Learning-based Interactive Wireless Virtual Reality (VR) Network

Liu, Xiaonan

Awarding institution:
King's College London

Learning-based Interactive Wireless Virtual Reality (VR) Network

Xiaonan Liu

Supervisor: Dr. Yansha Deng
Dr. Reza Nakhai

The Department of Engineering
King's College London

This dissertation is submitted for the degree of
Doctor of Philosophy

June 2022

I would like to dedicate this thesis to my loving parents,
who always believe in me and support me to do what I want to.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables, and equations and has fewer than 150 figures.

Xiaonan Liu
June 2022

Acknowledgements

My PhD study was far from easy or smooth, and I can't imagine I would have gone through it on my own. This is why I would like to express my deepest gratitude to my supervisor, Dr. Yansha Deng, for her continuous guidance, suggestions, and mentoring. When I sat alone in my room throughout lockdowns, lost in front of my computer, Yansha was always there to help me keep on track. The depth of her knowledge, work ethic, attention to detail, and dedication are far beyond what I could have hoped from a supervisor.

I would like to thank my collaborators, Prof. Chong Han, at Shanghai Jiao Tong University, and Prof. Marco Di Renzo, at Paris-Saclay University, for their constructive comments and suggestions on the research. In addition, I wish to take this chance to thank Prof. Nan Zhao at Dalian University of Technology who is the first supervisor in my research career, for all your generous help and advice that guides me from an innocent student to a mature researcher.

During my PhD study, my life at King's College London is quite enjoyable thanks to all my group members and colleagues, including Dadi Bi, Fenghe Hu, Hui Zhou, Yan Liu, Nan Jiang, Shengzhi Yang, Xiaokang Zhou, and Liyana. Our group is a big and warm family. We worked hard and played hard together. We shared and helped each other whenever in need. I will miss all the fun and happy moments with all of you.

Most importantly, I would like to thank my parents, Jian Liu and Qianwen Xiao, and my grandparents, Shijun Liu, Guixiang Li, Zhen Xiao, and Huaiying Liang, for their unconditional support, meticulous care, and unlimited love throughout my life. None of this would have been possible without your support.

Last, a special thank should be dedicated to my fiancee, Dr. Dongzhu Liu, a lovely girl with beauty both inside and outside. What a fortune that I can meet you, such a perfect girl, among the crowd. Now, I know you are the destination that I have been always seeking. Thank you, Dongzhu, for all your support and love, and wish us a forever happy life.

Abstract

Wireless-connected Virtual Reality (VR) provides an 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 are challenging due to its requirements in high Quality of Experience (QoE), low VR interaction latency under latency threshold, and privacy. To address these issues, in this thesis, we mainly focus on optimizing the uplink, rendering, downlink, and privacy of wireless VR in 5G and terahertz (THz) networks.

If the viewpoint of the VR user can be predicted, the corresponding VR video frames can be rendered and delivered in advance, which can reduce the VR interaction latency. Thus, in the first chapter, we use offline and online learning algorithms to predict the viewpoint of the VR user using a real VR dataset. For the offline learning algorithm, the trained learning model is directly used to predict the viewpoint of VR users in continuous time slots. While for the online learning algorithm, based on the VR user's actual viewpoint delivered through uplink transmission, we compare it with the predicted viewpoint and update the parameters and input viewpoints of the online learning algorithm to further improve the prediction accuracy. To guarantee the reliability of the uplink transmission, we integrate the Proactive retransmission scheme of 5G into our proposed online learning algorithm. Simulation results show that our proposed online learning algorithm with the proactive retransmission scheme for wireless VR networks only exhibits about 5% prediction error.

Rendering real-time VR videos with high quality demands a computing unit with high processing ability. Thus, in the second chapter, we propose a mobile edge computing (MEC)-enabled wireless VR network in 5G, 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 the VR device to the 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.

In an indoor scenario, a high data rate over short transmission distances may be achieved via the abundant bandwidth in the THz band. However, THz waves experience severe signal attenuation, which may be compensated by the reconfigurable intelligent surface (RIS) technology with programmable reflecting elements. Meanwhile, the low VR interaction latency can be achieved with the MEC network architecture due to its high computation capabilities. Motivated by these considerations, in the third chapter, we propose an MEC-enabled and RIS-assisted THz VR network in an indoor scenario, by taking into account the uplink viewpoint prediction and position transmission, the MEC rendering, and the downlink transmission. We propose two methods, which are referred to as centralized online gated recurrent unit (GRU) and distributed federated averaging (FedAvg), to predict the viewpoints of the VR users. In the uplink, an algorithm that integrates online long-short term memory (LSTM) and convolutional neural networks (CNN) is deployed to predict the locations and the line-of-sight (LoS) or non-line-of-sight (NLoS) statuses of VR users over time. In the downlink, we develop a constrained DRL (C-DRL) algorithm to select the optimal phase shifts of the RIS under latency constraints. Simulation results show that our proposed learning architecture achieves near-optimal QoE as that of the genie-aided benchmark algorithm, and about two times improvement in QoE compared to the random phase shift selection scheme.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

5G	Fifth Generation of Mobile Communication Networks
AC	Actor Critic
ACK	Acknowledge
AI	Artificial Intelligence
ANN	Artificial Neural Network
AoI	Age of Information
BPTT	BackPropagation Through Time
CDF	Cumulative Distribution Function
CDRL	Constrained Deep Reinforcement Learning
CNN	Convolutional Neural Network
DAC	Distributed Actor Critic
DDQN	Distributed Deep Q Network
DL	Deep Learning
DNN	Deep Neural Network
DQN	Deep Q Network
DRL	Deep Reinforcement Learning
E2E	End-to-End
eMBB	Extreme Mobile Broadband

ERP	Equirectangular Projection
ESN	Echo State Network
FCFS	First Come First Serve
FL	Federated Learning
FoV	Field of View
Gbps	Gigabits-per-second
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
LCFS	Last Come First Serve
LoS	Line-of-Sight
LR	Linear Regression
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MEC	Mobile Edge Computing
MIMO	Multiple-Input Multiple-Output
MISO	Multi-Input Single-Output
ML	Machine Learning
MMTC	Massive Machine Type Communication
mmWave	Millimeter Wave
MRC	Maximum Ration Combining
MSE	Mean Square Error
NACK	Non-Acknowledge
NLoS	Non-Line-of-Sight
NN	Neural Network

POMDP	PartiallyObservable Markov Decision Process
PSNR	Peak Signal-to-Noise Ratio
QoE	Quality of Experience
QoS	Quality of Service
RIS	Reconfigurable Intelligent Surface
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SBS	Small-cell Base Station
SGD	Stochastic Gradient Descent
SIMO	Single-Input-Multiple-Output
Tbps	Terabits-per-second
TD	Time-Difference
THz	Terahertz
TTI	Transmission Time Interval
URLLC	Ultra-Reliable Low Latency Communication
V2V	Vehicle to Vehicle
VRMM	Virtual Reality Mobility Model
VR	Virtual Reality

Chapter 1

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 [56]. 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 wired connected VR devices is the limited mobility of VR users. To overcome this disadvantage, wireless connected VR devices can be a potential solution in providing ubiquitous user experiences from anywhere at anytime, and also can unleash plenty of novel VR applications. Nevertheless, there are some unique challenges in wireless VR systems that do not exist in wired VR systems and traditional wireless video transmission systems as identified in [69]. 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 [133, 51, 25]

Furthermore, in recent years, novel communication technology such as 5G, Millimeter wave (mmWave), and terahertz (THz) can provide high transmission rate, low interaction latency, and high transmission reliability. For example, the transmission rate of 5G, mmWave, and THz can achieve 20 Gbps, 25Gbps, and 100 Gbps, respectively [17, 33, 11]. Thus, the requirement of the low latency of wireless VR can be guaranteed. Meanwhile, some techniques in these novel communication technologies, such as proactive retransmission scheme in 5G and reconfigurable intelligence surface (RIS) in mmWave and THz, can guarantee the high reliability of VR video streaming. In addition, cloud and edge artificial intelligence (AI) can help the edge server predict the VR user preferences and render the required VR video frames in advance, which decrease the VR interaction latency. Also, the edge server is equipped with high computation capability, which further decreases the rendering latency.

In this chapter, we mainly focus on the related works of wireless VR and the background knowledge used to optimize the wireless VR network, including 5G, terahertz, mobile edge computing, and machine learning.

1.1 Related Works

In this section, related works on 5G VR networks, MEC-enabled wireless VR networks, viewpoint prediction-assisted VR networks, THz VR networks, and RIS-assisted THz networks are briefly introduced in the following five subsections.

5G VR Networks

In [42], 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 [42], the authors in [43] optimized the resource block allocation using ESN to maximize the success transmission probability accounting for the VR user data correlation. In [68] and [57], a Recurrent Neural Network (RNN) based on long short-term memory (LSTM) was applied for viewpoint prediction. In [62] and [95], the authors studied optimal multicast of tiled 360 VR video from one base station (BS) to multiple VR users in OFDMA and TDMA systems, and focused on downlink transmission via optimizing transmission power.

MEC-enabled wireless VR networks

In [131] and [52], a joint caching and computing optimization problem was formulated to minimize the average transmission rate and maximize the average tolerant delay, respectively. In [96], through effectively exploiting the characteristics of multi-quality tiled 360 VR videos and computation resources, the optimal wireless streaming of a multi-quality tiled 360 VR video to multiple users in wireless networks was investigated to decrease the energy consumption. In [92], a decoupled learning strategy was developed to optimize the real-time VR video streaming in wireless networks, by taking into account the FoV prediction and rendering MEC association. Through the design of centralized/distributed deep reinforcement learning algorithms to select proper MECs to render and transmit predicted VR video frames, the QoE was enhanced and the VR interaction latency was reduced. However, the authors of [131, 52, 96] mainly used convex optimization to optimize the wireless VR network, which can not guarantee the long-term QoE of the VR users. The authors of [92] only considered the MEC association problem without using real VR datasets.

Viewpoint prediction-assisted VR networks

The viewpoint prediction problem in wireless VR system has been studied in [85, 86, 92, 23, 22, 21]. The authors in [85] considered viewpoint prediction via constant angular velocity and constant acceleration after every 20 ms. In [86], the authors proposed a double exponential smoothing method to predict users' head position and rotation after every 50 ms. In [92], the authors considered Brownian Motion to simulate the eye movement and used RNN to predict viewpoint preferences in continuous time slots. However, the methods in [85] and [92] were not data-driven, and prediction results obtained in [85] and [86] would be unaligned with the viewpoint preference of VR users, which may not fit for real-time VR video transmission. In [22], [23] and [21], the authors only used offline Linear Regression (LR) [65] and Neural Network (NN) [105] to predict the viewpoints of VR users in continuous time slots with real VR dataset, and assumed that all viewpoint requests are available at the SBS, which is not possible without 100% reliable uplink transmission. Based on such predictions in [22], [23] and [21], the authors minimized the multicast bandwidth consumption by sending the predicted part of the spherical VR videos.

THz VR networks

In [38, 36, 54, 37], the reliability, transmission rate, viewpoint rendering, and energy consumption of a THz VR system were investigated. Specifically, in [38], the reliability of VR services in the THz band was studied, and the theoretical analysis of the end-to-end (E2E) delay was performed. In [36], a risk-based framework was proposed to optimize the rate and reliability of THz-band for VR applications. In [54], through jointly optimizing the viewport rendering offloading and downlink transmit power control, the long-term energy consumption of a THz wireless access-based MEC system for high quality immersive VR video services was minimized. In [37], the age of information (AoI) of AR services in a THz cellular network with RISs was studied, and the cumulative distribution function (CDF) of the AoI was derived for two different scheduling policies, which are the last come first served (LCFS) queues and the first come first served (FCFS) queues. However, in [38], [36], and [37], the authors mainly focused on the theoretical analysis of the reliability and E2E delay of the THz VR. In [54], the authors mainly optimized the long-term energy consumption via deep reinforcement learning, rather than the QoE and VR interaction latency, without the consideration of RIS.

RIS-assisted THz networks

In [99, 98, 75, 139, 79, 125, 39, 73, 74], the sum rate maximization problem in RIS-assisted THz network was investigated. Specifically, in [99], based on channel estimation, a deep neural network (DNN) was proposed to select the optimal phase shift configurations to maximize the sum rate in the RIS-assisted THz multiple-input multiple-output (MIMO) system. In [98], through optimizing the beamforming vector of the transmitter and reflection coefficients matrix of the RIS, the coverage probability in indoor THz communication scenarios was improved, and a low complexity phase shift search scheme was used to achieve near-optimal coverage performance. In [75], energy-efficient designs for both the transmit power allocation and the phase shifts of the RIS subject to downlink multi-user communication were developed. In [139], the downlink of an RIS-empowered multiple-input single-output (MISO) communication system was considered, where the alternating least squares and vector approximate message passing methods were used to estimate channels between the BS and the RIS, as well as the channels between the RIS and users. In [79], RIS-assisted secure wireless communications were investigated, and the successive convex approximation-based algorithm was used to solve the transmit covariance matrix optimization problem, which maximized the secrecy rate. In [125], the network architecture and spectrum access of AI-enabled Internet of Things in 5G and 6G networks were proposed. In [39], a comprehensive roadmap outlining the seven defining features of THz wireless systems that guarantee a successful deployment in future wireless generations was proposed. In [73] and [74], a novel hybrid beamforming scheme for multi-hop RIS-assisted communication networks was proposed to improve the coverage range of THz networks. Nevertheless, in [99, 98, 75, 139, 79, 125, 39, 73, 74], the authors mainly optimized the phase shift and beamforming vector to maximize the sum rate of RIS-assisted THz networks using a DNN or non-learning based approaches.

1.2 Background

1.2.1 Wireless VR System

In wireless VR systems, wireless VR video streaming mainly includes three parts. First, VR users transmit the tracking information, such as head and eye movements to the cloud or edge server. Then, the cloud or edge server renders and encodes the required VR video frames, and transmits them to the VR users through downlink transmission. Finally, the VR users decode the received VR video frames and adjust them to the VR devices. If the cloud or edge server can predict the VR user preferences, the corresponding VR video frames can be rendered and delivered in advance, which can further decrease the VR interaction latency. The process of wireless VR is shown in Fig. 1.1.

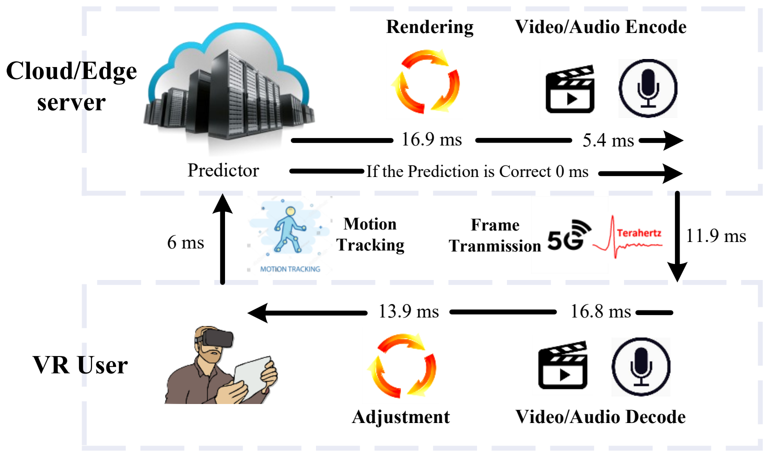


Fig. 1.1 Decomposition of end-to-end VR interaction latency in 5G and THz with cloud/edge server.

1.2.2 Overview of 5G Technology

In recent three decades, concerning the transition from 1G to 4G, there is rapid growth in the field of wireless network [27, 103]. The main reasons behind this are the requirements of high bandwidth and low latency. Fortunately, 5G is able to provide high data rate, high quality of service (QoS), low latency, high coverage, high reliability, and economically affordable services. The services provided by the 5G are classified into three categories: (1) Extreme mobile broadband (eMBB), it provides high-speed network connectivity, larger bandwidth, and moderate latency. eMBB is designed for UltraHD streaming videos, VR, and AR media [13]. (2) Massive machine type communication (MMTC), it provides long-range and broadband machine-type communication at a very cost-effective price with less power consumption. Also, it brings a high data rate service, low power, and extended coverage via less device complexity through mobile carriers for IoT applications [120]. (3) Ultra-reliable low latency communication (URLLC) offers ultra-high reliability, ultra-low latency, and high QoS, which are not possible for traditional mobile network architecture from 1G to 4G. URLLC is designed for real-time interaction such as remote surgery, vehicle to vehicle (V2V) communication, industry 4.0, and smart grids [123].

Although 5G wireless networks provide ubiquitous coverage with high speed and low latency, with billions of mobile devices connected to the network, 5G technology still faces several significant challenges. First, with the increasing number of mobile devices access to the network, they are easily vulnerable to be attacked from the Internet. Thus, privacy and security are the most critical factors to be considered when promoting the development of 5G applications, such as VR and AR [82]. Second, when mobile applications in 5G become popular, millions of mobile devices will operate and transmit data continuously every day. As a consequence, a large amount of energy will be consumed every day. Thus, energy-efficient communication solutions are a real challenge [34, 143].

1.2.3 Overview of Terahertz Communication

5G provides a high-speed internet facility from anywhere at anytime for everyone. However, mobile data traffic has increased exponentially in the present decade, and the available bandwidths are inadequate to meet the data rate demands of users with diverse bandwidth-hungry applications. For example, future wireless local area networks (WLAN) and wireless personal area networks (WPAN) require the data rate at least a couple of ten multi-gigabits-per-second (Gbps) [126]. Meanwhile, the minimum data rate of some wireless applications, such as VR and AR, is expected to be 10 Gbps. In addition, the data rate of uncompressed ultra-high-definition videos and 3D videos will reach 24 Gbps and 100 Gbps, respectively [76]. The present deployed 5G communications will find it difficult to satisfy the data rates and ultra-low latency requirements of these applications. To satisfy the requirements of the high data rate mentioned above, spectrum utilization will move to the terahertz (THz) bands (0.1 - 10 THz), where multi-gigahertz (GHz) contiguous bandwidth is available to enable Gbps and/or terabits-per-second (Tbps) rates [15].

Despite promising available bands in the THz spectrum, wireless signals in the THz frequency range experience severe propagation attenuations and water-molecular absorption losses because of the high frequency, which limits the propagation distance [16]. Furthermore, obstacles in the wireless communication environment can easily block THz signals due to their poor diffraction and scattering capabilities [58, 108]. To address these issues, smart wireless networks for the THz band have attracted significant attention recently, which rely on the concept of reconfigurable intelligent surfaces (RISs) [24, 112–114, 158]. In particular, RIS consists of flat electromagnetic (EM) material surfaces with an array of dispersive elements. Each element induces an amplitude and/or phase shift to THz signals to enhance the received signal power and create a desirable multipath effect without the need for complex coding and decoding schemes or additional frequency radio (FR) operations [146, 147].

1.2.4 Mobile Edge Computing

In the last decade, cloud computing has emerged as a new paradigm of computing. The resources in the clouds are leveraged to provide elastic computing capability and memory to support resource-constrained mobile devices [156]. However, cloud computing has some challenges. First, cloud computing requires transmissions from end-users to the cloud server with distances ranging from tens of kilometers, which can result in high propagation delay. Second, cloud computing requires the delivered information to pass through a radio access network and a backhaul network, where traffic control, routing, and other resource management techniques can lead to extra delay. Finally, although cloud computing has a huge computation capability, it has to be shared by a huge number of devices. If the computation resource cannot be allocated properly, it can lead to high computation latency [136].

