQN}}$ according to (28).
18:     Update parameter $\bar{\boldsymbol{\theta}}_{\text{DQN}}$ of the target network every $\bar{K}$ steps.
19:    **end for**
20: **end for**

---

target Q-network which is utilized to estimate the future value of the Q-function in the update rule. Meanwhile, $\bar{\boldsymbol{\theta}}_{\text{DQN}}^t$ is periodically copied from the current value $\boldsymbol{\theta}_{\text{DQN}}^t$ and kept fixed for some episodes. The use of minibatch, rather than a single sample, to update the state-action value function $V_{\text{DQN}}(S, A; \boldsymbol{\theta}_{\text{DQN}})$ is able to improve the convergent reliability of value function.

Through deriving the loss function in (29) and calculating the expectation of the selected previous samples in minibatch, $V_{\text{DQN}}^*(S, A)$ can be obtained. The DQN algorithm is presented in Algorithm 1.

2) Distributed DQN: In the centralized DRL algorithm, it learns a single optimization policy centrally at the central controller, which requires the global observations, rewards, and actions of each MEC. When the number of MECs and VR users increase, the size of the proposed model and parameters can expand exponentially. In this case, the GPU memory in central controller not only needs to hold the model and batch of data, but also the intermediate outputs of the feedforward computation. With dense VR users, GPU memory can be easily overloaded in practice, especially for the GPUs with lower computation capability. Meanwhile, as the number of MECs and VR users scaling up, the centralized

DRL can become inefficient due to the following issues. First, the training time is bound by the gradient computation time, and the frequency of parameter updating grows linearly with the number of MECs and VR users. Second, as the frequency of parameter updating grows, it could potentially slow down the optimization process and result in problems with convergence [43].

Unlike the centralized DRL algorithm, the global objective in distributed DRL algorithm is the combination of each agent's local objective, and each agent needs to optimize its own objective. In the distributed DQN method, each agent learns independently from the other agents. When one of the agents selects an action based on the current state, the other agents can be approximated as part of the environment [44].

In our model, the central controller stores a copy of the model parameter $\boldsymbol{\theta}_{\text{DDQN}}$. The ith MEC obtains the latest model parameter $\boldsymbol{\theta}_{\text{DDQN}}$ from the central controller with $\widetilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_{\text{DDQN}}$. Based on the observed state S_t^i, it will select an action A_t^i in all available actions in $\mathcal{A}^i$. As a result, the environment will make a transition to the new state S_{t+1}^i and a reward R_t^i will be generated and fed back to the ith MEC. During training, the parameter $\widetilde{\boldsymbol{\theta}}_i$ of the ith MEC can be updated as

$$\widetilde{\boldsymbol{\theta}}_i^{t+1} = \widetilde{\boldsymbol{\theta}}_i^t - \lambda_{\text{DDQN}} \nabla L_i(\widetilde{\boldsymbol{\theta}}_i^t), \tag{31}$$

where $\lambda_{\text{DDQN}} \in (0,1]$ is the learning rate, $L_i(\widetilde{\boldsymbol{\theta}}_i^t)$ is the loss function of the ith MEC, which can be denoted as

$$L_i(\widetilde{\boldsymbol{\theta}}_i^t) = (\hat{V}_{\text{DDQN}} - V_{\text{DDQN}}(S_j^i, A_j^i; \widetilde{\boldsymbol{\theta}}_i^t))^2, \tag{32}$$

where

$$\hat{V}_{\text{DDQN}} = R_{j+1}^i + \gamma \max_{A^i} V_{\text{DDQN}}(S_{j+1}^i, A^i; \bar{\widetilde{\boldsymbol{\theta}}}_i^t). \tag{33}$$

$(S_j^i, A_j^i, S_{j+1}^i, r_{j+1}^i)$ are randomly selected previous samples for $j \in \{t - M_r, .., t\}$ of the ith MEC. $\bar{\widetilde{\boldsymbol{\theta}}}_i^t$ is the target Q-network which is used to estimate the future value of the state-action value function in the update rule. Furthermore, through deriving the loss function in (32) and computing the expectation of the selected samples, $V_{\text{DDQN}}^*(S^i, A^i)$ can be obtained. In addition, the updated parameter $\widetilde{\boldsymbol{\theta}}_i$ of the ith MEC will be transmitted to the central controller and the model parameter $\boldsymbol{\theta}_{\text{DDQN}}$ can be updated as

$$\boldsymbol{\theta}_{\text{DDQN}} = \frac{1}{K_{\text{DDQN}}^{\text{MEC}}} \sum_{i=1}^{K_{\text{DDQN}}^{\text{MEC}}} \widetilde{\boldsymbol{\theta}}_i, \tag{34}$$

where $K_{\text{DDQN}}^{\text{MEC}}$ is the number of the MECs associated with the VR user groups.

3) Centralized AC: In the DQN algorithm, the optimal policy of the MEC rendering scheme is indirectly obtained through optimizing the state-action value function. However, unlike the DQN algorithm, AC algorithm is able to directly optimize the policy of the MEC rendering scheme.