To address the issues of cloud computing mentioned above, in recent years, a new computing trend is happening with cloud computing moving towards the network edges, so-called mobile edge computing (MEC) [2]. The main feature of MEC is to push mobile computing, network control, and storage to the network edges, such as base stations (BSs) and access points (APs), which enables computation-intensive and latency-critical applications to be executed at the resource-constrained mobile devices [12]. Compare to cloud computing, the MEC achieves lower latency. First, the transmission distances are typically tens of meters, and no longer than one kilometer for general cases, which decreases transmission latency. Second, the traffic control, routing, and network-management operations of the MEC are much easier than that of cloud computing, which decreases the extra delay. Finally, the computation capability of a modern BS with edge computing is powerful enough for running sophisticated computing programs, which may achieve lower computation latency [118]. To sum up, with short transmission distances and simple protocols, MEC has the potential of realizing tactile-level latency for latency-critical 5G applications, such as VR and AR [124].

1.2.5 Machine Learning

Machine learning (ML) was born from pattern recognition and it is based on the assumption that machines are endowed with artificial intelligence (AI), so that they can learn from previous computations or adapt to their environment through experience [19, 130]. Because of the rapid growth of data, especially in wireless networks, the need for intelligent data analysis and the deployment of ML algorithms have become necessary and ubiquitous. Through using ML algorithms to build models, system operators can predict dynamic systems or make intelligent decisions without human intervention [18].

Deep Neural Network

Among ML algorithms, a deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers [50]. DNN can learn any nonlinear function and is capable of learning weights that map any input to the output.

One of the main reasons behind it is the activation functions, which introduce nonlinear properties to the network and help the network learn any complex relationship between the input and output [14].

When solving an image recognition problem with DNN, the first step is to convert the two-dimensional images into one-dimensional vectors. However, it has two disadvantages. First, with the increasing size of the image, the number of neurons and weights of the DNN increases drastically, which can occupy a large amount of memory and increase training latency. Second, the DNN cannot extract the spatial features of an image, where spatial features are the arrangement of pixels in an image. Thus, the learning accuracy severely decreases if the DNN is used for image classification. In addition, the DNN cannot capture sequential information of the input data especially when the input data is time-correlated. To deal with these issues, two different learning architectures are proposed, namely, convolution neural network (CNN) and recurrent neural network (RNN).

Convolution Neural Network

A CNN is a type of DNN for processing data that has grid patterns, such as images, which is designed to automatically and adaptively learn spatial hierarchies of features [121]. The CNN typically consists of three types of layers, which are convolution layer, pooling layer, and fully connected layer. Convolution and pooling layers perform feature extraction, and the fully connected layer maps the extracted features into the final output, such as image recognition and classification. The convolution layer plays an important role in CNN, which is consisted of a stack of mathematical operations, such as convolution. In images, pixel values are stored in a two-dimensional grid, i.e., an array of numbers, and a small grid of parameters called kernel (an optimizable feature extractor) is applied at each image position, which makes the CNN highly efficient for image processing. As one layer feeds its output into the next layer, extracted features can hierarchically and progressively become more complex.

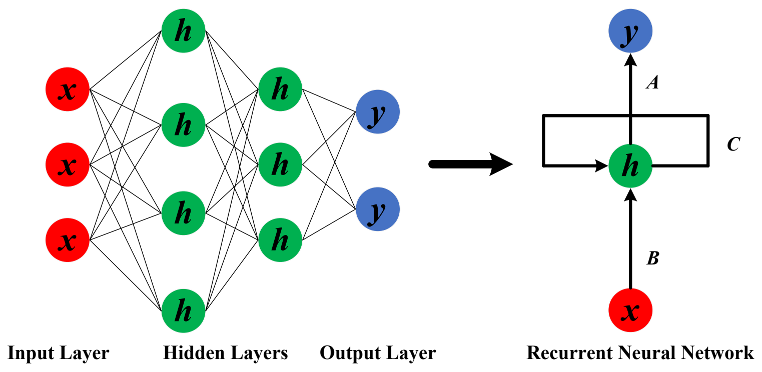


Fig. 1.2 Recurrent neural network architecture.

Recurrent Neural Network

RNN is a class of neural networks that allow previous outputs to be used as inputs while having hidden states, which is adapted to work for time series data or data that involve sequences [91]. Unlike the DNN, the neurons in different layers of the DNN are compressed to form a single layer of the RNN, as shown in Fig. 1.2. In Fig. 1.2, $\mathbf{x}$ is the input layer, $\mathbf{h}$ is the hidden layer, $\mathbf{y}$ is the output layer, and A, B, and C are the network parameters used to improve the output of the model. At the tth time slot, the input is a combination of input at the tth and $(t-1)$th time slots. The output at each time slot is then feedback to the input to improve the output. However, RNN is hard to train because of two gradient problems, namely, vanishing gradient problem and exploding gradient problem. The gradients carry information used in the RNN, and when the gradient becomes too small, the weights updates become insignificant, so-called vanishing gradient, which makes the learning of long-term data sequences difficult. When the slope of the gradient tends to grow exponentially instead of decaying, so-called exploding gradient, there will be large updates to the weights of the network model during the training process. These gradient problems lead to a long training time, poor convergence performance, and low learning accuracy. To solve the gradient problems, the most popular and efficient way is deploying long short-term memory (LSTM) or gated recurrent unit (GRU).

Long Short-term Memory LSTM is capable of learning long-term dependencies and uses a gating mechanism to control the memorizing process [66]. Information in the LSTM can be stored, written, or read through opening or closing gates. There are three different gates in an LSTM cell, which are a forget gate, an input gate, and an output gate. The forget gate decides which information needs to be kept or ignored. The input gate decides which relevant information can be added to the long-term memory and it only works with the information from the current input and short-term memory from the previous steps. The output gate determines the value of the next hidden state, where the hidden state consists of information from previous inputs.

Gated Recurrent Unit GRU shares many properties of LSTM, and it still uses a gating mechanism to control the memorizing process. Fortunately, GRU is less complex than LSTM and faster to execute [47]. GRU has two gates, which are an update gate and a reset gate. The update gate is responsible for determining the amount of previous information that needs to pass along to the next state. The reset gate is used from the model to decide how much of the past information is needed to ignore, which decides whether the previous cell state is important or not.

To sum up, compared with LSTM, GRU has fewer training parameters, occupies less memory, and executes faster than LSTM. However, LSTM is more accurate on a larger dataset. One can choose LSTM if dealing with large sequences and accuracy is concerned, and GRU is used when you have less memory consumption and want faster results.

In contrast to the previously discussed learning models, such as CNN and RNN, that need to be trained with labeled or unlabeled data, deep reinforcement learning (DRL) is trained by the observations from the environment.

Deep Reinforcement Learning

Reinforcement learning (RL) is a learning model where an agent learns to make decisions through feedback rewards. The problem is usually modeled as a Markov decision process (MDP). At each time slot, an agent observes a state of the environment, selects an action, receives a reward, and transitions to the next state. The agent attempts to learn an optimal policy to maximize the long-term reward [70]. However, in many practical decision-making problems, the state and action of the environment are high-dimensional and cannot be solved by traditional RL algorithms. Thus, deep RL (DRL), where DNNs are integrated with RL, is proposed to deal with this issue [81], which is shown in Fig. 1.3.

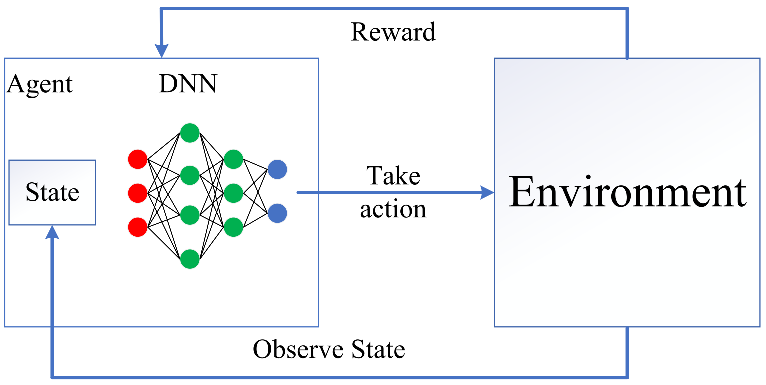


Fig. 1.3 Deep reinforcement learning architecture.

Online Learning

Traditional ML algorithms usually work in an offline learning mode. Offline learning can be divided into two phases, namely, the training phase and the testing phase. In the training phase, the learning model is first trained by the training datasets until an optimal set of hyperparameters for the learning model is obtained. In the testing phase, the trained learning model is deployed for prediction or decision-making without performing any further model updates. The offline training method suffers from expensive re-training costs when dealing with new training data/environment so that they are poorly generalized to the unseen data in real-world applications, especially when the data grows and the environment evolves rapidly.

Unlike offline learning, online learning includes an important family of ML algorithms that are designed to learn models incrementally from data in a sequential manner. Online learning overcomes the drawbacks of offline learning in that the learning models can be updated constantly and efficiently when new training data arrives. Thus, online learning models are much more efficient and scalable for large-scale ML tasks in real-world data applications, where datasets are not only large but also updated frequently [67].

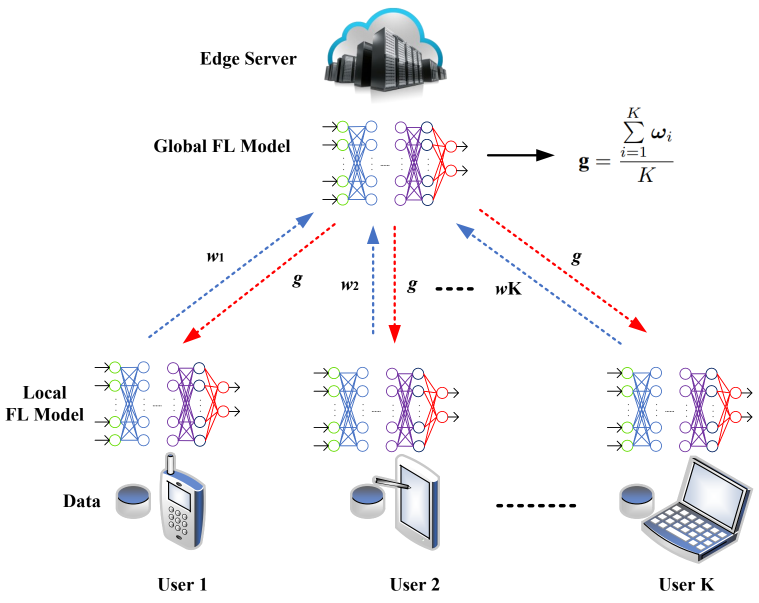


Fig. 1.4 Federated learning architecture.

Federated Learning

The inevitable rise of edge computing enables many ML algorithms to effectively analyze various types of data for resource management, interference management, autonomy, and decision making in computer or wireless networks. However, because of the hardware resource constraints, uncertain wireless environment, latency constraints, and privacy challenges, the edge devices cannot transmit their entire datasets to the edge servers for training their ML models centrally [157]. To guarantee the training data remaining trained on personal devices and facilitate collaborative ML of complex models among distributed IoT or mobile devices, a distributed ML model called federated learning (FL) was first introduced in [102]. As shown in Fig. 1.4, in the FL, 1) each device uses its dataset to train a local model; 2) they send their local models to the server for model aggregation; 3) the server transmits the updated global model to these devices. These steps are repeated in multiple iterations until convergence. Compared to the traditional central learning and the distributed learning that executes via exchanging raw data, the FL has following advantages:

- In the FL, only the model weights are transmitted between the server and devices rather than the datasets. Thus, less information is required to be delivered to the server, which significantly decreases the transmission overhead and latency, and relieves the burden on networks.
- The raw data of each device does not need to be exchanged or sent to the cloud or edge, which can guarantee the user privacy and decrease the probability of eavesdropping. Thus, more devices are willing to take part in the model training to achieve a high learning performance.

Given the aforementioned advantages, the FL has obtained successes in many applications, including training predictive models for human trajectory or behavior via mobile devices [128], automatically learning users' behavior patterns via smart home IoT [151], and diagnosis in health AI for cooperation among multiple hospitals and government agencies [31, 110, 135].

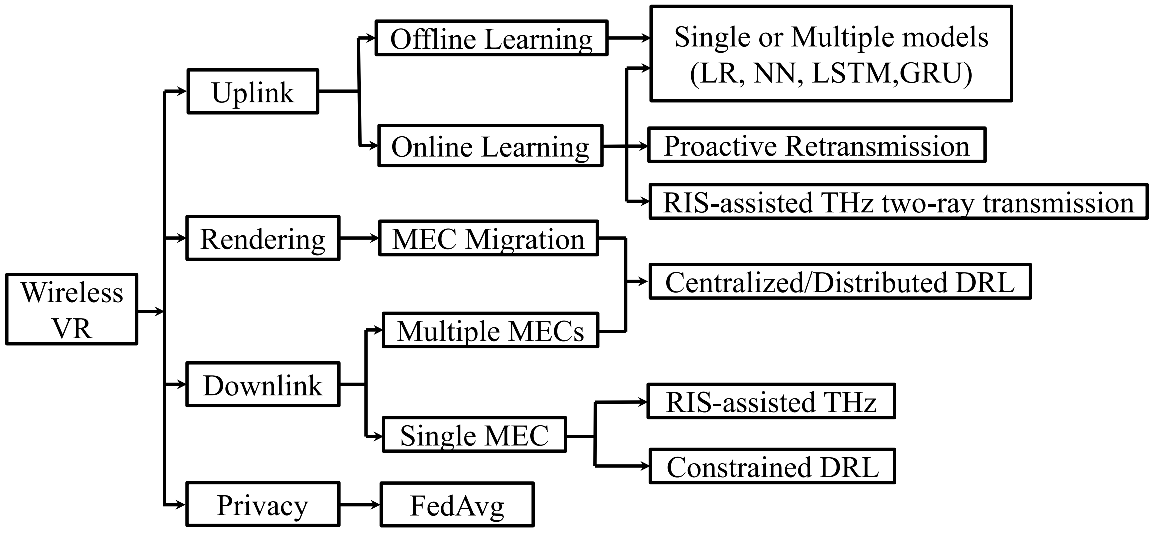


Fig. 1.5 Diagram of main contribution.

1.3 Contributions

In this dissertation, we integrate the machine learning into the wireless VR and focus on optimizing four parts, which are uplink transmission, rendering, downlink transmission, and privacy. As shown in Fig. 1.5, predicting viewpoints of VR users by offline or online learning can decrease uplink latency. Based on the predicted viewpoints, their corresponding VR video frames can be rendered by MEC and transmitted to VR users in advance. Thus, VR users can directly enjoy VR video frames without waiting. In addition, to guarantee the privacy of required VR video frames, federated averaging is deployed to predict VR video frames at VR devices, so that VR users do not need to exchange datasets with the MEC. The main contributions are summarized as follows:

- To decrease uplink transmission latency, we consider offline and online learning for viewpoint prediction in Chapter 2. Due to the fact that there are 16 VR videos in the real VR dataset, we consider two learning schemes, which are 16 learning models for these 16 VR videos, and only one learning model for all VR videos. For online learning, we consider it in 5G and THz networks. To guarantee the reliability of uplink transmission in 5G networks, we integrate a proactive retransmission scheme into online learning. While for the THz VR networks, we use RIS-assisted THz transmission in Chapter 4.

- To decrease rendering latency, we consider MEC to render the VR video frames in Chapter 3 and 4. For the VR users who need the same VR video frame while associated with different MECs because of their locations, the MEC with the highest computation capacity is used to render the VR video frames and migrate to the other MECs.

- To decrease downlink transmission latency, we consider multiple MECs in 5G networks in Chapter 3 and a single MEC in THz networks in Chapter 4. For multiple MECs in 5G networks, we consider centralized and distributed DRL to select proper MECs for VR video frame rendering and performing downlink transmission. For a single MEC in THz networks, we consider RIS-assisted THz to transmit VR video frames for VR users in the NLoS area and use constrained DRL to select proper phase shift elements of the RIS.

- To guarantee the privacy of VR users in viewpoint prediction, we consider federated learning in Chapter 4, where the VR datasets are stored in VR devices, the learning models directly predict the viewpoints of VR users over time, and then the VR users transmit the updated learning models to the MEC for model aggregation.

1.4 Outline of the Dissertation

The outline of this dissertation is organized as follows.

- The introduction is proposed in Chapter 1.
- Learning-based prediction and proactive uplink retransmission for wireless VR network is considered in Chapter 2.
- Learning-based prediction, rendering and association Optimization for MEC-enabled wireless VR network is proposed in Chapter 3, where the prediction methods of Chapter 2 are deployed in Chapter 3.
- Learning-based prediction, rendering and transmission for interactive VR in RIS-assisted terahertz networks is presented in Chapter 4, where the prediction methods of Chapter 2 and rendering methods of Chapter 3 are deployed in Chapter 4.
- The conclusions and future works are described in Chapter 5.

1.5 Publications

The manuscript includes the following works:

[1] **Xiaonan Liu**, Xinyu Li, and Yansha Deng. Learning-based Prediction and Uplink Retransmission for Wireless Virtual Reality (VR) Network. **IEEE Transactions on Vehicular Technology (TVT)**, 2021, 70(10): 10723 - 10734.

[2] **Xiaonan Liu**, and Yansha Deng. Learning-based Prediction, Rendering and Association Optimization for MEC-enabled Wireless Virtual (VR) Network. **IEEE Transactions on Wireless Communications (TWC)**, 2021, 20(10): 6356 - 6370.

[3] **Xiaonan Liu**, Yansha Deng, Chong Han, and Marco Di Renzo. Learning-based Prediction, Rendering and Transmission for Interactive Virtual Reality in RIS-Assisted Terahertz Networks. **IEEE Journal on Selected Areas in Communications (JSAC)**, 2022, 40(2): 710 - 724.

[4] **Xiaonan Liu**, and Yansha Deng. Viewpoint Prediction and Uplink Retransmission for Wireless Virtual Reality (VR) Network. **IEEE ICC 2021**.

[5] **Xiaonan Liu**, and Yansha Deng. A Decoupled Learning Strategy for MEC-enabled Wireless Virtual Reality (VR) Network. **IEEE ICC Workshop 2021**.

[6] **Xiaonan Liu**, Yansha Deng, Chong Han, and Marco Di Renzo. Ensemble Learning Strategy for RIS-Assisted Terahertz Virtual Reality Networks. **IEEE Globecom 2021**.

Other works published or submitted during the redaction of this manuscript, which are not included:

[7] Liyana Burhanuddin, **Xiaonan Liu**, Yansha Deng, Ursula Challita, and Andras Zahemszky. QoE Optimization for Live Video Streaming in UAV-to-UAV Communications

via Deep Reinforcement Learning. **IEEE Transactions on Vehicular Technology (TVT)**, 2022, 71(5): 5358 - 5370.

[8] Liyana Burhanuddin, **Xiaonan Liu**, Yansha Deng, Giovanni Geraci, Maged Elkashlan, and Arumugam Nallanathan. Interference Mitigation for Cellular-Connected UAV Networks via Deep Reinforcement Learning. **Submitted to IEEE Globecom 2022**

Work in progress, not yet submitted:

[9] **Xiaonan Liu**, and Yansha Deng. Marriage between the Federated Learning and the Wireless Communication: A Survey.