The core idea of the AC algorithm is to combine the advantages of Q-learning (value-based function) and the policy-gradient (policy-based function) algorithms. Consequently, the fast convergence of the value-based function and the directness of the policy-based function are all taken into
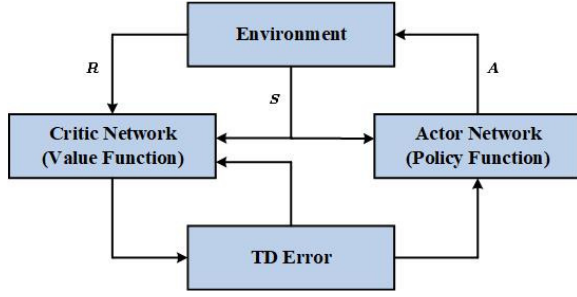
Fig. 7. The Actor-Critic diagram of the MEC rendering scheme.

consideration [45], [46]. As shown in Fig. 7, the AC network is consisted of two independent networks, namely, an actor network and a critic network. Through learning the relationship between the environment and the rewards, the critic network is able to get the potential rewards of the current state. Then, the critic network will guide the actor network to select proper actions and update the actor network in each epoch. Therefore, the AC algorithm is usually developed as a two-time-scale algorithm, including critic updating step and actor updating step, which leads to slow learning efficiency. A parameterized policy $\pi(A_t|S_t; \boldsymbol{\theta}_{\text{AC}})$ is learned to select actions according to the current environment state. Then, the critic network will obtain the reward feedback from the environment and use the state-value function $V_{\text{AC}}(S_t; \boldsymbol{w}_{\text{AC}})$ to evaluate the performed action. Meanwhile, a time-difference (TD) error is generated to reflect the performance of the performed action.

In particular, after performing action A_t based on S_t with policy π, the critic network uses TD error to evaluate the action under the current state, which can be expressed as

$$\delta_t = R_t + \gamma V_{\text{AC}}(S_{t+1}; \boldsymbol{w}_{\text{AC}}^t) - V_{\text{AC}}(S_t; \boldsymbol{w}_{\text{AC}}^t). \quad (35)$$

Then, $\boldsymbol{w}_{\text{AC}}^t$ can be updated as

$$\boldsymbol{w}_{\text{AC}}^{t+1} = \boldsymbol{w}_{\text{AC}}^t + \lambda_{\text{critic}} \delta_t \nabla_{\boldsymbol{w}_{\text{AC}}} V_{\text{AC}}(S_t; \boldsymbol{w}_{\text{AC}}^t). \quad (36)$$

where $\lambda_{\text{critic}} \in (0,1]$ is the learning rate of the critic network.

Meanwhile, in the actor network, the policy gradient method is usually adopted, which directly selects actions via parameterized policy. The parameter $\boldsymbol{\theta}_{\text{AC}}^t$ can be updated as

$$\boldsymbol{\theta}_{\text{AC}}^{t+1} = \boldsymbol{\theta}_{\text{AC}}^t + \lambda_{\text{actor}} \delta_t \nabla_{\boldsymbol{\theta}_{\text{AC}}} \log \pi(A_t|S_t; \boldsymbol{\theta}_{\text{AC}}^t), \quad (37)$$

where $\lambda_{\text{actor}} \in (0,1]$ is the learning rate of the actor network.

Correspondingly, the parameters in the actor and critic network will be iteratively updated to maximize the objective function. The detailed AC algorithm of MEC rendering scheme is proposed in Algorithm 2.

4) Distributed AC: Unlike the centralized AC algorithm, the agent in the distributed AC algorithm performs action and obtains reward based on its own observed state. For the critic network in each agent, it shares its estimate of the value function with others through the central controller. While for the actor network in each agent, it performs individually without the need to infer the policies of others [47].

In our model, the ith MEC obtains the latest critic model parameter $\boldsymbol{w}_{\text{DAC}}$ from the central controller, and let its own critic parameter $\bar{\boldsymbol{w}}_t^i = \boldsymbol{w}_{\text{DAC}}$. At the tth time slot, according to the current environment state S_t^i obtained by the ith MEC,

---

Algorithm 2 Actor-Critic to Dynamic Decision-Making and Optimization of Wireless VR System

1: Initialize learning rate $\lambda_{\text{critic}} \in (0,1]$, $\lambda_{\text{actor}} \in (0,1]$ and discount factor $\gamma \in [0,1)$.
2: Initialize parameters $\boldsymbol{\theta}_{\text{AC}}$ and $\boldsymbol{w}_{\text{AC}}$ for the actor and critic network, respectively.
3: Input the network state S of the MEC rendering scheme.
4: **for** t = 1, . . . ,T **do**
5:     According to $\pi(A|S_t; \theta)$, select the action $A \in \mathcal{A}$.
6:     The selected MECs render the required FoVs and multi-cast/unicast them to VR users due to the selected action A_t.
7:     MECs calculate the immediate reward R_t and obtain the environment state S_{t+1}.
8:     Store transition (S_t, A_t, R_t, S_{t+1}).
9:     Calculate TD error δ_t according to (35).
10:     Update the parameters $\boldsymbol{w}_{\text{AC}}$ of the critic network via (36).
11:     Update the parameters $\boldsymbol{\theta}_{\text{AC}}$ of the actor network via (37).
12: **end for**

---

a parameterized policy $\pi_i(S_t^i; \bar{\boldsymbol{\theta}}_t^i)$ is learned to select action A_t^i. Then, the critic network in the ith MEC will receive the reward feedback by the environment and evaluate the state-value function $V_{\text{DAC}}(A_t^i|S_t^i; \bar{\boldsymbol{w}}_t^i)$. Similarly, the TD error δ_t^i of the ith MEC can be calculated to judge the performance of the performed action A_t^i, and the parameter $\bar{\boldsymbol{w}}_t^i$ of the critic network of the ith MEC can be updated as

$$\bar{\boldsymbol{w}}_{t+1}^i = \bar{\boldsymbol{w}}_t^i + \bar{\lambda}_{\text{critic}} \delta_t^i \nabla_{\bar{\boldsymbol{w}}^i} V_{\text{DAC}}(S_t^i; \bar{\boldsymbol{w}}_t^i), \quad (38)$$

where $\bar{\lambda}_{\text{critic}} \in (0,1]$ is the learning rate of the critic network, and

$$\delta_t^i = R_t^i + \gamma V_{\text{DAC}}(S_{t+1}^i; \bar{\boldsymbol{w}}_t^i) - V_{\text{DAC}}(S_t^i; \bar{\boldsymbol{w}}_t^i). \quad (39)$$

Furthermore, for the parameter $\bar{\boldsymbol{\theta}}_t^i$ of the actor network of the ith MEC, it can be updated via

$$\bar{\boldsymbol{\theta}}_{t+1}^i = \bar{\boldsymbol{\theta}}_t^i + \bar{\lambda}_{\text{actor}} \delta_t^i \nabla_{\bar{\boldsymbol{\theta}}^i} \log \pi(A_t^i|S_t^i; \bar{\boldsymbol{\theta}}_t^i), \quad (40)$$

where $\bar{\lambda}_{\text{actor}} \in (0,1]$ is the learning rate of the actor network. In addition, the updated parameter $\bar{\boldsymbol{w}}_t^i$ in the critic network of the ith MEC will be sent to the central controller and the critic model parameter $\boldsymbol{w}_{\text{DAC}}$ can be updated as

$$\boldsymbol{w}_{\text{DAC}} = \frac{1}{K_{\text{DAC}}^{\text{MEC}}} \sum_{i=1}^{K_{\text{DAC}}^{\text{MEC}}} \bar{\boldsymbol{w}}^i, \quad (41)$$

where $K_{\text{DAC}}^{\text{MEC}}$ is the number of the MECs associated with the VR user groups. Correspondingly, the parameters in the actor and critic network will be iteratively updated to maximize the objective function.

C. Computational Complexity and Implementation Analysis

For the computation complexity of the RNN based on the GRU architecture, it can be computed as $O(\widetilde{m}\widetilde{n} \log \widetilde{n})$, where $\widetilde{m}$ is the number of layers, and $\widetilde{n}$ is the number of units per learning layer. The computational complexity of the centralized DQN/AC algorithm, which includes DQN/AC
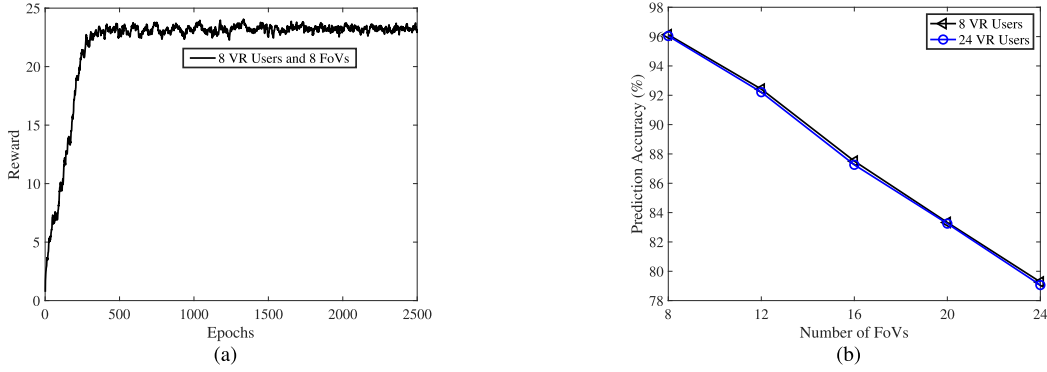
Fig. 8. (a) Total reward of FoV prediction of each epoch via GRU with 8 VR users and 8 FoVs. (b) FoV prediction accuracy of GRU with 8 and 24 VR users for varying number of FoV.