[10] Liyana Burhanuddin, **Xiaonan Liu**, and Yansha Deng. Inter-cell Interference Mitigation for Cellular-connected UAVs using Deep Reinforcement Learning.

Publications [1] and [4] are corresponding to Chapter 2. Publications [2] and [5] are corresponding to Chapter 3. Publications [3] and [6] are corresponding to Chapter 4.

Chapter 2

Learning-based Prediction and Proactive Uplink Retransmission for Wireless VR Network

2.1 Introduction

When the VR user enjoys the VR video frame, it mainly focuses on a certain direction at any given time slot. Based on the viewing direction, the corresponding portion of the image, defined by the Field of View (FoV) [142], needs to be rendered and delivered. The FoV determines the extent of the virtual environment that can be viewed. The center of the FoV that the VR user is watching is called Viewpoint [22]. If the viewpoint of the VR user is able to be well predicted, only its corresponding FoV is required to be rendered and delivered in advance, rather than rendering and transmitting the whole spherical video, which can further reduce the VR interaction latency [9].

2.1.1 Motivation

Because of the random nature of the head motion of VR users, viewpoint prediction based on delayed uplink viewpoint transmission may be prone to error, and only using trained linear regression (LR) and neural network (NN) cannot guarantee the highest prediction accuracy and capture the complex dynamics viewpoint preference over time, which may further degrade the quality of experience (QoE) [69] of VR users. To address this issue, an RNN based on the state-of-the-art Long Short-Term Memory (LSTM) [66] or Gated Recurrent Units (GRU) [47] architecture can be designed to predict the viewpoint of the VR user. LSTM and GRU could be successfully applied to sequence prediction and sequence labeling tasks due to their capability in dealing with the time-correlation tasks. It is important to know that they have the capability to learn the correlation of time-varying inputs, and they are well-suited to classify, process, and predict time-series events. Also, they can be used to deal with the exploding and vanishing gradient problem when training traditional RNNs [129]. However, the trained LR, NN, LSTM, and GRU learning models, namely, offline learning models, cannot adapt to the dynamic changing environment and has poor adaptability to viewpoint prediction of new VR users.

In contrast to the offline learning algorithm, the online learning model is updated with each FoV request received, and can automatically adjust the model itself according to the change of the received data. In [92], through transmitting the FoV request to the MEC-enabled SBS in the wireless VR network, the MEC was able to accurately predict the required FoV of the VR user over time, and render and deliver the FoV in advance, which could decrease the VR interaction latency and improve the QoE of the VR user. Therefore, the online learning algorithm has the potential to learn and update the best predictor for future FoV preference at each time slot and can be updated instantly once the FoV request of a new VR user is received [67, 109, 153].

To update the hyper-parameters in an online fashion, the VR users need to transmit its actual viewpoint to the SBS through uplink transmission, and the SBS will compare the actual viewpoint with the predicted viewpoint to reduce the loss between them. In the wireless VR network, due to the unstable wireless channels and the interference from other VR users, it is possible that the uplink transmission between the VR user and the SBS fails. Nevertheless, in the aforementioned wireless VR systems [85, 42, 43, 131, 86, 92, 23, 22, 21], the authors did not consider the potential uplink transmission failure caused by the wireless fluctuation to the online training, and the potential uplink transmission enhancement for better online training. To deal with this issue, the Proactive retransmission scheme [7, 119, 154, 5] is first proposed for the uplink viewpoint transmission to achieve ultra-reliable low-latency communication (URLLC) requirement for this type of small data transmission.

2.1.2 Contributions

Motivated by above, in this chapter, we develop offline and online learning algorithms for a wireless uplink VR system under proactive retransmission scheme to efficiently maximize the viewpoint prediction accuracy of VR users. The main contributions can be summarized as follows:

- Based on the historical and current viewpoint of the VR user in the real VR dataset, we develop offline and online learning algorithms to predict the viewpoint of the VR user in continuous time slots, in order to capture the dynamical viewpoint preference of VR users over time.

- There are 16 VR videos in the real VR dataset, each VR video has its own property, and we first learn separate learning model for each VR video to predict the viewpoint of its corresponding VR users. When the number of VR videos increases, one learning model for each VR video may occupy much more computation resource and memory of the SBS. Therefore, to evaluate the generality of the viewpoint prediction model, we further propose one learning model for all VR videos. To ensure that the FoV request of each VR user in the VR dataset has the opportunity to be tested and avoid any biased performance, we use K Cross Validation to train the learning model.

- According to [22], the viewpoint of each VR user has strong short-term auto-correlation, which means that the viewpoint can be well predicted based on the historical viewpoint of each VR user. For the offline learning algorithms, we train the n-order Linear Regression (LR), Neural Network (NN), and Recurrent Neural Network (RNN) based on the state-of-the-art Long-short Term Memory (LSTM)/Gated Recurrent Unit (GRU) architecture to predict the viewpoint of the VR user over time. However, the offline learning model cannot adapt to the dynamic changing environment when new VR users exist.

- In the online learning algorithms, we take into account the effect of the failure during the uplink transmission on viewpoint prediction. The uplink transmission may fail because of the unstable wireless channels and interference, which result in incomplete training data and input of the online learning algorithms, and may decrease the prediction accuracy of the learning model. To guarantee the reliability of the uplink transmission, we introduce the proactive retransmission scheme to the uplink VR viewpoint transmission during the online learning. Interestingly, our results shown that the online GRU algorithm for uplink wireless VR network with the proposed retransmission scheme can achieve 95% prediction accuracy.

2.1.3 Organization

The rest of this chapter is organized as follows. The VR data description and analysis are proposed in Section 2.2. The system model and problem formulation are presented in Section 2.3. Learning algorithms for viewpoint prediction is proposed in Section 2.4. The simulation results and conclusions are described in Section 2.5 and Section 2.6, respectively.

2.2 VR Data Description and Analysis

The VR dataset obtained from [4] includes 16 clips of VR videos with 153 VR users, and 969 data samples of the motion in three dimensions, pitch, yaw, and roll, namely, X, Y and Z viewing angles, which are shown in Fig. 2.1. Each dimension is presented by an angle (-180° to 180°), and each data includes the X, Y, and Z viewing angles of each VR user at each time slot.

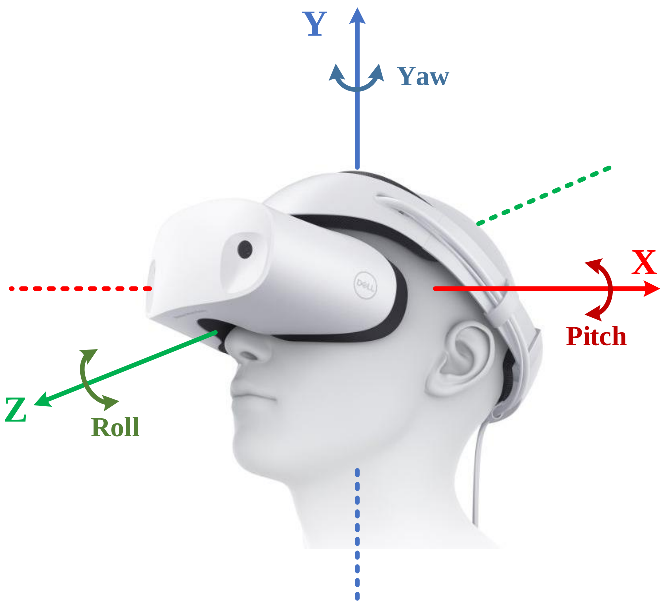


Fig. 2.1 VR user viewing direction.

2.2.1 VR Video Description

The scene of the VR video is shown in Fig. 2.2. From Fig. 2.2, we can observe that the VR videos can be divided into three categories: 1) 7 of them are sports content, including Surfing, Basketball, Boxing, Football, Skiing, and Soccer; 2) 2 of them are Landscapes content, including Grand Canyon, and Survivorman; and 3) 5 of them are Entertainment, including Airplane flight, Underwater game, Roller coaster, Dancing girl, Flying Kite, and Giant Dinosaur.

The 16 VR video clips are downloaded from YouTube. The duration of each VR video is 30 seconds and each VR video is divided into 300 equal parts, which means that each sample point of the VR video lasts for 0.1 seconds. Among these VR videos, 14 of them are 4K resolution, one of them is 2K resolution, and one of them is 1080P. The detailed attributes of each VR video are shown in Table 2.1.

In the experimental data, 153 VR users watched these VR videos, where 35 of them enjoyed all 16 VR video clips, and 118 of them enjoyed 3 to 5 randomly selected VR video

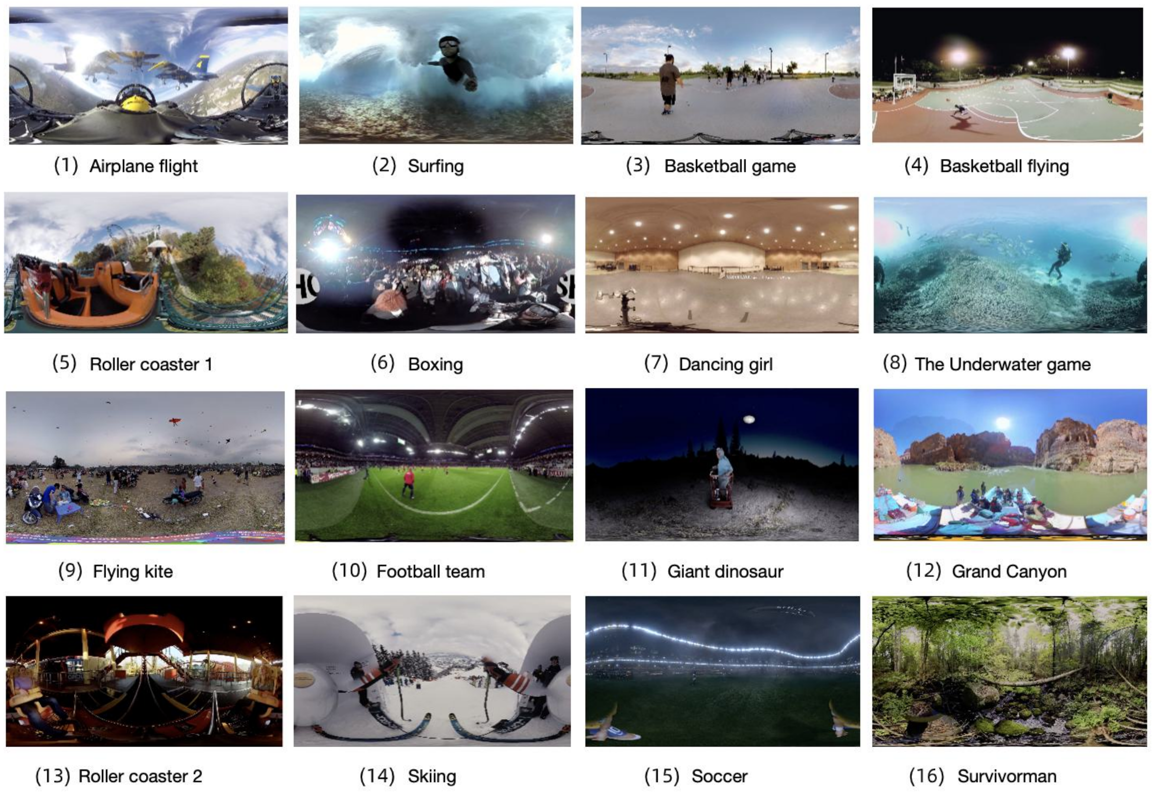


Fig. 2.2 VR video screenshot.

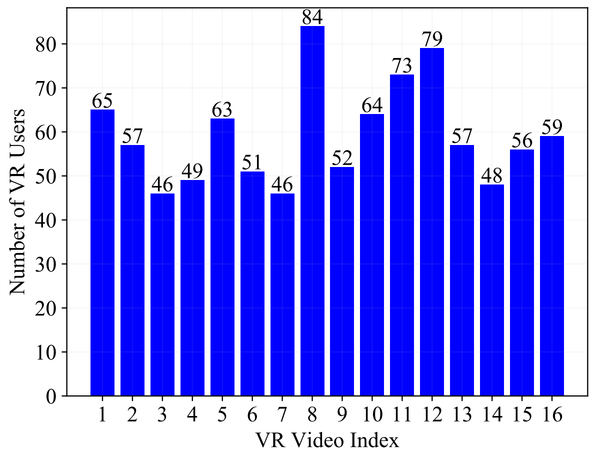


Fig. 2.3 The detailed number of VR users watching each VR video.

clips. The detailed number of VR users watching each VR video is shown in Fig. 2.3. We can obtain that each VR video is watched by an average of 60 VR users, with a minimum of 46, and a maximum of 84. Meanwhile, the age distribution of all VR users is shown in Fig. 2.4. From Fig. 2.4, we can see that more than half of VR users are between 20 and 30. In addition, for all VR users, 38% VR users are female, and 34% VR users wear glasses.

Table 2.1 VR Video Information

Video Number	VR Video Scene	Resolution	Bitrate
(1)	Airplane Flight	4k	13.04 Mbps
(2)	Surfing	4k	23.2 Mbps
(3)	Basketball Game	4k	9.2 Mbps
(4)	Basketball Flying	4k	5.42 Mbps
(5)	Roller Coaster1	4k	31.85 Mbps
(6)	Boxing	4k	4.4 Mbps
(7)	Dancing Girl	4k	6.58 Mbps
(8)	The Underwater World	4k	23.47 Mbps
(9)	Flying Kite	4k	10.98 Mbps
(10)	Football Team	4k	8.24 Mbps
(11)	Giant Dinosaur	1080p	1.35 Mbps
(12)	Grand Canyon	2k	5.12 Mbps
(13)	Roller Coaster2	4k	22.87 Mbps
(14)	Skiing	4k	21.08 Mbps
(15)	Soccer	4k	12.76 Mbps
(16)	Survivorman	4k	29.1 Mbps

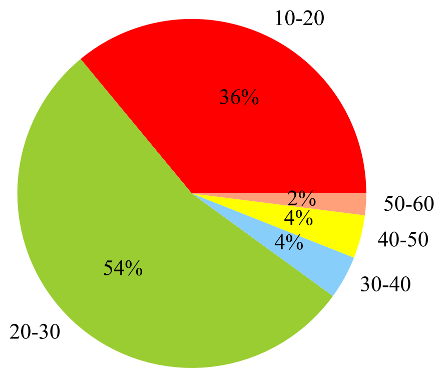


Fig. 2.4 Age distribution of all VR users.

2.2.2 Viewpoint Distribution

In the VR dataset, most VR users have similar viewpoints when enjoying the same video. We plot the viewpoint of all VR users for 16 VR videos in X, Y, and Z angles, respectively, which are shown in Fig. 2.5, 2.6, and 2.7, respectively. The X-axis and Y-axis of the viewpoint distribution figure are VR video playout time and degree of angles at each time slot, respectively. From Fig. 2.5, 2.6, and 2.7, we can know that the viewpoint range of X, Y and Z angles are (-50°, 50°), (-150°, 150°) and (-50°, 50°), respectively.

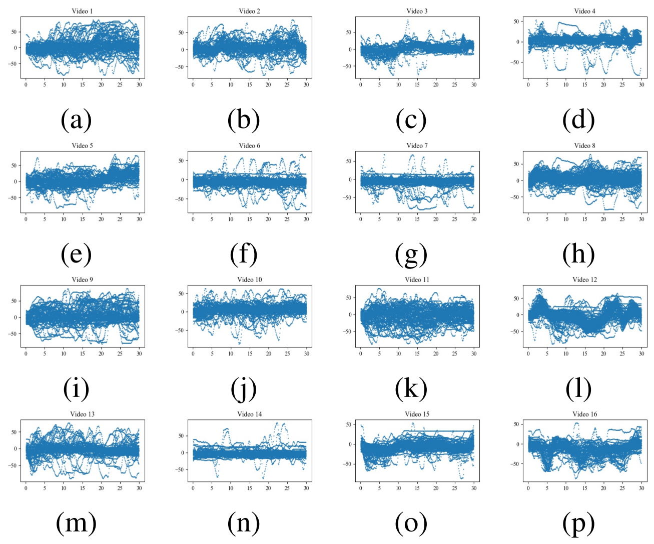


Fig. 2.5 *X* angle distribution of all VR users.

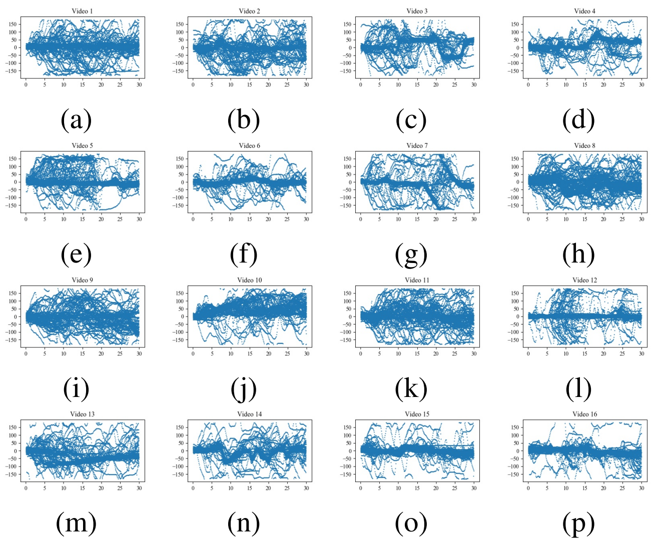


Fig. 2.6 *Y* angle distribution of all VR users.

2.3 System Model and Problem Formulation

We consider a wireless VR system, where a small-cell base station (SBS) is connected to the core network through a fiber link to serve K^{VR} VR users via wireless links, as shown in Fig. 2.8. The SBS is equipped with M antennas and each VR user is equipped with a single antenna, respectively.

2.3.1 Uplink Transmission Model

In the uplink transmission, the VR system is a single-input-multiple-output (SIMO) system, which means that the transmit signal of an antenna is transmitted to a receiver with multiple

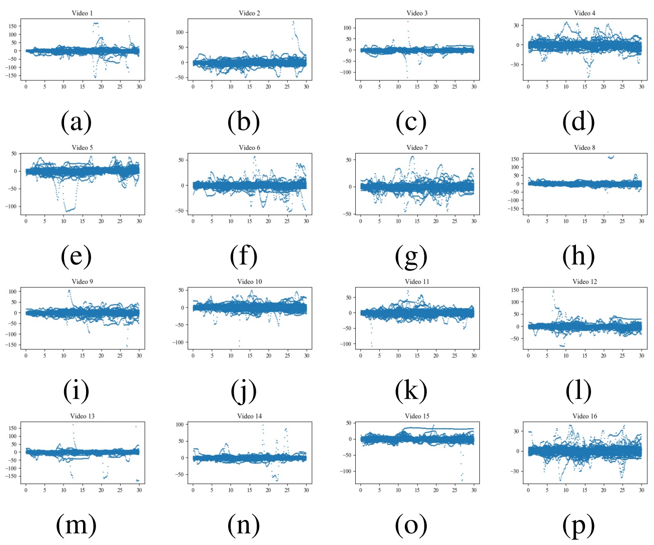


Fig. 2.7 Z angle distribution of all VR users.

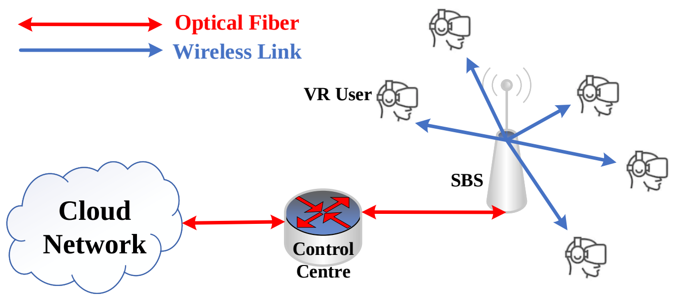


Fig. 2.8 Wireless VR network.

antennas. We use maximum ration combining (MRC) to combine the signals, so that the uplink reliability can be enhanced [1, 35]. At the $(t-1)$th time slot, the SBS will predict the viewpoint $\hat{X}_t^k$ or $\hat{Y}_t^k$ or $\hat{Z}_t^k$ of the kth VR user for the tth time slot. Then, at the tth time slot, the VR user will transmit its actual viewpoint X_t^k or Y_t^k or Z_t^k to the SBS via uplink transmission. The uplink transmission signal from the kth VR user to the SBS at the tth time slot can be denoted as

$$y_{k,t}^{\text{up}} = \mathbf{u}_{k,t}^H \mathbf{h}_{k,t} x_{k,t}^{\text{up}} + \sum_{i=1, i\neq k}^{K^{\text{VR}}} \mathbf{u}_{k,t}^H \mathbf{h}_{i,t} x_{i,t}^{\text{up}} + n_t^{\text{up}}, \tag{2.1}$$

where $\mathbf{h}_{k,t} \in \mathbb{C}^{M\times 1}$ is the uncorrelated Rayleigh fading channel vector between the kth VR user and the SBS at the tth time slot, M is the number of antennas equipped at the SBS, α is the large-scale fading coefficient, $\mathbf{u}_{k,t} \in \mathbb{C}^{M\times 1}$ is the beamforming vector at the SBS, which can be denoted as $\mathbf{u}_{k,t} = \frac{\mathbf{h}_{k,t}}{\|\mathbf{h}_{k,t}\|}$ [53], $x_{k,t}^{\text{up}}$ is the transmit message of the kth VR user at the tth time slot, $\sum_{i=1,i\neq k}^{K} \mathbf{u}_{k,t}^{H}\mathbf{h}_{i,t}x_{i,t}^{\text{up}}$ is the interference from other VR users at the tth time slot, and $n_{k,t}^{\text{up}} \sim \mathcal{CN}(0,\sigma^2)$ is the additive white Gaussian noise at the tth time slot.

Furthermore, at the tth time slot, the data rate between the kth VR user and the SBS can be written as

$$R_{k,t}^{\text{up}} = \log_2\left(1+\frac{|\mathbf{u}_{k,t}^{H}\mathbf{h}_{k,t}|^2}{\sum\limits_{i=1,i\neq k}^{K^{\text{VR}}}|\mathbf{u}_{k,t}^{H}\mathbf{h}_{i,t}|^2+\sigma^2}\right). \tag{2.2}$$

To guarantee the successful uplink transmission, the uplink transmission rate should larger than a threshold $R_{\text{th}}^{\text{up}}$, namely, $R_{k,t}^{\text{up}} \geq R_{\text{th}}^{\text{up}}$. However, the uplink transmission rate of the kth VR user may be smaller than the threshold because of interference or poor channel state information.

To guarantee the reliability of uplink transmission, we consider the proactive retransmission scheme. If the actual viewpoint from the VR user is successfully transmitted via the uplink transmission, the SBS will send an acknowledge (ACK) feedback, otherwise, it will send a non-acknowledge (NACK) feedback.

According to the proactive scheme, the kth VR user will repeat the uplink transmission in consecution transmission time intervals (TTIs) with a maximum number of K_{re} repetitions, but can receive the feedback after each repetition. The kth VR user is allowed to stop repetitions once receiving positive feedback (ACK). We assume that the processing time of the received viewpoint and feedback time at the SBS are one TTI, respectively. For example, when $K_{\text{re}} = 8$, as shown in Fig. 2.9, we can observe that the kth VR user is able to receive the 1st feedback in 4TTIs after the 1st repetition, which means that the minimum round trip time is 4TTIs. Nevertheless, if the kth VR user cannot obtain the ACK at the first round trip time, it will continue waiting for the ACK until $(K_{\text{re}}+3)$TTIs. However, if the kth VR user cannot obtain the ACK during the initial transmission, it needs to continue repetitions until either it receives ACK from the SBS, or the latency is larger than the uplink transmission latency threshold. If the 1st successful uplink transmission of the kth VR user occurs in the lth repetition during the first round trip, the uplink latency of the kth VR user can be computed as

$$T_{k,l} = (l+3)\text{TTIs}, \tag{2.3}$$

Furthermore, the latency after m round trips for the Proactive scheme with a maximum K_{re} repetitions can be derived as

$$\begin{aligned} T_{k,l}^{m} &= (m-1)T_{k,K_{\text{re}}} + T_{k,l} \\ &= [(m-1)(K_{\text{re}}+3)+(l+3)]\text{TTIs}, \end{aligned} \tag{2.4}$$

where $(m-1)T_{k,K_{\mathrm{re}}}$ means that the uplink transmissions in former $(m-1)$ round trips are not successful, and $T_{k,l}$ denotes the successful uplink retransmission in the final mth round trip given in (2.3).

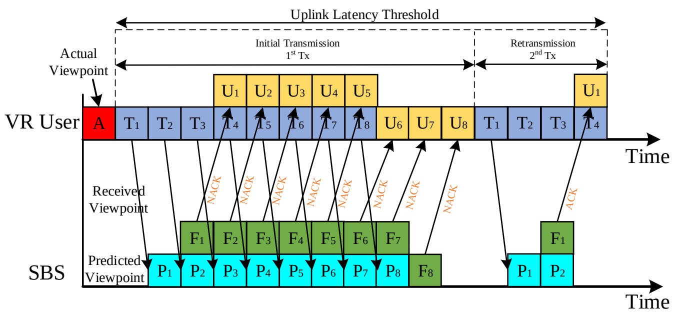


Fig. 2.9 Proactive retransmission scheme.

2.3.2 Viewpoint Prediction Methods

When the VR user enjoys the VR video frame, the viewpoint has three degrees of freedom (pitch, yaw, and roll) and is determined by the rotation angles in X, Y, and Z axis. Therefore, predicting the viewpoint of the VR user is equal to predicting the X, Y, and Z angles. We consider a sliding window to predict the viewpoint of the VR user over time, which is shown in Fig. 2.10. According to Fig. 2.10, the future viewpoint of the VR user is predicted based on the current and past rotation statuses. We assume that the pitch, yaw, and roll angles of the VR user at the tth time slot are X_t, Y_t, and Z_t, respectively. Furthermore, we use $\mathbf{X}_{t:(t+d)} = (X_t, X_{t+1}, \ldots, X_{t+d})$, $\mathbf{Y}_{t:(t+d)} = (Y_t, Y_{t+1}, \ldots, Y_{t+d})$, and $\mathbf{Z}_{t:(t+d)} = (Z_t, Z_{t+1}, \ldots, Z_{t+d})$ to denote the continuous viewpoints in X, Y and Z angles from the tth time slot to the $(t+d)$th time slot.

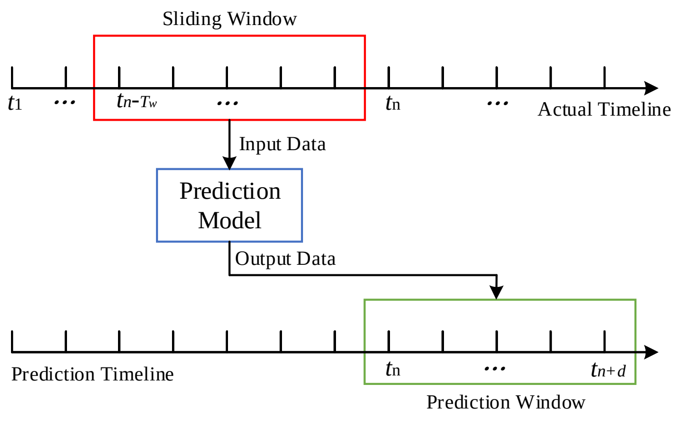


Fig. 2.10 Sliding window.

To predict the future viewpoint $(X_{t+d}, Y_{t+d}, Z_{t+d})$ at the tth time slot, we use previous viewpoint $\mathbf{X}_{(t-T_w):t}$, $\mathbf{Y}_{(t-T_w):t}$, and $\mathbf{Z}_{(t-T_w):t}$, where T_w is the size of the sliding window.

Then, the predicted viewpoint at the $(t+d)$th time slot can be presented as

$$\hat{X}_{t+d} = f_{x,t+d}(\mathbf{X}_{(t-T_w):t}), \tag{2.5}$$

$$\hat{Y}_{t+d} = f_{y,t+d}(\mathbf{Y}_{(t-T_w):t}), \tag{2.6}$$

$$\hat{Z}_{t+d} = f_{z,t+d}(\mathbf{Z}_{(t-T_w):t}), \tag{2.7}$$

where $f_{x,t+d}(.)$, $f_{y,t+d}(.)$, and $f_{z,t+d}(.)$ are prediction function.

To predict the viewpoint of the VR user accurately, we consider two learning algorithms, namely, offline learning and online learning.

Offline Learning

In offline learning algorithms, we propose three methods, which are trained n-order Linear Regression (LR), Neural Network (NN), and Recurrent Neural Network (RNN) based on Long-short Term Memory(LSTM)/Gated Recurrent Unit (GRU) architecture to predict the viewpoints of VR users. Through dividing the VR dataset into training and testing datasets, the VR user data in the training dataset are used to train the models for three offline methods, where the trained models are directly used to predict the viewpoints of VR users. We assume that the offline learning model executed at the VR user side and the input viewpoints of the offline learning model are correct at each time slot. Therefore, the VR user does not need to transmit the actual viewpoints to the SBS, which decreases the overhead and improves the communication efficiency between the SBS and the VR user.

Online Learning

In online learning algorithms, we still use n-order LR, NN, and LSTM/GRU algorithms. Meanwhile, we use Mean Square Error (MSE) [28] as a cost function in the training step to update the parameters of the online learning model, and predict the viewpoints of new VR users. The MSE of the VR users at the tth time slot can be presented as

$$\mathrm{MSE}_t = \frac{1}{K^{\mathrm{VR}}} \sum_{k=1}^{K^{\mathrm{VR}}} (\hat{V}_t^k - V_t^k)^2. \tag{2.8}$$

In (2.8), $\hat{V}_t^k$ can be $\hat{X}_t^k$ or $\hat{Y}_t^k$ or $\hat{Z}_t^k$, and V_t^k can be X_t^k or Y_t^k or Z_t^k. It is because viewing directions can be predicted independently due to their strong auto-correlations [22]. We assume that the online learning model is executed at the SBS, and the VR user needs to transmit the actual viewpoints to the SBS. If the online learning model is executed at the VR user side, the wireless VR device needs a computing unit with high computation capability, which leads to heavy weight of the VR device, and it is inconvenient for the mobile VR user to enjoy the VR video with heavy VR device. Meanwhile, the online learning model costs more energy, which can decrease the working hours of the battery of the VR device. At the tth time slot, the SBS will predict the viewpoint of the VR user for the $(t+1)$th time slot. At the $(t+1)$th time slot, the VR user will transmit the actual viewpoint to the SBS via uplink transmission. Through comparing it with the predicted viewpoint, the SBS will further update the trained learning model to improve the prediction accuracy.

2.3.3 Rendering and Downlink Transmission Model

When the future viewpoint of the VR user is predicted via offline or online learning algorithms, the SBS will render the predicted viewpoint and transmit it to the VR user through downlink transmission in advance. Therefore, the VR interaction latency can be reduced [92]. In the downlink transmission, the VR system is a multiple-input-single-output (MISO) system, it ensures that a stronger signal arrives at the VR user. In this chapter, we mainly focus on prediction and uplink retransmission in the wireless VR system, which can be easily integrated into the rendering and downlink transmission in our previous work [92].

2.3.4 Problem Formulation

For the viewpoint prediction, we use offline and online learning algorithms to minimize the average prediction error of VR users, the optimization problem can be formulated as

$$\min \frac{1}{T_i^{\text{tot}}\widetilde{N}_i}\sum_{t=1}^{T_i^{\text{tot}}}\sum_{k=1}^{\widetilde{N}_i}(\hat{V}_t^k - V_t^k)^2, \tag{2.9}$$

where $\widetilde{N}_i$ is the number of the VR users watching the ith VR video, and T_i^{tot} is the total time slots of the ith VR video.

2.4 Learning Algorithms for Viewpoint Prediction

In offline learning, we directly use the trained n-order LR, NN, and LSTM/GRU network to predict the viewpoint of the VR user in continuous time slots. However, for online learning, the VR user will deliver the actual viewpoint to the SBS via uplink transmission in real-time to further update the models in the n-order LR, NN, and LSTM/GRU learning algorithms and the input of the sliding window, which can improve the prediction accuracy. If the actual viewpoint at a specific time slot has not been successfully delivered to the

SBS, the learning algorithms will predict the viewpoint in the next time slot based on the models trained in the previous time slots, and the input of the sliding window at the current time slot is set to be null.

2.4.1 Offline Learning Algorithm

According to Fig. 2.3, the VR dataset contains dozens of VR users enjoying each VR video, and there are 16 VR videos and 969 VR user samples. Therefore, we have enough user samples to train and test the learning models. To train the learning model, we split the data samples of the VR dataset into training and testing datasets. The training dataset is used to train the learning model, and the testing dataset is used to validate it on data it has never seen before. The classic approach is to do a simple 80%-20% [20], which means that we randomly select 80% data samples of the dataset to construct the training dataset, while the remaining 20% data samples of the dataset are the testing dataset. However, with a simple 80-20 split, there is a possibility of high bias if we have limited data. More importantly, we will miss some important information about the data samples which are not used for training, which is able to get good or bad performance only due to chance. To ensure that each data sample from the original dataset has the chance of appearing in the training and testing dataset, we use K Cross Validation [116].

Through using the K Cross Validation to train the learning models in the VR dataset, each VR user sample has the opportunity of being tested. We split the VR dataset into K_{cross} datasets: one dataset is used for validation, and the remaining $(K_{\text{cross}} - 1)$ datasets are merged into a training dataset for prediction learning model evaluation [111]. In our VR dataset, there are 16 different VR videos. According to Fig 2.5, 2.6, and 2.7, the viewpoint distribution of VR users in each VR video are different. Therefore, for each VR video, we can use its corresponding VR user samples to train a viewpoint prediction learning model. However, if the number of VR videos increases, training one model for one VR video may cost much more energy and occupy much more computation resources and memory of the SBS. Therefore, in order to evaluate the generality of the trained models, we propose two viewpoint prediction learning models, namely, one for a single VR video, and the other for all VR videos. The detailed K Cross Validation for the proposed two viewpoint prediction schemes is introduced as follows:

(a) **One Model for One VR Video:** For each VR video, there are dozens of VR user samples, and we assume that the number of VR user samples of the kth VR video is $\widetilde{N}_k$. We split these dozens of VR user samples into K_{cross}^k datasets, where the number of VR user samples in each sub-dataset is $\widetilde{N}_k / K_{\text{cross}}^k$, and randomly select $(K_{\text{cross}}^k - 1)$ sub-datasets to train the learning model and one sub-dataset to test the trained learning model. Through K_{cross}^k times of training and testing, we can obtain the average prediction error of the

K-folder cross validation. For example, for the 8th VR video with 84 VR user samples, when the number of VR users in each sub-dataset is 10, the VR user samples are divided into 9 sub-datasets. To train the learning models, 8 sub-datasets will be randomly selected to train the learning models and the remaining 1 sub-dataset will be used to test the trained learning models. After 9 times of training and testing, we can obtain the average prediction error.

(b) **One Model for All VR Videos:** For all 16 VR videos and 969 VR user samples, we split these VR user samples according to the index of VR videos. Therefore, there are 16 VR sub-datasets. At each training round of the K cross validation, $16/K_{\text{cross}}$ VR sub-datasets are used for testing, and the remaining $(16-16/K_{\text{cross}})$ VR sub-datasets are merged into a training VR sub-dataset to train the viewpoint prediction model. For example, in Fig. 2.11, we consider 4 cross validation, namely, $K_{\text{cross}} = 4$. At each training iteration, 12 VR sub-datasets will be randomly selected to train the learning model, and the remaining 4 VR sub-datasets will be used to test the trained learning model. After 4 training iterations, we can calculate the average prediction error of these 4 trained learning models.

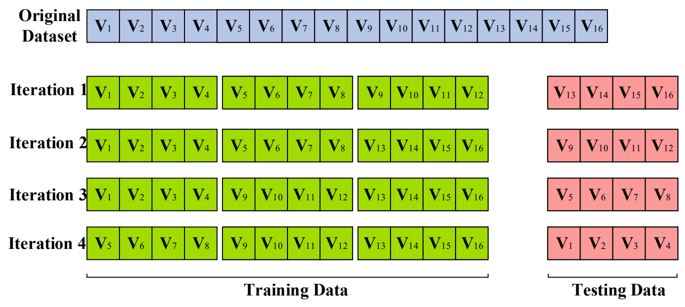

Original Dataset
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
Iteration 1
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12
V13 V14 V15 V16
Iteration 2
V1 V2 V3 V4 V5 V6 V7 V8 V13 V14 V15 V16
V9 V10 V11 V12
Iteration 3
V1 V2 V3 V4 V9 V10 V11 V12 V13 V14 V15 V16
V5 V6 V7 V8
Iteration 4
V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16
V1 V2 V3 V4
Training Data
Testing Data

Fig. 2.11 4 cross validation for training learning models.

2.4.2 Online Learning Algorithm

In the online learning algorithms, the learning model will first be trained via the training dataset through the K Cross Validation described in Section IV-A [1]. Then, for the VR user samples in the testing dataset, at each time slot, each VR user will update its actual viewpoint to the SBS through uplink transmission. The online learning algorithms are introduced in detail as follows.

[1]Through K Cross Validation, the parameters of the proposed online learning algorithms are updated via different sub-datasets, so that the online learning model can be adapted to new VR user samples. Thus, the proposed online learning models can be applied to other datasets.

n-order Linear Regression

n-order LR algorithm uses the least square function to model the nonlinear relationship between the input sliding window and the predicted viewpoint. It is able to fit the nonlinear relationship between the input and output and can be written as

$$\hat{V} = \mathbf{W}^{\mathrm{LR}}\mathbf{g}^{H} + b^{\mathrm{LR}}, \tag{2.10}$$

where $\mathbf{W}^{\mathrm{LR}} = [w_1^{\mathrm{LR}}, w_2^{\mathrm{LR}}, ..., w_n^{\mathrm{LR}}]$ and b^{LR} are parameters of the *n*-order LR model. In (2.10), $\mathbf{g}$ can be $\mathbf{X}_{(t-T_w):t}$ or $\mathbf{Y}_{(t-T_w):t}$ or $\mathbf{Z}_{(t-T_w):t}$), which includes the X or Y or Z viewing angles in T_w time slots. Meanwhile, $\hat{V}_{t+1}$ can be $\hat{X}_{t+1}$ or $\hat{Y}_{t+1}$ or $\hat{Z}_{t+1}$, which is the predicted viewing angles for the $(t+1)$th time slot. The loss function of the *n*-order LR can be calculated as

$$\mathcal{L}_t^{\mathrm{LR}} = \frac{1}{K^{\mathrm{VR}}} \sum_{k=1}^{K^{\mathrm{VR}}} (V_t^k - \hat{V}_t^k)^2. \tag{2.11}$$

Through gradient descent method [117], the parameters $\boldsymbol{\theta}^{\mathrm{LR}} = \{\mathbf{W}^{\mathrm{LR}}, b^{\mathrm{LR}}\}$ can be updated as

$$\boldsymbol{\theta}_{t+1}^{\mathrm{LR}} = \boldsymbol{\theta}_t^{\mathrm{LR}} - \Delta\mathcal{L}_t^{\mathrm{LR}}(\boldsymbol{\theta}_t^{\mathrm{LR}}), \tag{2.12}$$

where $\Delta\mathcal{L}^{\mathrm{LR}}(.)$ is the gradient of the loss function. The proposed Proactive retransmission scheme integrated into the online *n*-order LR is illustrated in Algorithm 1.