learning architecture, the MEC migration scheme, and the uplink and downlink transmission, are given by $O(mn \log n + N_{\text{migration}} + N_{\text{MEC}}K_{\text{VR}})$. Here, m is the number of layers, n is the number of units per learning layer, $N_{\text{migration}}$ is the number of MECs selected for rendering migration, and N_{MEC} and K_{VR} are the number of MECs and VR users, respectively. For the distributed DQN/AC algorithm, the computational complexity of the algorithms can be given by $O(N_{\text{MEC}}\hat{m}\hat{n} \log \hat{n} + N_{\text{MEC}}K_{\text{VR}})$, where $\hat{m}$ is the number of layers in each MEC, and $\hat{n}$ is the number of units per layer in each MEC [2] [48]. For the implementation of the proposed algorithms, according to Fig. 5, at the tth time slot, the RNN-based FoV predictor will predict the FoV preference of each VR user for the $(t + 1)$th time slot. The centralized and distributed DRL algorithms will select proper MECs to render and transmit the required FoVs to VR users based on the observed state.

IV. SIMULATION RESULTS

In this section, we examine the effectiveness of our proposed schemes with learning algorithms via simulation. For the learning algorithms, we set the learning algorithms use fully-connected neural network with two hidden layers and each layer has 128 ReLU units. The simulation parameters are summarized in Table I.

A. FoV Prediction

In the FoV prediction scheme, Brownian motion is deployed to simulate the eye movement of VR users. To obtain high accuracy in predicting FoV preference of each VR user in continuous time slots, a RNN model based on GRU architecture is deployed. Fig. 8 (a) plots the total reward of FoV prediction of each epoch via GRU with 8 VR users and 8 FoVs, and Fig. 8 (b) plots the FoV prediction accuracy of GRU with 8 and 24 VR users for varying number of FoVs. It is observed that the prediction accuracy decreases with increasing number of FoVs. This is because the eye

---

[2]The proposed decoupled learning strategy is an online learning strategy. It can effectively adapt to the dynamic wireless VR environment through observing the state of the network at each time slot, which can be used for real-time control in VR video streaming.
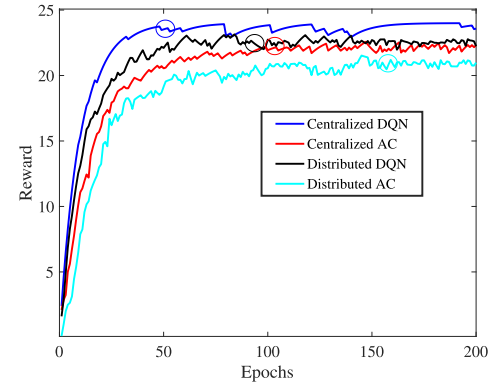

Fig. 9. Total reward of the MEC rendering with prediction and migration scheme of each epoch via centralized/distributed DQN/AC learning algorithms.

movements follow Brownian motion, and the randomness of FoV selection increases with increasing number of FoVs, which reduces the prediction accuracy. We also observe that the prediction accuracy changes slightly with the same number of FoVs for different number of VR users. This is because RNN utilizes a memory window with length 20 for each VR user to store with the input observations, which can capture the FoV preference of VR users over time.

B. MEC Rendering Scheme

Four DRL algorithms, including centralized DQN, distributed DQN, centralized AC, and distributed AC, are proposed to select proper MECs to render and transmit the required FoVs to VR users. For simplicity, we use "w/ Pred", "w/o Pred", "w/ Migra", and "w/o Migra" to represent "with prediction", "without prediction", "with migration", and "without migration" in the figures, respectively. To guarantee the fairness of each VR user, we use average QoE and VR interaction latency in the performance results.

Fig. 9 plots the total reward of the MEC rendering with prediction and migration scheme of each epoch via centralized/distributed DQN/AC learning algorithms. Each result is averaged over 100 training trails. It is observed that the total reward and the convergence speed of these four DRL learning algorithms follows: Centralized DQN > Distributed

TABLE I
SIMULATION PARAMETERS OF WIRELESS VR NETWORK

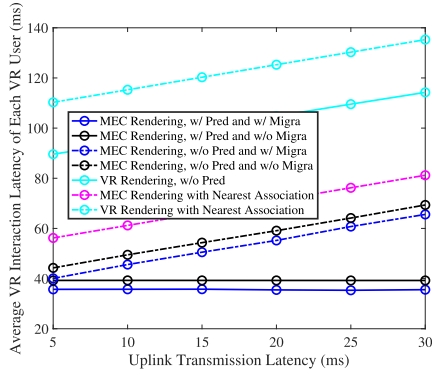| LSTM memory size	20	Minibatch size	64
RNN learning rate	0.005	Number of MECs	8
Number of VR users	8	N_{FoV}	8
$D(t)$	3	σ^2	-110 dBm
γ	0.9	α, β	3
λ_{DQN}	0.05	λ_{actor}	0.005
λ_{critic}	0.05	T^{th}	30 ms
N_p	1080p	$\mathcal{C}_{\mathcal{R}}$	200
F^{MEC}_{max}	5 GHz	F^{MEC}_{min}	4 GHz
F^{VR}	2 GHz	$f^{\text{MEC}}, f^{\text{VR}}$	1000 Cycles/bit
R^{fiber}	10 Gb/s	Side length of the limited square	100 meters

Fig. 10. Average VR interaction latency of various MEC and VR rendering schemes via centralized DQN algorithm for varying uplink transmission latency.
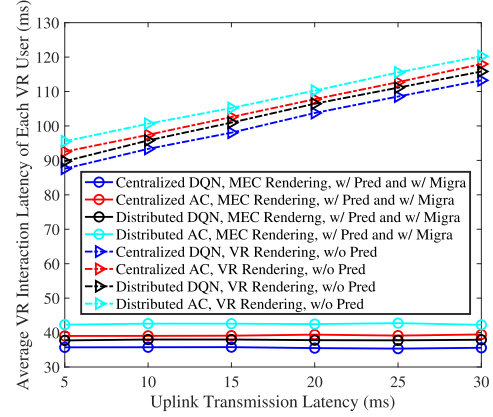

Fig. 11. Average VR interaction latency of the MEC rendering with prediction and migration scheme and the VR rendering scheme via centralized/distributed DQN/AC learning algorithms for varying uplink transmission latency.

DQN > Centralized AC > Distributed AC. This is due to the experience replay mechanism and randomly sampling in DQN, which use the training samples efficiently and smooth the training distribution over the previous behaviours. As the model parameters in AC algorithm are updated in two steps, including critic step and actor step, the convergence speed of the AC algorithm is lower. Apparently, the convergence speed of the centralized learning algorithms is faster than that of the distributed learning algorithms. This is because the distributed learning needs more time to learn from each agent with only local observation and reward, whereas centralized learning can learn from global observations and rewards.

Fig. 10 plots the average VR interaction latency of various MEC and VR rendering schemes via centralized DQN algorithm for varying uplink transmission latency. We observe that all the MEC rendering schemes outperform that of the VR rendering schemes, with around 40 ms gain. This is because the processing ability of the MECs is much higher than that of the VR devices, and the data size of the FoV is smaller than that of the stitched 2D picture, which jointly decrease the rendering and downlink transmission latency. We also observe that the average VR interaction latency of the MEC rendering with prediction and migration scheme remains the same with increasing the uplink transmission latency, as the MECs do not need to wait for the uplink transmission of requested FoV from the VR devices before performing rendering.

In Fig. 10, we also compare our proposed learning-based schemes with those without learning. By comparing with the MEC/VR rendering scheme with nearest association scheme plotted using dash lines, we see our proposed learning-based MEC/VR rendering schemes achieve substantial gain in terms of VR interaction latency. This is due to that in the non-learning scheme, the VR user needs to transmit its requested FoV through uplink transmission and is always associated with the nearest MEC. Thus, it is possible that the MEC with low processing ability is selected to render the required FoV, which can increase the rendering latency.