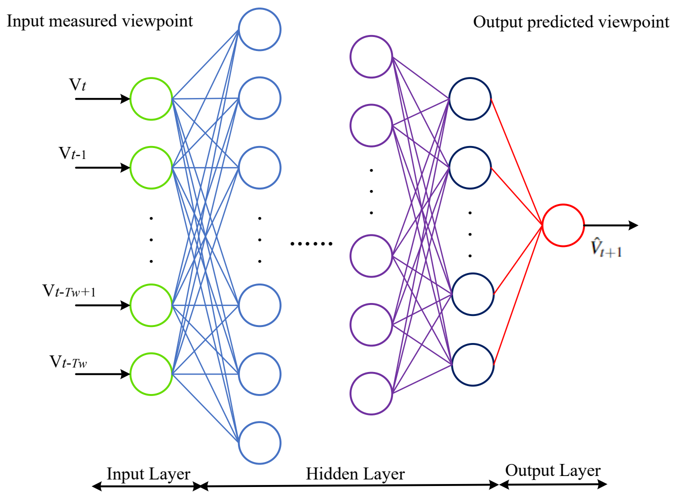


Fig. 2.12 Proposed multi-layer NN architecture.

Neural Network

In the L-layer NN shown in Fig. 2.12, we assume that $\Theta^{\mathrm{NN}} = \{\boldsymbol{\theta}_1^{\mathrm{NN}}, \boldsymbol{\theta}_2^{\mathrm{NN}}, ..., \boldsymbol{\theta}_L^{\mathrm{NN}}\}$ contains L sets of parameters, and the parameters at the lth $(1 \leq l \leq L)$ layer can be denoted as

$\boldsymbol{\theta}_l^{\mathrm{NN}} = \{\boldsymbol{W}_l^{\mathrm{NN}}, \boldsymbol{b}_l^{\mathrm{NN}}\}$, where $\boldsymbol{W}_l^{\mathrm{NN}}$ and $\boldsymbol{b}_l^{\mathrm{NN}}$ are the neurons' weights and bias vector at the lth layer. A feedforward NN with L layers describes a mapping function $f^{\mathrm{NN}}(\mathbf{r}^{\mathrm{NN}}, \boldsymbol{\theta}^{\mathrm{NN}})$, where $\mathbf{r}^{\mathrm{NN}}$ is the input vector. We can obtain the output of the NN through L iterative processing steps, and the output of the lth layer in NN can be written as

$$\mathbf{r}_l^{\mathrm{NN}} = f_l^{\mathrm{NN}}(\mathbf{r}_{l-1}^{\mathrm{NN}}; \boldsymbol{\theta}_l^{\mathrm{NN}}), l = 1, 2, ..., L, \tag{2.13}$$

where $f_l^{\mathrm{NN}}(\mathbf{r}_{l-1}^{\mathrm{NN}}; \boldsymbol{\theta}_l^{\mathrm{NN}})$ is the mapping function calculated by the lth NN layer.

At the tth time slot, we input the historical viewpoint of the VR user and obtain the predicted viewpoint via the feedforward function in the L-layer NN. Then, we use the MSE criterion among the predicted viewpoint and the actual viewpoint of the $(t+1)$th time slot to compute the loss of the NN, which can be denoted as

$$\mathcal{L}_{t,l}^{\mathrm{NN}}(\boldsymbol{\theta}_{t,l}^{\mathrm{NN}}) = \|\boldsymbol{\phi}_{t,l}^{\mathrm{NN}} - \hat{\boldsymbol{\phi}}_{t,l}^{\mathrm{NN}}\|_2, \tag{2.14}$$

where $\boldsymbol{\phi}_{t,l}^{\mathrm{NN}}$ is the desired output of the lth layer in NN, $\hat{\boldsymbol{\phi}}_{t,l}^{\mathrm{NN}}$ is the dependence of the NN's output to the lth layer's parameters. To minimize the loss function, we adopt the backpropagation method based on stochastic gradient descent (SGD) [29]. The parameters of the lth layer can be updated as

$$\boldsymbol{\theta}_{t+1,l}^{\mathrm{NN}} = \boldsymbol{\theta}_{t,l}^{\mathrm{NN}} - \lambda^{\mathrm{NN}} \Delta \mathcal{L}_{t.l}^{\mathrm{NN}}(\boldsymbol{\theta}_{t,l}^{\mathrm{NN}}), \tag{2.15}$$

where $\lambda^{\mathrm{NN}} \in (0, 1]$ denotes the learning rate of the NN and $\Delta\mathcal{L}^{\mathrm{NN}}(.)$ is the gradient of the loss function. The proposed Proactive retransmission scheme integrated into the online NN is presented in Algorithm 1.

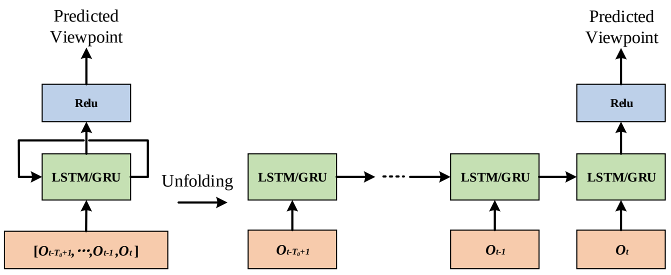


Fig. 2.13 Proposed LSTM/GRU architecture (left) with its unfolding structure (right).

Long-short Term Memory/Gated Recurrent Unit

To capture the dynamics in viewpoint of the VR user for the $(t+1)$th time slot, we use not only the most recent observation $O_t = \{O_t^1, O_t^2, ..., O_t^K\}$, where $O_t^k = \{(X_t^k, Y_t^k, Z_t^k)\}$ is the actual viewpoint of the kth VR user at the tth time slot, but also the previous observations

$H_t = \{O_{t-T_o+1}, ..., O_{t-2}, O_{t-1}\}$, where T_o is the size of the memory window. In order to recognize the viewpoint in continuous time slots, we leverage an RNN model with parameters $\boldsymbol{\theta}^{\text{RNN}} = \{\boldsymbol{W}^{\text{RNN}}, \boldsymbol{b}^{\text{RNN}}\}$, where $\boldsymbol{W}^{\text{RNN}}$ and $\boldsymbol{b}^{\text{RNN}}$ are the neurons' weights and bias vectors of the RNN. The RNN is capable of capturing time correlation of the viewpoint of the VR user, which can help learn the time-varying viewpoint for better prediction accuracy.

The LSTM/GRU layer contains multiple standard LSTM/GRU units and receives the current and historical observations $[O_{t-T_o+1}, ..., O_{t-1}, O_t]$ at the tth time slot and is connected to an output layer with a Relu non-linearity activation function, which is shown in Fig. 2.13. The Relu layer outputs the predicted viewpoint of the VR user for the $(t+1)$th time slot. To update the model parameter $\boldsymbol{\theta}^{\text{RNN}}$, we first use MSE to calculate the loss function, and then use the standard SGD via BackPropagation Through Time (BPTT) [141] to update parameters. At the $(t+1)$th time slot, $\boldsymbol{\theta}^{\text{RNN}}$ can be updated as

$$\boldsymbol{\theta}_{t+1,l}^{\text{RNN}} = \boldsymbol{\theta}_{t,l}^{\text{RNN}} - \lambda^{\text{RNN}} \Delta \mathcal{L}_{t,l}^{\text{RNN}}(\boldsymbol{\theta}_{t,l}^{\text{RNN}}), \tag{2.16}$$

where $\lambda^{\text{RNN}} \in (0,1]$ is the learning rate of the RNN, $\Delta \mathcal{L}_{t,l}^{\text{RNN}}(\boldsymbol{\theta}_{t,l}^{\text{RNN}})$ is the gradient of the loss function $\mathcal{L}_{t,l}^{\text{RNN}}(\boldsymbol{\theta}_{t,l}^{\text{RNN}})$ to train parameters of the RNN. $\mathcal{L}_{t,l}^{\text{RNN}}(\boldsymbol{\theta}_{t,l}^{\text{RNN}})$ can be computed by the MSE as

$$\mathcal{L}_{t,l}^{\text{RNN}}(\boldsymbol{\theta}_{t,l}^{\text{RNN}}) = \|\boldsymbol{\phi}_{t,l}^{\text{RNN}} - \hat{\boldsymbol{\phi}}_{t,l}^{\text{RNN}}\|_2, \tag{2.17}$$

where $\boldsymbol{\phi}_{t,l}^{\text{RNN}}$ is the desired output of the lth layer in RNN, $\hat{\boldsymbol{\phi}}_{t,l}^{\text{RNN}}$ is the dependence of the RNN's output to the lth layer's parameters. The proposed Proactive retransmission scheme integrated into the online RNN is presented in Algorithm 1.

2.4.3 Computational Complexity Analysis for Learning Algorithms

For the computation complexity of n-order LR, it can be computed as $O(nK^{\text{VR}})$ [30], where K^{VR} is the number of VR users. For the NN and RNN based on the LSTM/GRU architecture, they can be given by $O(K^{\text{VR}}\hat{m}\hat{n}\log\hat{n})$ and $O(K^{\text{VR}}\widetilde{m}\widetilde{n}\log\widetilde{n})$, respectively. Here, $\hat{m}$ and $\widetilde{m}$ are the number of layers of NN and RNN, and $\hat{n}$ and $\widetilde{n}$ are the number of units per learning layer of NN and RNN , respectively [49].

2.5 Simulation Results

In this section, we examine the effectiveness of our proposed offline and online learning algorithms on the uplink viewpoint prediction of VR users under the Proactive retransmission scheme. We set the size of the sliding window as 10 time slots and the size of the prediction window as 1 time slot. For the n-order LR, we consider $n = 15$. For the NN,

Algorithm 1 The Proactive retransmission scheme integrated into Online Learning Algorithms with n-order LR, NN, and LSTM/GRU

```
1: Initialize the order $n$ of LR, parameters $\boldsymbol{\theta}^{\mathrm{LR}}$ or $\boldsymbol{\theta}^{\mathrm{NN}}$ or $\boldsymbol{\theta}^{\mathrm{RNN}}$ , and sliding window size $T_w$.
2: Use K Cross Validation to train the parameters of the $n$-order LR, NN, and RNN learning model.
3: **for** t = 1,...,T **do**
4:     Get historical viewpoints from the $(t-T_w)$th time slot to the $(t-1)$th time slot from the updated sliding window.
5:     Use the updated online $n$-order LR, NN, LSTM/GRU to predict the viewpoint of the VR user for the $t$th time slot.
6:     The VR user transmits its actual viewpoint of the $t$th time slot via uplink transmission with the Proactive retransmission scheme.
7:     **if** the uplink transmission is successful **then**
8:         Update parameters $\boldsymbol{\theta}_t^{\mathrm{LR}}$ or $\boldsymbol{\theta}_t^{\mathrm{NN}}$ or $\boldsymbol{\theta}_t^{\mathrm{RNN}}$ of the $n$-order LR, NN and RNN learning model via (2.12), (2.15) and (2.16).
9:         Update the sliding window with the actual required viewpoint of the $t$th time slot.
10:    **else**
11:        $\boldsymbol{\theta}_{t-1}^{\mathrm{LR}} \to \boldsymbol{\theta}_t^{\mathrm{LR}}$ or $\boldsymbol{\theta}_{t-1}^{\mathrm{NN}} \to \boldsymbol{\theta}_t^{\mathrm{NN}}$ or $\boldsymbol{\theta}_{t-1}^{\mathrm{RNN}} \to \boldsymbol{\theta}_t^{\mathrm{RNN}}$.
12:        Update the sliding window with null of the $t$th time slot.
13:    **end if**
14: **end for**
```

we use the fully-connected NN with two hidden layers, where the first and second layers have 12 and 10 neurons, respectively. While for the RNN, it has one hidden layer with 12 units. The learning rate for the learning algorithm is 0.001. For the uplink transmission, we set $M = 30$, $\alpha = 3$, $\mathrm{TTI} = 0.125$ ms, $R_{\mathrm{th}}^{\mathrm{up}} = 2$ MB/s, $\sigma^2 = -110$ dBm, and $K_{\mathrm{re}} = 8$. If the uplink transmission rate calculated by (2.2) is smaller than $R_{\mathrm{th}}^{\mathrm{up}}$, the viewpoint cannot be transmitted successfully within one TTI. We consider the VR users in a limited square area whose side length is 100 meters. The proactive uplink retransmission scheme is standardized in 5G, thus, the selected VR center frequency for VR transmission is 5G band [9], and the selected frequency band is from 5.15 GHz to 5.23 GHz [10, 6], which can guarantee the latency and reliability of the VR uplink transmission. For simplicity, we use “w/ Proac" and “w/o Proac" to represent “with Proactive Retransmission" and “without Proactive Retransmission”, respectively. Meanwhile, in the Genie-aided scheme, the online learning model is directly trained with the correct actual viewpoint of each VR user at each time slot, which is the upper bound of the online learning algorithm with a proactive retransmission scheme and cannot be reached in the practical wireless VR system.

2.5.1 VR Dataset Processing

We first save all VR user samples in a MATLAB file. Then, we use Python 3.6 to delete the useless rows and columns, and import the VR user data into training and testing datasets. According to [22], the motion of the VR user has strong short-term auto-correlations in all three dimensions. Due to the fact that auto-correlations are much stronger than the correlation between these three dimensions, the angles in each direction can be trained independently and separately. According to Fig. 2.5, 2.6, and 2.7, we can obtain that the range of Y angle distribution is much larger than that of X and Z. Therefore, for simplicity, we use offline and online learning algorithms to predict Y angle of VR users in this section, however, our algorithms can also be used for the prediction of X and Z angles. Meanwhile, we use different VR user sample lists in Python to differentiate the training and testing datasets.

2.5.2 Viewpoint Prediction

The simulation results of our proposed two viewpoint prediction learning models, namely, one training model for a single VR video, and one training model for all VR videos, are introduced as follows:

(a) **One Training Model for One VR Video:** In this scheme, for each VR video, we use the VR user samples in the training datasets to train the offline and online learning models to predict the Y angle of VR users in the testing datasets, and average the prediction error of all VR videos.

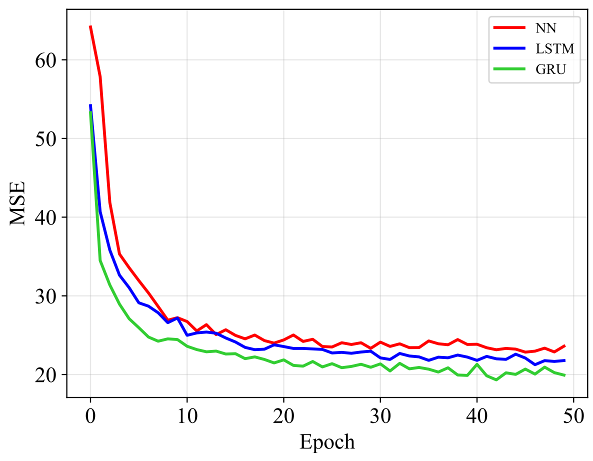


Fig. 2.14 Loss of offline NN, LSTM, and GRU algorithms of each epoch.

Fig. 2.14 plots the loss of offline NN, LSTM, and GRU algorithms of each epoch. It is seen that the performance of the offline GRU algorithm outperforms that of LSTM and

NN. This is because the structure of the LSTM is more complex than that of GRU so that the parameters of the GRU can be trained faster and easier to be modified [32].

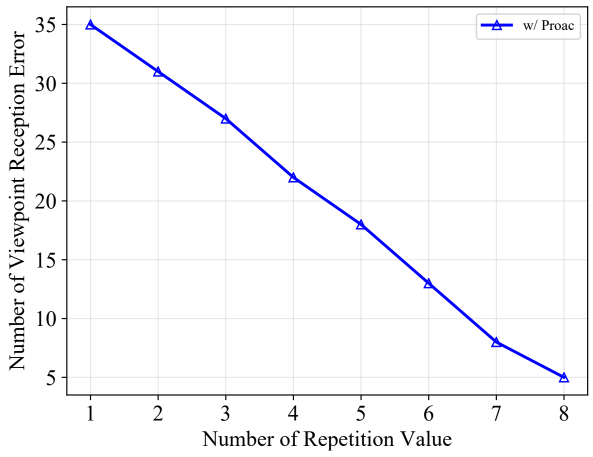


Fig. 2.15 The number of viewpoint reception errors versus the number of repetition values of the proactive retransmission scheme.

Fig. 2.15 plots the number of viewpoint reception errors versus the number of repetition values of the proactive retransmission scheme. We can see that the number of viewpoint reception errors decreases with the increasing number of repetition values. This is because the success probability of the uplink transmission increases with the increasing number of repetition values of the proactive retransmission scheme [94].

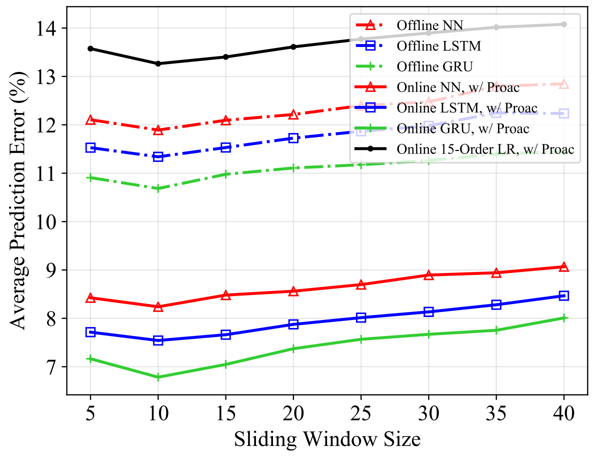


Fig. 2.16 Average prediction error of offline/online learning algorithms via different size of sliding window with a Proactive retransmission scheme.

Fig. 2.16 shows the average prediction error of offline/online learning algorithms via different size of sliding window for uplink VR viewpoint transmission with a proactive

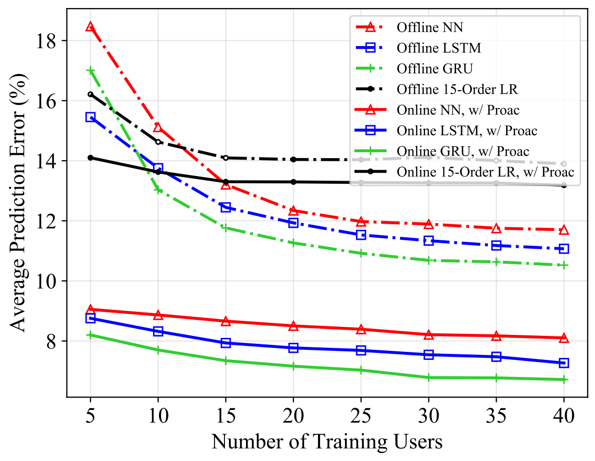


Fig. 2.17 Average prediction error of different number of VR users in the training dataset to train the learning model via offline/online 15-order LR, NN, LSTM, and GRU with a Proactive retransmission scheme.

retransmission scheme. It is noted that the average prediction error of the offline/online 15-order LR, NN, LSTM, and GRU algorithms are not significantly affected by changing the size of the sliding window due to their capability to adapt to the viewpoint preference. When the size of the sliding window is 10, it can obtain the best performance.