Fig. 11 plots the average VR interaction latency of the MEC rendering with prediction and migration scheme and the VR rendering scheme via centralized/distributed DQN/AC learning algorithms for varying uplink transmission latency. It is observed that for the MEC rendering scheme achieves much lower latency (about 40 ms) compared to VR rendering scheme. It is also seen that for the same rendering scheme, either the MEC or VR, the centralized DQN algorithm can achieve the minimum average VR interaction latency. This can be explained by the fact that the centralized learning algorithm learns a single policy common to the whole wireless VR system based on the global observations, while in the distributed learning algorithm, each agent only learns its own policy based on local observation.
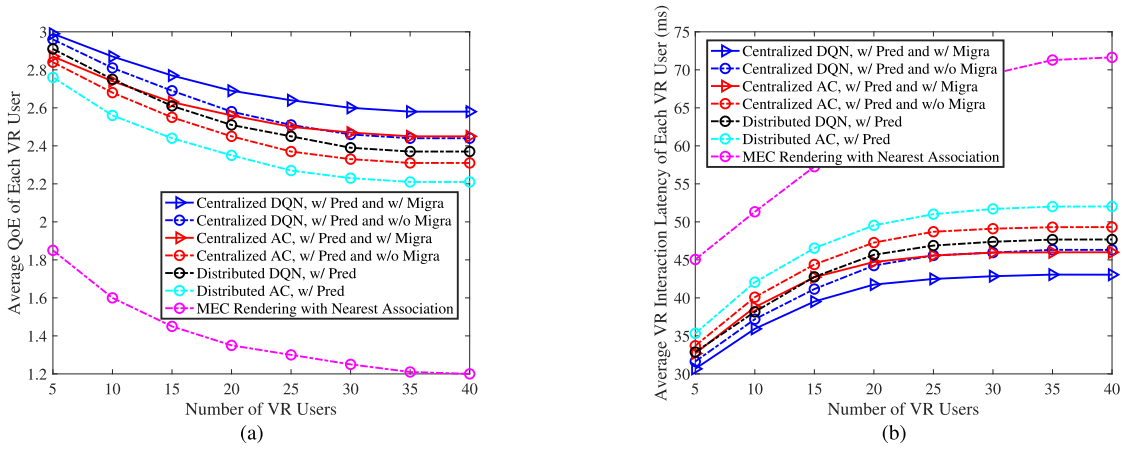
Fig. 12.   Average QoE and VR interaction latency of MEC rendering with prediction with/without migration schemes via centralized/distributed DQN/AC learning algorithms with increasing number of VR users.
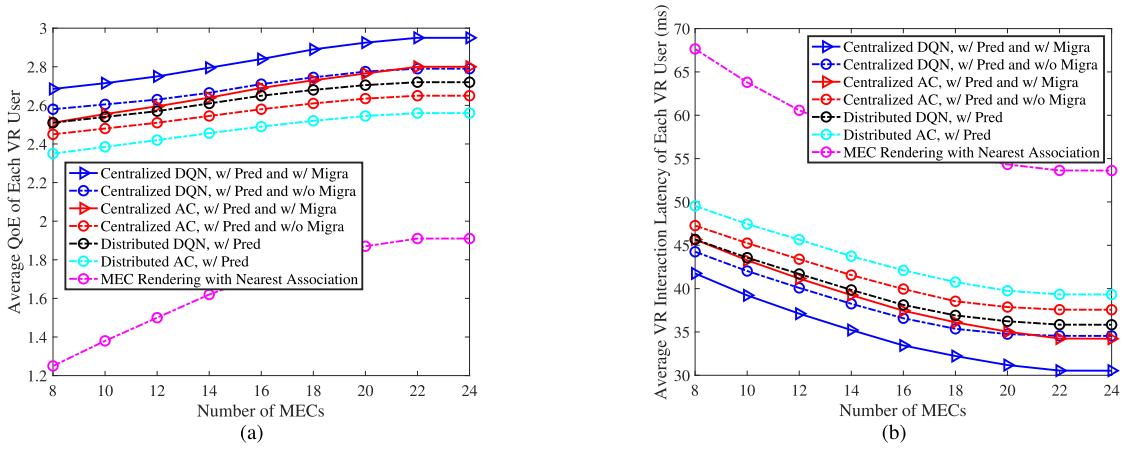

Fig. 13.   Average QoE and VR interaction latency of MEC rendering with prediction with/without migration schemes via centralized/distributed DQN/AC learning algorithms with increasing number of MECs.

Fig. 12 plots the average QoE and VR interaction latency of MEC rendering with prediction with/without migration schemes via centralized/distributed DQN/AC learning algorithms with increasing number of VR users, respectively. With increasing the number of VR users, the average QoE of VR user first decreases then becomes nearly stable as shown in Fig. 12 (a), whereas the average VR interaction latency first increases, then becomes nearly stable as shown in Fig. 12 (b). This is because with increasing number of VR users, more MECs are activated to provide the downlink transmission for more requested different FoVs, which increase the interference among those transmission. As the number of VR users becomes too large, all the MECs become active to serve all VR users to render most of FoVs, the rendering latency and interference among VR users become stable.

Interestingly, we notice that for both centralized DQN and AC algorithms, we can see the performance gain of the MEC rendering with migration scheme over that without migration scheme in Fig. 12 (a) and (b). This is because the MECs with higher computing ability will be selected to render the same required FoV for migration, which decreases the rendering latency. Importantly, all the learning-based MEC

rendering with prediction schemes substantially outperform the conventional non-learning based MEC rendering with nearest association scheme.

Fig. 13 plots the average QoE and VR interaction latency of MEC rendering with prediction with/without migration schemes via centralized/distributed DQN/AC learning algorithms with increasing number of MECs, respectively. With increasing the number of MECs, the average QoE of VR user first increases then becomes nearly stable as shown in Fig. 13 (a), whereas the average VR interaction latency first decreases, then becomes nearly stable as shown in Fig. 13 (b). This is because as the number of MECs increases, the VR users will have more MEC choices to be selected, thus, nearer MECs with higher execution ability can be utilized to render the required FoVs, which reduces the rendering and downlink transmission latency. However, as the number of MECs becomes too large, all MECs may be activated for rendering and downlink transmission, which leaves little gain for improvement.

V. CONCLUSION AND FUTURE WORK

In this paper, a decoupled learning strategy was developed to optimize real-time VR video streaming in wireless network,

which considered FoV prediction and rendering MEC association. Specifically, based on GRU architecture, a RNN model was used to predict FoV preference of each VR user over time. Then, based on the correlation between the location and predicted FoV request of VR users, centralized and distributed DRL strategies were proposed to determine the optimal association between MEC and VR user group, and optimal rendering MEC for model migration, so as to maximize the long-term QoE of VR users. Simulation results shown that our proposed MEC rendering with prediction and migration scheme based on RNN and DRL algorithms substantially improved the long-term QoE of VR users and the VR interaction latency.

REFERENCES

[1] Facebook. *Oculus Rift*. Accessed: Jul. 22, 2019. [Online]. Available: https://www.oculus.com/

[2] (Oct. 2017). *Virtual Reality and Augmented Reality Device Sales to Hit 99 Million Devices in 2021*. [Online]. Available: https://www.capacitymedia.com/articles/3755961/VR-and-AR-device-shipments-to-hit-99m-by-2021

[3] (Jan. 2017). *The Reality of VR/AR Growth*. [Online]. Available: https://techcrunch.com/2017/01/11/the-reality-of-vrar-growth/

[4] F. Hu, Y. Deng, W. Saad, M. Bennis, and A. H. Aghvami, "Cellular-connected wireless virtual reality: Requirements, challenges, and solutions," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 105–111, May 2020.

[5] Z. Tan, Y. Li, Q. Li, Z. Zhang, Z. Li, and S. Lu, "Supporting mobile VR in LTE networks: How close are we?" *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 1, pp. 1–31, Mar. 2018.

[6] E. Cuervo, K. Chintalapudi, and M. Kotaru, "Creating the perfect illusion: What will it take to create life-like virtual reality headsets?" in *Proc. 19th Int. Workshop Mobile Comput. Syst. Appl.*, Feb. 2018, pp. 7–12.

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

[8] M. Chen, W. Saad, and C. Yin, "Virtual reality over wireless networks: Quality-of-service model and learning-based resource management," *IEEE Trans. Commun.*, vol. 66, no. 11, pp. 5621–5635, Nov. 2018.

[9] M. Chen, W. Saad, C. Yin, and M. Debbah, "Data correlation-aware resource management in wireless virtual reality (VR): An echo state transfer learning approach," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4267–4280, Jun. 2019.

[10] Y. Sun, Z. Chen, M. Tao, and H. Liu, "Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff," *IEEE Trans. Commun.*, vol. 67, no. 11, pp. 7573–7586, Nov. 2019.

[11] T. Dang and M. Peng, "Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 7, pp. 1594–1607, Jul. 2019.

[12] X. Hou, S. Dey, J. Zhang, and M. Budagavi, "Predictive view generation to enable mobile 360-degree and VR experiences," in *Proc. AR/VR Netw.*, Aug. 2018, pp. 20–26.

[13] C.-L. Fan, J. Lee, W.-C. Lo, C.-Y. Huang, K.-T. Chen, and C.-H. Hsu, "Fixation prediction for 360° video streaming in head-mounted virtual reality," in *Proc. NOSSDAV*, Jun. 2017, pp. 67–72.

[14] C. Guo, Y. Cui, and Z. Liu, "Optimal multicast of tiled 360 VR video in OFDMA systems," *IEEE Commun. Lett.*, vol. 22, no. 12, pp. 2563–2566, Dec. 2018.

[15] C. Guo, Y. Cui, and Z. Liu, "Optimal multicast of tiled 360 VR video," *IEEE Wireless Commun. Lett.*, vol. 8, no. 1, pp. 145–148, Feb. 2019.

[16] K. Long, Y. Cui, C. Ye, and Z. Liu, "Optimal transmission of multi-quality tiled 360 VR video by exploiting multicast opportunities," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[17] 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*, early access, Oct. 12, 2020, doi: 10.1109/TMM.2020.3029880.

[18] (Oct. 2015). *Under the Hood: Building 360 Video*. [Online]. Available: https://engineering.fb.com/video-engineering/under-the-hood-building-360-video/

[19] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart. 2017.

[20] N. Jiang, Y. Deng, and A. Nallanathan, "Traffic prediction and random access control optimization: Learning and non-learning based approaches," Feb. 2020, *arXiv:2002.07759*. [Online]. Available: http://arxiv.org/abs/2002.07759

[21] N. Jiang, Y. Deng, A. Nallanathan, and J. Yuan, "A decoupled learning strategy for massive access optimization in cellular IoT networks," May 2020, *arXiv:2005.01092*. [Online]. Available: http://arxiv.org/abs/2005.01092

[22] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3039–3071, 4th Quart. 2019.

[23] X. Chen, J. Wu, Y. Cai, H. Zhang, and T. Chen, "Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 4, pp. 627–640, Apr. 2015.

[24] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 5, pp. 1046–1061, May 2017.

[25] Z. Chen, T. Lin, and C. Wu, "Decentralized learning-based relay assignment for cooperative communications," *IEEE Trans. Veh. Technol.*, vol. 65, no. 2, pp. 813–826, Feb. 2016.

[26] J. Song, M. Sheng, T. Q. S. Quek, C. Xu, and X. Wang, "Learning-based content caching and sharing for wireless networks," *IEEE Trans. Commun.*, vol. 65, no. 10, pp. 4309–4324, Oct. 2017.

[27] H. Lee, S. H. Lee, and T. Q. S. Quek, "Deep learning for distributed optimization: Applications to wireless resource management," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2251–2266, Oct. 2019.