Fig. 2.17 plots the average prediction error of different number of VR users in the training dataset to train the learning model via offline/online 15-order LR, NN, LSTM, and GRU for uplink VR viewpoint transmission with a proactive retransmission scheme. For the offline learning algorithms, we observe that the average prediction error becomes smaller with the increasing number of VR users. With the increasing number of VR users, the offline learning algorithms can be trained to adapt to the viewpoints of VR users much more accurately. It is also seen that the performance of the LSTM/GRU is better than that of the NN. This is because the LSTM/GRU is able to capture the correlation of viewpoints in continuous time slots. In addition, it can be seen that the average prediction error of 15-order LR algorithm is much higher than that of offline/online NN, LSTM, and GRU. It is because the learning structure of the LR algorithm is simpler than that of NN, LSTM, and GRU, and its ability to be fit for the nonlinear VR data is worse than that of the NN, LSTM, and GRU. In addition, the LR algorithm may get overfit with so many VR users training the LR model.

Meanwhile, for the proactive retransmission scheme integrated into the online learning algorithms in Fig. 2.17, it is interesting to note that their average prediction errors are much smaller than that of offline learning algorithms and change slightly with the increasing number of VR users. This is due to that through updating the parameters in the

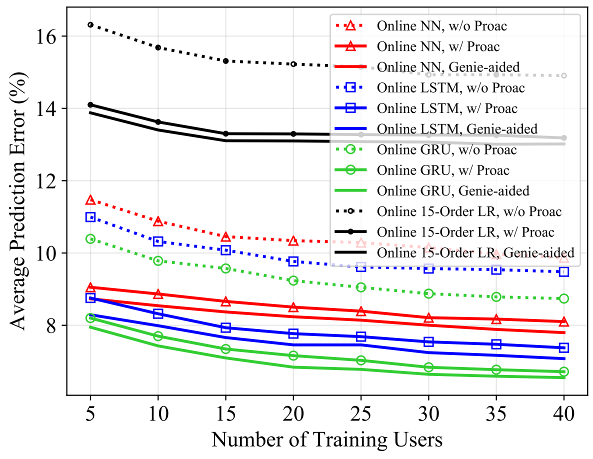


Fig. 2.18 Average prediction error of different number of VR users in the training dataset to train the learning model via online 15-order LR, NN, LSTM, and GRU with/without a Proactive retransmission scheme.

trained learning models, the online learning algorithms are able to adapt to the viewpoint preferences of new VR users over time. Thus, the prediction accuracy is improved.

Fig. 2.18 plots the average prediction error for various number of VR users in the training dataset to train the learning model via online 15-order LR, NN, LSTM, and GRU for uplink VR viewpoint transmission with/without a proactive retransmission scheme. We can observe that the performance of the proactive retransmission scheme integrated into the online learning algorithm is better than that without a proactive retransmission scheme and is close to the performance of the Genie-aided scheme. In the uplink viewpoint transmission without proactive retransmission, each VR user only transmits its actual viewpoint to the SBS once even this transmission fails. This transmission failure is usually because of the unstable channel state and interference from other VR users. To cope with this, the proactive retransmission scheme is applied here to improve the success transmission of uplink transmission [94], and the online learning algorithms are capable of better capturing historical trends of viewpoint preference of the VR user, which can further improve the prediction accuracy. While in the Genie-aided scheme, the online learning algorithms are directly trained with actual viewpoints of VR users.

(b) **One Training Model for All VR Videos:** In this model, for all 16 VR videos, we consider 4 Cross Validation shown in Fig. 2.11 and use VR user samples in the training datasets to train offline/online 15-order LR, NN, and GRU learning models to predict Y angle of VR users in testing datasets.

Fig. 2.19 plots the average prediction error of offline/online 15-order LR, NN and GRU integrated with a proactive retransmission scheme in continuous time slots. For the offline learning algorithms, it can be seen that the performance of the GRU is a bit better

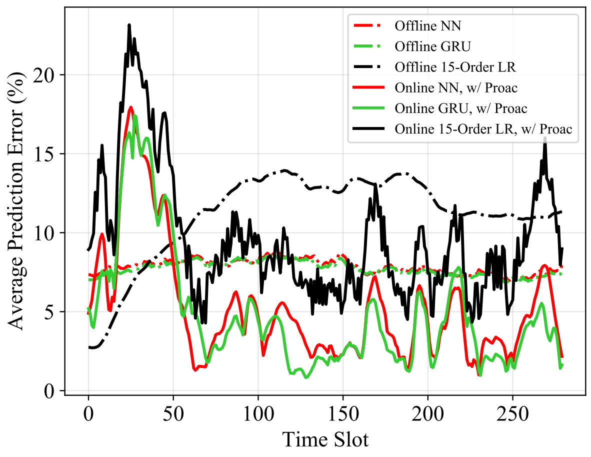


Fig. 2.19 Average prediction error of offline/online 15-order LR, NN and GRU with a Proactive retransmission scheme in continuous time slots.

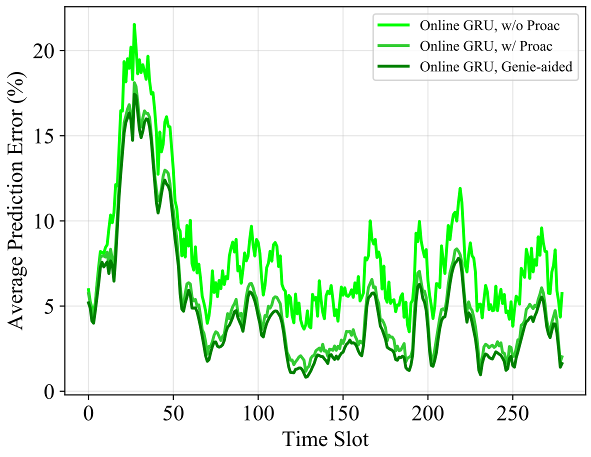


Fig. 2.20 Average prediction error of the online GRU algorithm with/without a Proactive retransmission scheme in continuous time slots.

than that of the NN. Meanwhile, it can be observed that at the beginning 30 time slots, the performance of the 15-order LR is better than that of NN and GRU. It is because according to Fig. 2.6, in the beginning, when the VR user watches the VR video, its viewpoint mainly focuses on the zero point and the 15-order LR fits well at the beginning. However, after 50 time slots, the performance of GRU is much better than that of 15-order LR. This is due to that after 50 time slots, the viewpoint of the VR user will change substantially as shown in Fig. 2.6, and the GRU is able to capture the correlation of the viewpoint of the VR user over continuous time slots.

Furthermore, it is also noted that at the beginning, there are large fluctuations in the performance of the proactive retransmission scheme integrated into online learning algorithms. It is because the parameters in the online learning algorithms should be modified to capture the viewpoint preference of the VR user. In addition, when the viewpoint of the VR user changes over time, the online learning algorithms need to further update their parameters to be fit for the viewpoint changing of VR users. Therefore, there are small fluctuations in the performance of online learning algorithms.

Fig. 2.20 plots the average prediction error of the online GRU algorithm of uplink viewpoint transmission with/without a proactive retransmission scheme in continuous time slots. We can obtain that the performance of the proactive retransmission scheme with the online GRU algorithm is still better than that of the scheme without a proactive retransmission scheme. Meanwhile, it can be seen that the performance of the Genie-aided online GRU algorithm slightly outperforms that of the online GRU algorithm with the proactive retransmission scheme, while their gap is small.

2.6 Conclusions

In this chapter, offline and online learning algorithms for uplink wireless VR networks with a proactive retransmission scheme were developed to predict the viewpoints of wireless VR users with real VR datasets. Specifically, for the offline learning algorithm, K Cross Validation was used to train offline n-order LR, NN, and LSTM/GRU learning algorithms for each VR video and all VR videos. The trained offline learning algorithms were used to directly predict the viewpoint of the VR user. In the online learning algorithms, the online n-order LR, NN, and LSTM/GRU algorithms would update their parameters according to the actual viewpoints delivered from new VR users through uplink transmission, which could further improve the prediction accuracy. Meanwhile, a proactive retransmission scheme was introduced to the online learning algorithms to enhance the reliability of uplink transmission, which can correctly update the parameters and input viewpoints of online learning models. Simulation results show that our proposed online GRU algorithm with the proactive retransmission scheme can achieve the highest prediction accuracy. Meanwhile, our results show that the one model for one VR video and one model for all VR videos achieve 92% and 98% prediction accuracy, respectively. Thus, the proposed learning models can be applied to other datasets, and our results are representative due to the 16 diverse VR videos in our dataset.

Chapter 3

Learning-based Prediction, Rendering and Association Optimization for MEC-enabled Wireless VR Network

3.1 Introduction

For real-time interactive VR applications, the latest 2D video contents need to be first delivered to the VR device via wired/wireless communication, and then rendered to 3D VR videos 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' motions, including head and eye movements. According to [3], 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.

3.1.1 Motivation

Rendering real-time VR videos with high quality demands a 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 the network edge, so as to enable computation-intensive and latency-critical applications at the resource-limited mobile devices, which promise a dramatic reduction in latency and energy consumption [101].

Shifting the FoV rendering task from the VR device to the 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 a single VR user selecting a unique FoV, the downlink transmission cost of the network can be further decreased. In practice, each MEC has a 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 migrating the rendered FoV wired to other MECs, namely, rendered FoV migration.

3.1.2 Contributions

As pointed out by [69], 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 chapter, we focus on optimizing the QoE of VR users with interactive VR applications in MEC-enabled wireless VR networks, and we develop a decoupled learning strategy to efficiently optimize the QoE in wireless VR systems, which can improve the training efficiency [77, 78]. The main contributions can be summarized as follows:

- We propose a MEC-enabled wireless VR network, where the field of view (FoV) of each VR user can be real-time predicted, and the rendering of VR content is moved from the VR device to the 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 show 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 DRL strategies based on Deep Q-Network (DQN) and Actor Critic (AC) [40, 87, 41, 44, 127, 88, 122] to maximize the long-term QoE of

VR users, via determining the 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 show substantial gain in both QoE and VR interaction latency. Interestingly, the rendering model migration further improves these gains.

3.1.3 Organization

The rest of this chapter is organized as follows. The system model and problem formulation are proposed in Section 3.2. RNN-based FoV prediction and DRL-based MEC rendering, migration, and association scheme are presented in Section 3.3. The simulation results and conclusions are described in Section 3.4 and Section 3.5, respectively.

3.2 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 serve K_{VR} VR users via wireless links, as shown in Fig. 3.1. 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 [132], 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 [1].

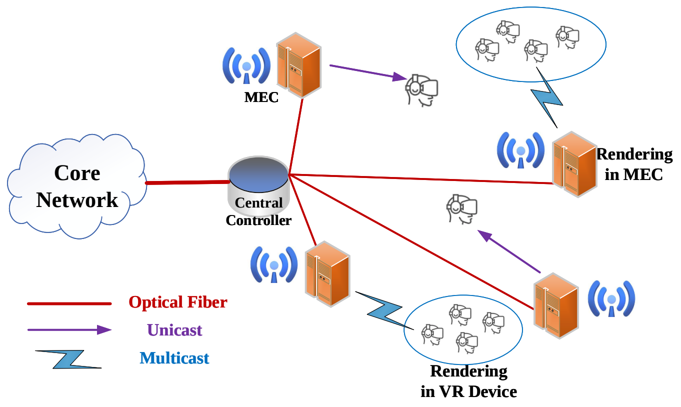


Fig. 3.1 Wireless VR system in cellular network.

[1] In a 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.

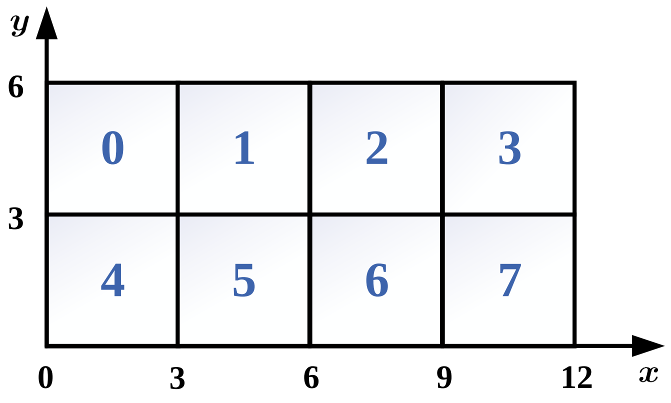


Fig. 3.2 Brownian motion for FoV selection.

3.2.1 System Model

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

FoV Selection and Prediction

When VR users enjoy VR videos, because of the restricted area of vision in human eyes, they usually watch only a portion of the VR video, namely, the Field of View (FoV) [23, 8], which specifies a $150^\circ \times 135^\circ$ (i.e., diagonal 200°) FoV requirement. As the FoV is part of the actual rendered 3D VR video, the MEC 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 $N_{\rm FoV}$. 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 enjoy in the VR environment, and this is the so-called fixational eye movement [48, 55]. The fixational eye movements have been widely assumed to be a random uncorrelated process following Brownian motion [115]. Therefore, the Brownian motion can be used to model the eye movement of each VR user corresponding to different FoVs over time. 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 $N_{\rm FoV}$ FoVs at each time. When the VR user's eyes move inside the cubic in Fig. 3.2, the corresponding FoV will be selected. According to Brownian motion [26, 80], 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

$$\triangle S_k(t) = \{\mathcal{N}(0, 2D_k(t)), \mathcal{N}(0, 2D_k(t))\}. \tag{3.1}$$

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. 3.2. If the VR user selects the 2nd FoV at a certain time slot, in the next time slot, it will select one of the 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.

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

$$\widehat{F_t^{\mathrm{oV}}} = F_t^{\mathrm{oV}}. \tag{3.2}$$

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 as

$$P_{\mathrm{FoV}} = \frac{N_{\widetilde{F_t^{\mathrm{oV}}}}}{N_{F_t^{\mathrm{oV}}}} \times 100\%, \tag{3.3}$$

where $N_{\widetilde{F_t^{\mathrm{oV}}}}$ is the number of correct FoV predictions and $N_{F_t^{\mathrm{oV}}}$ is the total number of required FoVs.

FoV Rendering

When a 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. 3.3 (a) and (b), when the rendering function is executed at the MEC, a stitched 2D image, whose color mode is RGB, will be rendered into the

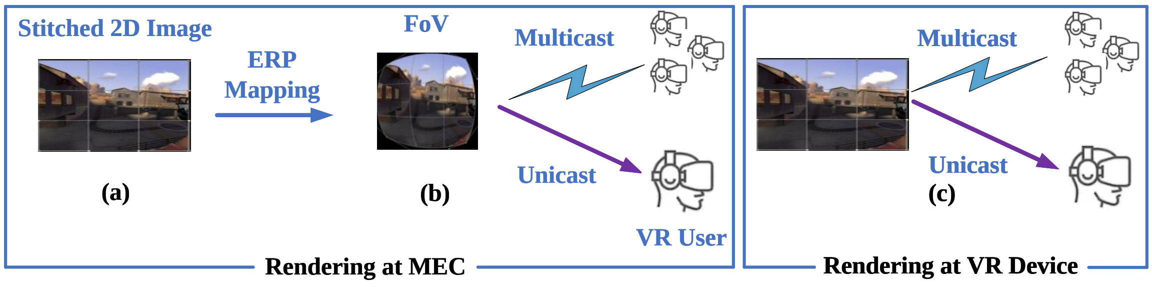


Fig. 3.3 FoV rendering.

required FoV through equirectangular projection (ERP) mapping [8] for downlink multicast or unicast transmission.

- When the rendering function is processed at the VR device, the stitched 2D pictures 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. 3.3 (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 perform 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.

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 a unique FoV, respectively. Let us consider a set of $\mathcal{B}$ ={1, 2,..., B} MECs, and each MEC is equipped with N transmit antennas. These B MECs serve the downlink transmission for B VR user groups $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_B\}$ with a 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 muslticast groups and U unicast groups can be denoted as the sets $\mathcal{V}^{\text{mul}} = \{\mathcal{V}_1^{\text{mul}}, \mathcal{V}_2^{\text{mul}}, ..., \mathcal{V}_M^{\text{mul}}\}$ and $\mathcal{V}^{\text{uni}} = \{\mathcal{V}_1^{\text{uni}}, \mathcal{V}_2^{\text{uni}}, ..., \mathcal{V}_U^{\text{uni}}\}$, respectively. With the number of VR users in the kth multicasting group denoted as $|\mathcal{V}_k^{\text{mul}}|$, the total number of VR users can be calculated as $K_{\text{VR}} = \sum_{k=1}^{M} |\mathcal{V}_k^{\text{mul}}| + U$. Note that each VR user can only be assigned to just one group.

3.2.2 Mathematical Model

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

$$C = 3 \times 8 \times N_p \times N_p \times V = 48N_p^2, \tag{3.4}$$

where 3 presents the red, green and blue color in RGB mode, 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 [69], 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 [3], 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.

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

$$\begin{aligned} y_{j_k,b}^{\text{mul}}(t) = \mathbf{h}_{j_k,b}^{H}(t)\mathbf{v}_{j,b}^{\text{mul}}(t)x_b^{\text{mul}}(t) + \\ \sum_{\mathcal{V}_i^{\text{mul}} \in \mathcal{V}^{\text{mul}}/\mathcal{V}_b^{\text{mul}}, m \in \mathcal{V}_i^{\text{mul}}} \mathbf{h}_{j_k,i}^{H}(t)\mathbf{v}_{m,i}^{\text{mul}}(t)x_i^{\text{mul}}(t) + \\ \sum_{\mathcal{V}_l^{\text{uni}} \in \mathcal{V}^{\text{uni}}, u \in \mathcal{V}_l^{\text{uni}}} \mathbf{h}_{j_k,l}^{H}(t)\mathbf{v}_{u,l}^{\text{uni}}(t)x_u^{\text{uni}}(t) + n(t), \end{aligned} \tag{3.5}$$

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 multiscast VR users. $\mathbf{v}^{\text{mul}}_{j,b}(t) \in \mathbb{C}^{M\times 1}$ and $\mathbf{v}^{\text{uni}}_{u,l}(t) \in \mathbb{C}^{M\times 1}$ 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 (3.5), $x^{\text{mul}}_b(t)$ and $x^{\text{uni}}_u(t)$ 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 $x^{\text{mul}}_b(t)$ and $x^{\text{uni}}_u(t)$ are independent from each other. Meanwhile, $\sum_{\mathcal{V}^{\text{mul}}_i \in \mathcal{V}^{\text{mul}}/\mathcal{V}^{\text{mul}}_b, m\in\mathcal{V}^{\text{mul}}_i} \mathbf{h}^H_{j_k,i}(t)\mathbf{v}^{\text{mul}}_{m,i}(t)x^{\text{mul}}_i(t)$ and $\sum_{\mathcal{V}^{\text{uni}}_l \in \mathcal{V}^{\text{uni}}, u\in\mathcal{V}^{\text{uni}}_l} \mathbf{h}^H_{j_k,l}(t)\mathbf{v}^{\text{uni}}_{u,l}(t)x^{\text{uni}}_u(t)$ are the interference from the other MECs that provide FoVs for the VR users in other multicast and unicast groups, respectively. In addition, $n(t) \sim \mathcal{CN}(0,\sigma^2)$ is the additive white Gaussian noise. Meanwhile, we do not consider transmit power control in downlink transmission.

Based on (3.5), 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

$$R^{\text{mul}}_{j_k,b}(t) = \log_2\left(1 + \frac{|\mathbf{h}^H_{j_k,b}(t)\mathbf{v}^{\text{mul}}_{j,b}(t)|^2}{I^{\text{mul}}_{j_k,b}(t)+\sigma^2}\right), \tag{3.6}$$

where

$$I^{\text{mul}}_{j_k,b}(t) = \sum_{\substack{\mathcal{V}^{\text{mul}}_i \in \mathcal{V}^{\text{mul}}/\mathcal{V}^{\text{mul}}_b \\ m\in\mathcal{V}^{\text{mul}}_i}} |\mathbf{h}^H_{j_k,i}(t)\mathbf{v}^{\text{mul}}_{m,i}(t)|^2 + \sum_{\substack{\mathcal{V}^{\text{uni}}_l \in \mathcal{V}^{\text{uni}} \\ u\in\mathcal{V}^{\text{uni}}_l}} |\mathbf{h}^H_{j_k,l}(t)\mathbf{v}^{\text{uni}}_{u,l}(t)|^2. \tag{3.7}$$

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

$$\begin{aligned} y^{\text{uni}}_{k,b}(t) = {} & \mathbf{g}^H_{k,b}(t)\mathbf{v}^{\text{uni}}_{k,b}(t)x^{\text{uni}}_b(t) + \\ & \sum_{\mathcal{V}^{\text{uni}}_i \in \mathcal{V}^{\text{uni}}/\mathcal{V}^{\text{uni}}_b, u\in\mathcal{V}^{\text{uni}}_i} \mathbf{g}^H_{k,i}(t)\mathbf{v}^{\text{uni}}_{u,i}(t)x^{\text{uni}}_i(t) + \\ & \sum_{\mathcal{V}^{\text{mul}}_l \in \mathcal{V}^{\text{mul}}, m\in\mathcal{V}^{\text{mul}}_l} \mathbf{g}^H_{k,l}(t)\mathbf{v}^{\text{mul}}_{m,l}(t)x^{\text{mul}}_l(t) + n(t), \end{aligned} \tag{3.8}$$

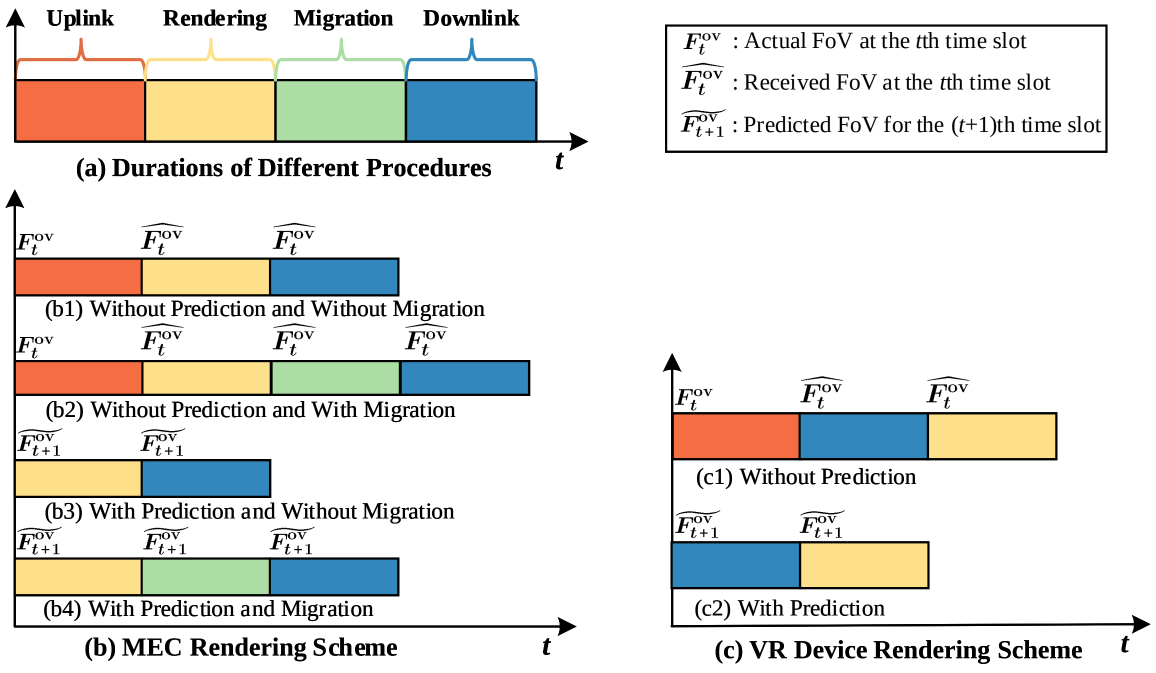
where $\mathbf{g}_{k,b}(t) \in \mathbb{C}^{M\times 1} \sim \mathcal{CN}(\mathbf{0}, \beta\mathbf{I}_M)$ is the uncorrelated Rayleigh fading channel vector between the bth MEC and the kth VR user in the unicast group, and β is the large-scale fading coefficient for the unicast VR users. Meanwhile, $\sum_{\mathcal{V}^{\text{uni}}_i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

Based on (3.8), the unicast transmission rate between the kth VR user and the bth MEC at the tth time slot can be written as

$$R_{k,b}^{\text{uni}}(t) = \log_2\left(1 + \frac{|\mathbf{g}_{k,b}^H(t)\mathbf{v}_{k,b}^{\text{uni}}(t)|^2}{I_{k,b}^{\text{uni}}(t) + \sigma^2}\right), \tag{3.9}$$

where

$$I_{k,b}^{\text{uni}}(t) = \sum_{\substack{\mathcal{V}_i^{\text{uni}} \in \mathcal{V}^{\text{uni}}/\mathcal{V}_b^{\text{uni}} \\ u \in \mathcal{V}_i^{\text{uni}}}} |\mathbf{g}_{k,i}^H(t)\mathbf{v}_{u,i}^{\text{uni}}(t)|^2 + \sum_{\substack{\mathcal{V}_l^{\text{mul}} \in \mathcal{V}^{\text{mul}} \\ m \in \mathcal{V}_l^{\text{mul}}}} |\mathbf{g}_{k,l}^H(t)\mathbf{v}_{m,l}^{\text{mul}}(t)|^2. \tag{3.10}$$



Uplink
Rendering
Migration
Downlink
(a) Durations of Different Procedures
t
F_t^{ov} : Actual FoV at the tth time slot
$\widehat{F_t^{\text{ov}}}$: Received FoV at the tth time slot
$\widetilde{F_{t+1}^{\text{ov}}}$: Predicted FoV for the (t+1)th time slot
F_t^{ov}
$\widehat{F_t^{\text{ov}}}$
(b1) Without Prediction and Without Migration
(b2) Without Prediction and With Migration
$\widetilde{F_{t+1}^{\text{ov}}}$
(b3) With Prediction and Without Migration
(b4) With Prediction and Migration
(b) MEC Rendering Scheme
(c1) Without Prediction
(c2) With Prediction
(c) VR Device Rendering Scheme

Fig. 3.4 VR interaction latency of the proposed MEC and VR device rendering schemes.

VR Interaction Latency

As defined in [69, 9], 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 consists 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

$$T^{\mathrm{loop}} = T^{\mathrm{uplink}} + T^{\mathrm{render}} + T^{\mathrm{migration}} + T^{\mathrm{downlink}}, \tag{3.11}$$

as shown in Fig. 3.4 (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. 3.4 (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 $\widehat{F_t^{\mathrm{oV}}}$ is executed at each MEC with associated VR user. As shown in Fig. 3.4 (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

$$\begin{aligned} T_k^{\mathrm{loop}} &= T_k^{\mathrm{uplink}} + T_k^{\mathrm{render}} + T_k^{\mathrm{downlink}} \\ &= T_k^{\mathrm{uplink}} + \frac{f_k^{\mathrm{MEC}}\mathcal{M}}{F_k^{\mathrm{MEC}}} + \frac{C}{\mathcal{C}_{\mathcal{R}} R_{k,k}^{\mathrm{down}}}, \end{aligned} \tag{3.12}$$

where $R_{k,k}^{\mathrm{down}} \in \{R_{k,k}^{\mathrm{mul}}, R_{k,k}^{\mathrm{uni}}\}$, and $R_{k,k}^{\mathrm{mul}}$ and $R_{k,k}^{\mathrm{uni}}$ are given in (3.6) and (3.9).

a-2) With prediction: As shown in Fig. 3.4 (b3), based on the uplink received FoVs at the tth time slot $\widehat{F_t^{\mathrm{oV}}}$ and several previous time slots, we predict the FoV preference of each VR user at the $(t+1)$th time slot $\widetilde{F_{t+1}^{\mathrm{oV}}}$. For the kth VR user directly served by the kth MEC, the VR interaction latency with predicted FoV can be written as (3.12) with $T_k^{\mathrm{uplink}} = 0$.

(b) **MEC Rendering with Migration:** *b-1) Without prediction:* For the VR users requesting the same FoV $\widehat{F_t^{\mathrm{oV}}}$ through uplink transmission, only one MEC is selected for rendering, and the rendered FoV can be migrated to other MECs. As shown in Fig. 3.4 (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

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{render}} + T_k^{\text{downlink}} \\ &= T_k^{\text{uplink}} + \frac{f_b^{\text{MEC}}\mathcal{M}}{F_b^{\text{MEC}}} + \frac{C}{\mathcal{C}_{\mathcal{R}} R_{k,b}^{\text{down}}}, \end{aligned} \tag{3.13}$$

where $R_{k,b}^{\text{down}} = R_{k,b}^{\text{mul}}$ and $R_{k,b}^{\text{mul}}$ is given in (3.6).

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

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{render}} + T_k^{\text{migration}} + T_k^{\text{downlink}} \\ &= T_k^{\text{uplink}} + \frac{f_b^{\text{MEC}}\mathcal{M}}{F_b^{\text{MEC}}} + \frac{\hat{L}_{k,b}}{R^{\text{fiber}}} + \frac{C}{\mathcal{C}_{\mathcal{R}} R_{k,k}^{\text{down}}}, \end{aligned} \tag{3.14}$$

where $R_{k,k}^{\text{down}} = R_{k,k}^{\text{mul}}$ and $R_{k,k}^{\text{mul}}$ is given in (3.6).

b-2) With prediction: As shown in Fig. 3.4 (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 (3.13) 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 (3.14) with $T_k^{\text{uplink}} = 0$.

(c) **VR Device Rendering:** *c-1) Without prediction:* According to Fig. 3.4 (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

$$\begin{aligned} T_k^{\text{loop}} &= T_k^{\text{uplink}} + T_k^{\text{downlink}} + T_k^{\text{render}} \\ &= T_k^{\text{uplink}} + \frac{\mathcal{M}}{\mathcal{C}_{\mathcal{R}} R_{k,k}^{\text{down}}} + \frac{f_k^{\text{VR}}\mathcal{M}}{F_k^{\text{VR}}}, \end{aligned} \tag{3.15}$$

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 (3.6) and (3.9).

c-2) With prediction: As shown in Fig. 3.4 (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 (3.15) with $T_k^{\text{uplink}} = 0$.

VR Quality of Experience

The quality of the FoV can be influenced by many factors, such as blockiness, blur, contrast distortion, freezing, color depth, and sharpness [138, 97]. To evaluate the performance of the proposed MEC rendering schemes, we focus on the objective of maximizing the Peak Signal-to-Noise Ratio (PSNR) [84], 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 [45], 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 an unpleasant human experience, thus, for the kth VR user, we revise the basic QoE model by incorporating a maximum VR interaction latency requirement T_k^{th}. More specifically, if $T_k \leq T_k^{\mathrm{th}}$, 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

$$\mathrm{MSE_k} = (\mathcal{I}_\mathrm{k} - \mathcal{D}_\mathrm{k})^2. \tag{3.16}$$

According to [45, Eq. (2)], the PSNR of the kth VR user is defined as

$$\mathrm{PSNR_k} = 10\log_{10}\frac{1}{\mathrm{MSE_k}}. \tag{3.17}$$

As can be seen from (3.17), for $\mathrm{MSE_k} = 0$, $\mathrm{PSNR_k} \to \infty$. To avoid the infinite value of PSNR, we introduce a positive number $\triangle$ and modify (3.17) as

$$\mathrm{PSNR_k} = 10\log_{10}\frac{1+\triangle}{\mathrm{MSE_k}+\triangle}, \tag{3.18}$$

where $\triangle > 0$ and we set $\triangle = 1$ in this chapter. Meanwhile, the value of $\triangle$ will not affect the learning performance.

3.2.3 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 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

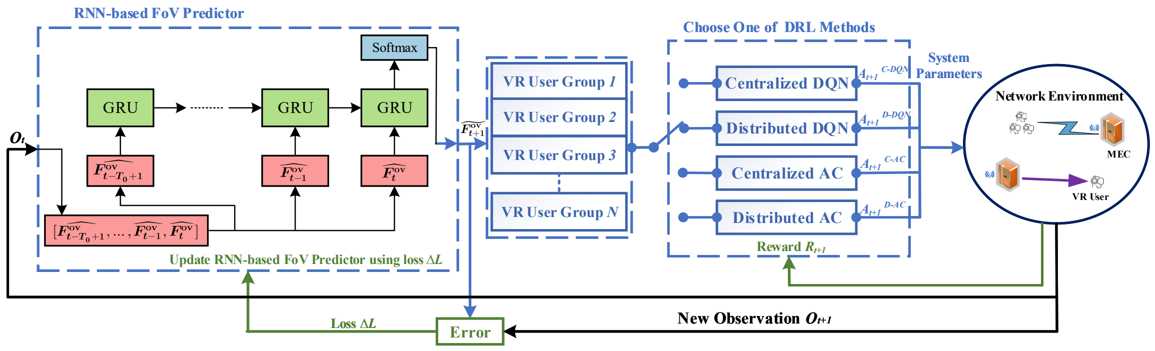


Fig. 3.5 Decoupled learning strategy for MEC rendering schemes in the wireless VR network.

formulated as

$$(\text{P1}) \quad \max_{\pi(\mathrm{A_t}|\mathrm{S_t})} \sum_{i=t}^{\infty} \sum_{k=1}^{K_{\mathrm{VR}}} \gamma^{i-t} \mathbb{E}_{\pi}[\mathrm{PSNR}_k^i] \tag{3.19}$$

$$T_k \leq T_k^{\mathrm{th}}, \tag{3.20}$$

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 about 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. In (3.19), π is the policy deployed to maximize the long-term PSNR of VR users based on the state observed from the network environment.

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 global information to achieve the optimal solution, which not only increases the overhead of signal transmission, but also increase the computation complexity. Approximate solutions will be discussed in Section 2.3.

3.3 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 [78]. In this section, to solve (P1), a decoupled learning strategy is proposed for FoV prediction and MEC rendering association, as shown in Fig. 3.5. Specifically, an RNN model based on GRU is used to predict the 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.

3.3.1 FoV Prediction

Brownian Motion is used to simulate eye movements of VR users over time, and assuming that the uplink received FoV preference of the kth VR user at the tth time slot is $\widehat{F_t^{\mathrm{oV}}}^k \in \{1,2,...,N_{\mathrm{FoV}}\}$. 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 $O_t = \{O_t^1, O_t^2, ..., O_t^{\mathrm{K_{VR}}}\}$, where $O_t^k = \{\widehat{F_t^{\mathrm{oV}}}^k\}$, but also the historical information in the previous observations $H_t = \{O_{t-T_0+1}, ..., O_{t-2}, O_{t-1}\}$ given a memory window T_0. To recognize FoV preference over time, an RNN model with parameters $\boldsymbol{\theta}_{\mathrm{RNN}}$, and specifically a GRU architecture, is leveraged. $\boldsymbol{\theta}_{\mathrm{RNN}}$ consists of both the GRU internal parameters and weights of the softmax layer. We choose RNN due to its ability in capturing time correlation of FoV preferences over time, which can help learn the time-varying FoV preference for better prediction accuracy.

As shown in Fig. 3.5, the GRU layer includes multiple standard GRU units and historical observations $[O_{t-T_0+1}, ..., O_{t-1}]$ are 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 $\mathcal{P}\{\widehat{F_t^{\mathrm{oV}}}^k = \hat{f} | [O_{t-T_0+1}^k, ..., O_{t-1}^k], \boldsymbol{\theta}_{\mathrm{RNN}}\}$ of the $\hat{f}$th FoV ($\hat{f} = 1, ..., N_{\mathrm{FoV}}$) for the tth time slot given historical observations $[O_{t-T_0+1}^k, ..., O_{t-1}^k]$.

To adapt the model parameter $\boldsymbol{\theta}_{\mathrm{RNN}}$, standard Stochastic Gradient Descent (SGD) via BackPropagation Through Time (BPTT) [141] is deployed. At the $(t+1)$th time slot, the parameters $\boldsymbol{\theta}_{\mathrm{RNN}}$ of the RNN predictor can be updated as

$$\boldsymbol{\theta}_{\mathrm{RNN}}^{t+1} = \boldsymbol{\theta}_{\mathrm{RNN}}^{t} - \lambda_{\mathrm{RNN}} \nabla L_{\mathrm{RNN}}(\boldsymbol{\theta}_{\mathrm{RNN}}^{t}), \tag{3.21}$$

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

$$L^t_{\text{RNN}}(\boldsymbol{\theta}_{\text{RNN}}) = -\sum_{t'=t-T_b+1}^{t} \log\left(\mathcal{P}\{\widehat{F^{\text{oV}}_{t'}} = \widetilde{F^{\text{oV}}_{t'}} | O^{t'}_{t'-T_0}, \boldsymbol{\theta}_{\text{RNN}}\}\right), \tag{3.22}$$

where

$$O^{t'}_{t'-T_0} = [O_{t'-T_0+1}, \ldots, O_{t'-1}, O_{t'}], \tag{3.23}$$

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 a unique FoV, respectively.

3.3.2 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 the 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 the problem (P1) are introduced as follows.

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

$$\begin{aligned} S_t &= (\widetilde{\mathcal{F}^{\text{oV}}_t}, \mathcal{L}^t_{k,i}, \mathcal{F}^{\text{MEC}}_i) \in \mathcal{S}, \\ \text{with } \widetilde{\mathcal{F}^{\text{oV}}_t} &= \{\widetilde{F^{\text{oV}}_t}^1, \widetilde{F^{\text{oV}}_t}^2, \ldots, \widetilde{F^{\text{oV}}_t}^{K_{\text{VR}}}\}, \\ \mathcal{L}^t_{k,i} &= \{l^t_{k,1}, l^t_{k,2}, \ldots, l^t_{k,B}, \}, \\ \mathcal{F}^{\text{MEC}}_i &= \{F^{\text{MEC}}_1, F^{\text{MEC}}_2, \ldots, F^{\text{MEC}}_B\}, \end{aligned} \tag{3.24}$$

where $\widetilde{F_t^{\mathrm{oV}}}^k$ 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

$$\begin{aligned} A_t &= \{\check{\mathcal{A}}_{k,q}^t, \acute{\mathcal{A}}_{k,i}^t\} \in \mathcal{A}, \\ \text{with } \check{\mathcal{A}}_{k,q}^t &= \{\check{A}_{k,1}, \check{A}_{k,2}, ..., \check{A}_{k,N_{\mathrm{FoV}}}\}, \\ \acute{\mathcal{A}}_{k,i}^t &= \{\acute{A}_{k,1}, \acute{A}_{k,2}, ..., \acute{A}_{k,K_{\mathrm{VR}}}\}, \end{aligned} \tag{3.25}$$

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

$$R_t(S_t, A_t) = \sum_{k=1}^{K_{\mathrm{VR}}} \mathrm{PSNR}_k^t. \tag{3.26}$$

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

$$V(S, \pi) = \sum_{i=t}^{\infty} (\gamma)^{i-t} R_i(S_i, A_i), \tag{3.27}$$

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

When the number of MECs and VR users is small, the 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 problems. In DRL algorithms, two methods can be used to obtain the optimal policy. One is called value-based optimization, such as DQN, which indirectly optimizes the policy by optimizing the 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.