[28] G. Naddafzadeh-Shirazi, P.-Y. Kong, and C.-K. Tham, "Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 59, no. 8, pp. 4157–4162, Oct. 2010.

[29] K. W. Sung *et al.*, "PriMO-5G: Making firefighting smarter with immersive videos through 5G," in *Proc. IEEE 2nd 5GWF*, Sep. 2019, pp. 280–285.

[30] Y. Bao, T. Zhang, A. Pande, H. Wu, and X. Liu, "Motion-prediction-based multicast for 360-degree video transmissions," in *Proc. 14th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2017, pp. 1–9.

[31] *5G; 3GPP Virtual Reality Profiles for Streaming Applications*, document ETSI TS 126 118, Apr. 2019.

[32] V. Clay, P. König, and S. U. König, "Eye tracking in virtual reality," *J. Eye Movement Res.*, vol. 12, no. 1, pp. 284–295, Apr. 2019.

[33] R. Engbert, K. Mergenthaler, M. Breakspear, and A. Pikovsky, "An integrated model of fixational eye movements and microsaccades," *Proc. Nat. Acad. Sci. USA*, vol. 39, no. 108, pp. E765–E770, Aug. 2011.

[34] J. A. Roberts, G. Wallis, and M. Breakspear, "Fixational eye movements during viewing of dynamic natural scenes," *Frontiers Psychol.*, vol. 4, no. 797, Oct. 2013.

[35] H. C. Berg, *Random Walks in Biology*. Princeton, NJ, USA: Princeton Univ. Press, 1993.

[36] S. Kadloor, R. S. Adve, and A. W. Eckford, "Molecular communication using Brownian motion with drift," *IEEE Trans. Nanobiosci.*, vol. 11, no. 2, pp. 89–99, Jun. 2012.

[37] *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Extended Reality (XR) in 5G*, document 3GPP TR 26.928, Feb. 2020.

[38] S. Wedel, M. Koppetz, J. Skowronek, and A. Raake, "ViProVoQ: Towards a vocabulary for video quality assessment in the context of creative video production," in *Proc. 27th ACM Int. Conf. Multimedia*, Oct. 2019, pp. 2387–2395.

[39] V. P. M. Kumar and S. Mahapatra, "Quality of experience driven rate adaptation for adaptive HTTP streaming," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 602–620, Jun. 2018.

[40] F. Kuipers, R. Kooij, D. De Vleeschauwer, and K. Brunnström, *Techniques for Measuring Quality of Experience*. Berlin, Germany: Springer, Jun. 2010.

[41] L. U. Choi, M. T. Ivrlac, E. Steinbach, and J. A. Nossek, "Sequence-level models for distortion-rate behaviour of compressed video," in *Proc. IEEE Int. Conf. Image Process.*, vol. 2, Sep. 2005, pp. 2–486.

[42] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.

[43] H. Yi Ong, K. Chavez, and A. Hong, "Distributed deep Q-learning," Oct. 2015, *arXiv:1508.04186*. [Online]. Available: http://arxiv.org/abs/1508.04186

[44] A. Galindo-Serrano and L. Giupponi, "Distributed Q-learning for interference control in OFDMA-based femtocell networks," in *Proc. IEEE 71st Veh. Technol. Conf.*, May 2010, pp. 1–5.

[45] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[46] Y. Wei, F. R. Yu, M. Song, and Z. Han, "User scheduling and resource allocation in HetNets with hybrid energy supply: An actor-critic reinforcement learning approach," *IEEE Trans. Wireless Commun.*, vol. 17, no. 1, pp. 680–692, Jan. 2018.

[47] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar, "Fully decentralized multi-agent reinforcement learning with networked agents," in *Proc. Int. Conf. Mach. Learn.*, Jul. 2018, pp. 5872–5881.

[48] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.

Yansha Deng (Member, IEEE) received the Ph.D. degree in electrical engineering from the Queen Mary University of London, U.K., in 2015. From 2015 to 2017, she was a Post-Doctoral Research Fellow with Kings College London, U.K. She is currently a Lecturer (Assistant Professor) with the Department of Engineering, King's College London. Her research interests include molecular communication and machine learning for 5G/B5G wireless networks. She was a recipient of the best paper awards from ICC 2016 and GLOBECOM 2017, as the first author. She also received the Exemplary Reviewer Award of the IEEE TRANSACTIONS ON COMMUNICATIONS in 2016 and 2017, and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS in 2018. She is also an Associate Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS and IEEE TRANSACTIONS ON MOLECULAR, BIOLOGICAL AND MULTI-SCALE COMMUNICATIONS, and the Senior Editor of the IEEE COMMUNICATION LETTERS. She has also served as a TPC Member for many IEEE conferences, such as IEEE GLOBECOM and ICC.

Xiaonan Liu (Student Member, IEEE) is currently pursuing the Ph.D. degree with the Department of Engineering, Kings College London, U.K. His research interests include wireless virtual reality, 5G, machine learning, wireless communications, and Terahertz (THz) networks.