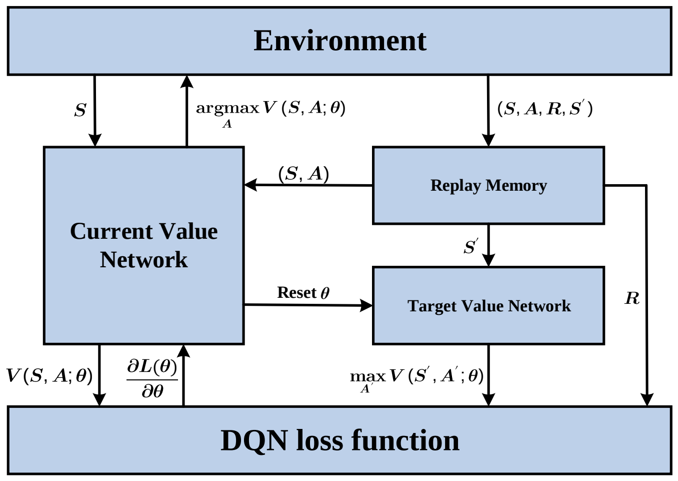


Fig. 3.6 The DQN diagram of the MEC rendering scheme.

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 the DQN algorithm, a fraction of states is 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 broken 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. 3.6, the action-state value function $V_{\mathrm{DQN}}(S,A)$ in the DQN agent can be parameterized by using a function $V_{\mathrm{DQN}}(S,A;\boldsymbol{\theta}_{\mathrm{DQN}})$, where $\boldsymbol{\theta}_{\mathrm{DQN}}$ is the weight matrix of the DNN with multiple layers. In the conventional DNN, 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 $V_{\mathrm{DQN}}(S,A;\boldsymbol{\theta}_{\mathrm{DQN}})$ with respect to the observed state S_t. Moreover, the parameter $\boldsymbol{\theta}_{\mathrm{DQN}}$ can be updated by using SGD as

$$\boldsymbol{\theta}_{\mathrm{DQN}}^{t+1} = \boldsymbol{\theta}_{\mathrm{DQN}}^{t} - \lambda_{\mathrm{DQN}} \nabla L_{\mathrm{DQN}}(\boldsymbol{\theta}_{\mathrm{DQN}}^{t}), \tag{3.28}$$

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

$$L_{\text{DQN}}(\boldsymbol{\theta}^t_{\text{DQN}}) = (\hat{V}_{\text{DQN}} - V_{\text{DQN}}(S_i, A_i; \boldsymbol{\theta}^t_{\text{DQN}}))^2, \tag{3.29}$$

where

$$\hat{V}_{\text{DQN}} = R_{i+1} + \gamma \max_A V_{\text{DQN}}(S_{i+1}, A; \bar{\boldsymbol{\theta}}^t_{\text{DQN}}). \tag{3.30}$$

$(S_i, A_i, S_{i+1}, R_{i+1})$ are randomly selected previous samples for some $i \in \{t - M_r, ..., t\}$ with respect to a so-called minibatch. M_r is the replay memory. $\bar{\boldsymbol{\theta}}^t_{\text{DQN}}$ is the so-called target Q-network which is utilized to estimate the future value of the Q-function in the update rule. Meanwhile, $\bar{\boldsymbol{\theta}}^t_{\text{DQN}}$ is periodically copied from the current value $\boldsymbol{\theta}^t_{\text{DQN}}$ 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 (3.29) 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 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 increases, the size of the proposed model and parameters can expand exponentially. In this case, the GPU memory in the 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 scales 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 increasing 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 [107].

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 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 [60].

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

Algorithm 2 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,...,$M$ do
4:     Input the network state $S$ of the MEC rendering scheme.
5:     for t = 1,...,T do
6:         Use $\varepsilon$-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 (3.28).
18:        Update parameter $\bar{\boldsymbol{\theta}}_{\text{DQN}}$ of the target network every $\bar{K}$ steps.
19:    end for
20: end for
```

$\widetilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_{\text{DDQN}}$. 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 process, the parameter $\widetilde{\boldsymbol{\theta}}_i$ of the ith MEC can be updated as

$$\widetilde{\boldsymbol{\theta}}_i^{t+1} = \widetilde{\boldsymbol{\theta}}_i^t - \lambda_{\text{DDQN}} \nabla L_i(\widetilde{\boldsymbol{\theta}}_i^t), \tag{3.31}$$

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

$$L_i(\widetilde{\boldsymbol{\theta}}_i^t) = (\hat{V}_{\text{DDQN}} - V_{\text{DDQN}}(S_j^i,A_j^i;\widetilde{\boldsymbol{\theta}}_i^t))^2, \tag{3.32}$$

where

$$\hat{V}_{\text{DDQN}} = R_{j+1}^i + \gamma \max_{A^i} V_{\text{DDQN}}(S_{j+1}^i,A^i;\bar{\widetilde{\boldsymbol{\theta}}}_i^t). \tag{3.33}$$

$(S^i_j, A^i_j, S^i_{j+1}, r^i_{j+1})$ are randomly selected previous samples for $j \in \{t - M_r, .., t\}$ of the ith MEC. $\tilde{\tilde{\boldsymbol{\theta}}}^t_i$ 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 (3.32) and computing the expectation of the selected samples, $V^*_{\mathrm{DDQN}}(S^i, A^i)$ 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 $\boldsymbol{\theta}_{\mathrm{DDQN}}$ can be updated as

$$\boldsymbol{\theta}_{\mathrm{DDQN}} = \frac{1}{K^{\mathrm{MEC}}_{\mathrm{DDQN}}} \sum_{i=1}^{K^{\mathrm{MEC}}_{\mathrm{DDQN}}} \widetilde{\boldsymbol{\theta}}_i, \tag{3.34}$$

where $K^{\mathrm{MEC}}_{\mathrm{DDQN}}$ is the number of the MECs associated with the VR user groups.

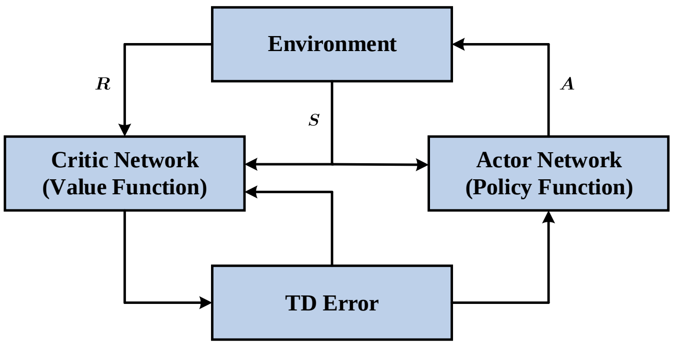

Environment
R
S
A
Critic Network (Value Function)
Actor Network (Policy Function)
TD Error

Fig. 3.7 The Actor-Critic diagram of the MEC rendering scheme.

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 consideration [61, 140]. As shown in Fig. 3.7, the AC network consists 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 the critic updating step and actor updating step, which leads to slow learning

Algorithm 3 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 multicast/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:     Calculate TD error $\delta_t$ according to (3.35).
9:     Update the parameters $\boldsymbol{w}_{\text{AC}}$ of the critic network via (3.36).
10:    Update the parameters $\boldsymbol{\theta}_{\text{AC}}$ of the actor network via (3.37).
11: end for
```

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

$$\delta_t = R_t + \gamma V_{\text{AC}}(S_{t+1};\boldsymbol{w}_{\text{AC}}^t) - V_{\text{AC}}(S_t;\boldsymbol{w}_{\text{AC}}^t). \tag{3.35}$$

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

$$\boldsymbol{w}_{\text{AC}}^{t+1} = \boldsymbol{w}_{\text{AC}}^t + \lambda_{\text{critic}}\delta_t \nabla_{\boldsymbol{w}_{\text{AC}}} V_{\text{AC}}(S_t;\boldsymbol{w}_{\text{AC}}^t). \tag{3.36}$$

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

$$\boldsymbol{\theta}_{\text{AC}}^{t+1} = \boldsymbol{\theta}_{\text{AC}}^t + \lambda_{\text{actor}}\delta_t \nabla_{\boldsymbol{\theta}_{\text{AC}}} \log \pi(A_t|S_t;\boldsymbol{\theta}_{\text{AC}}^t), \tag{3.37}$$

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 the MEC rendering scheme is proposed in Algorithm 3.

Distributed AC

Unlike the centralized AC algorithm, the agent in the distributed AC algorithm performs an 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 [155].

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, 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

$$\bar{\boldsymbol{w}}_{t+1}^i = \bar{\boldsymbol{w}}_t^i + \bar{\lambda}_{\text{critic}} \delta_t^i \nabla_{\bar{\boldsymbol{w}}^i} V_{\text{DAC}}(S_t^i;\bar{\boldsymbol{w}}_t^i), \tag{3.38}$$

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

$$\delta_t^i = R_t^i + \gamma V_{\text{DAC}}(S_{t+1}^i;\bar{\boldsymbol{w}}_t^i) - V_{\text{DAC}}(S_t^i;\bar{\boldsymbol{w}}_t^i). \tag{3.39}$$

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

$$\bar{\boldsymbol{\theta}}_{t+1}^i = \bar{\boldsymbol{\theta}}_t^i + \bar{\lambda}_{\text{actor}} \delta_t^i \nabla_{\bar{\boldsymbol{\theta}}^i} \log \pi(A_t^i|S_t^i;\bar{\boldsymbol{\theta}}_t^i), \tag{3.40}$$

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

$$\boldsymbol{w}_{\text{DAC}} = \frac{1}{K_{\text{DAC}}^{\text{MEC}}} \sum_{i=1}^{K_{\text{DAC}}^{\text{MEC}}} \bar{\boldsymbol{w}}^i, \tag{3.41}$$

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.

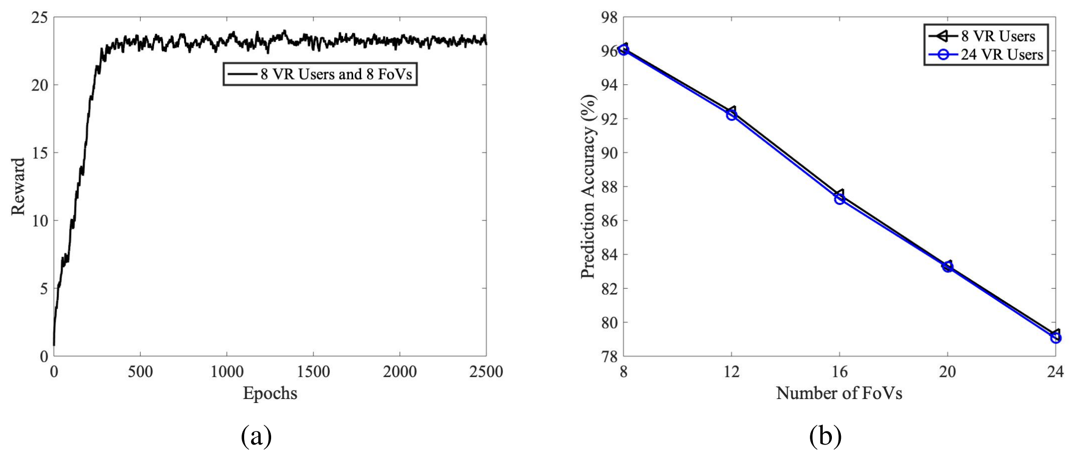


Fig. 3.8 (a) Total reward of FoV prediction of each epoch via GRU with 8 VR users and 8 FoVs. (b) FoV prediction accuracy of GRU with 8 and 24 VR users for varying number of FoV.

3.3.3 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 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 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] [49]. For the implementation of the proposed algorithms, according to Fig. 3.5, 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.

3.4 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 to

[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.

Table 3.1 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_{max}^{MEC}	5 GHz	F_{min}^{MEC}	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

use a fully-connected neural network with two hidden layers and each layer has 128 ReLU units. The simulation parameters are summarized in Table 3.1.

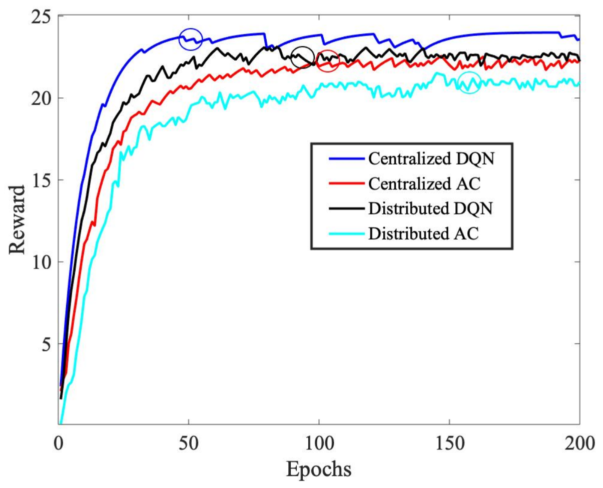


Fig. 3.9 Total reward of the MEC rendering with prediction and migration scheme of each epoch via centralized/distributed DQN/AC learning algorithms.

3.4.1 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 the FoV preference of each VR user in continuous time slots, an RNN model based on GRU architecture is deployed. Fig. 3.8 (a) plots the total reward of FoV prediction of each epoch via GRU with 8 VR users and 8 FoVs, and Fig. 3.8 (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

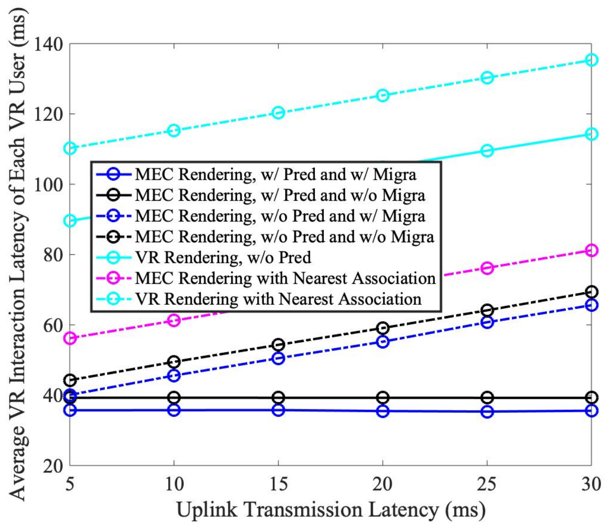


Fig. 3.10 Average VR interaction latency of various MEC and VR rendering schemes via centralized DQN algorithm for varying uplink transmission latency.

increasing number of FoVs. This is because the eye 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 a different number of VR users. This is because RNN utilizes a memory window with a length of 20 for each VR user to store the input observations, which can capture the FoV preference of VR users over time.

3.4.2 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 figures, respectively. To guarantee the fairness of each VR user, we use average QoE and VR interaction latency in the performance results.

Fig. 3.9 plots the total reward of the MEC rendering with the 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 follow: Centralized DQN $>$ Distributed 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 behaviors. As the model parameters in the AC algorithm are updated in two steps, including the critic step and the actor step, the convergence speed of the AC algorithm is lower. Apparently, the convergence

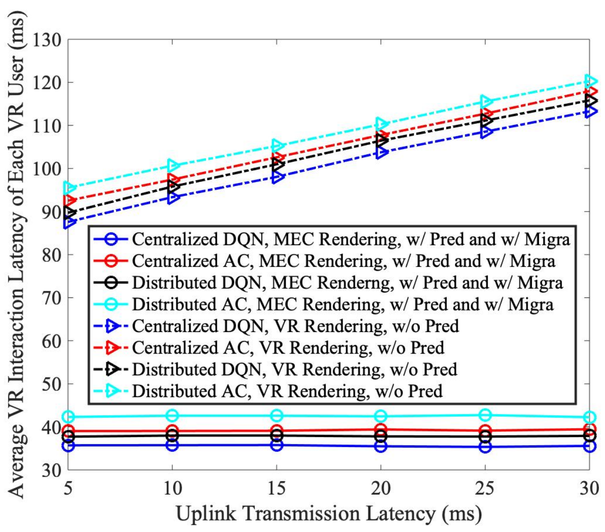


Fig. 3.11 Average VR interaction latency of the MEC rendering with prediction and migration scheme and the VR rendering scheme via centralized/distributed DQN/AC learning algorithms for varying uplink transmission latency.

speed of the centralized learning algorithms is faster than that of the distributed learning algorithms. This is because 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. 3.10 plots the average VR interaction latency of various MEC and VR rendering schemes via a centralized DQN algorithm for varying uplink transmission latency. We observe that all MEC rendering schemes outperform that of the VR device 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 decreases 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 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. 3.10, we also compare our proposed learning-based schemes with those without learning. By comparing the MEC/VR rendering scheme with the nearest association scheme plotted using dash lines, we see our proposed learning-based MEC/VR rendering schemes achieve a 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.

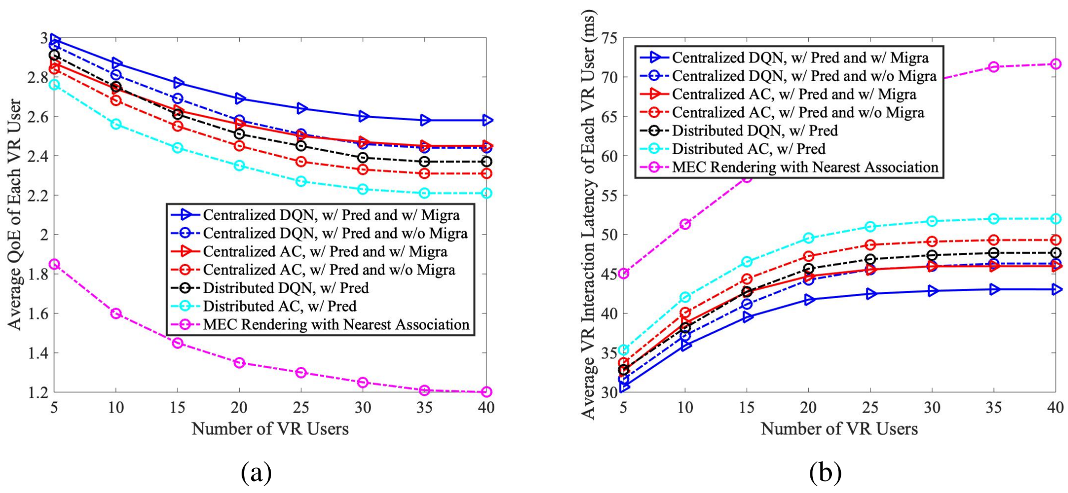
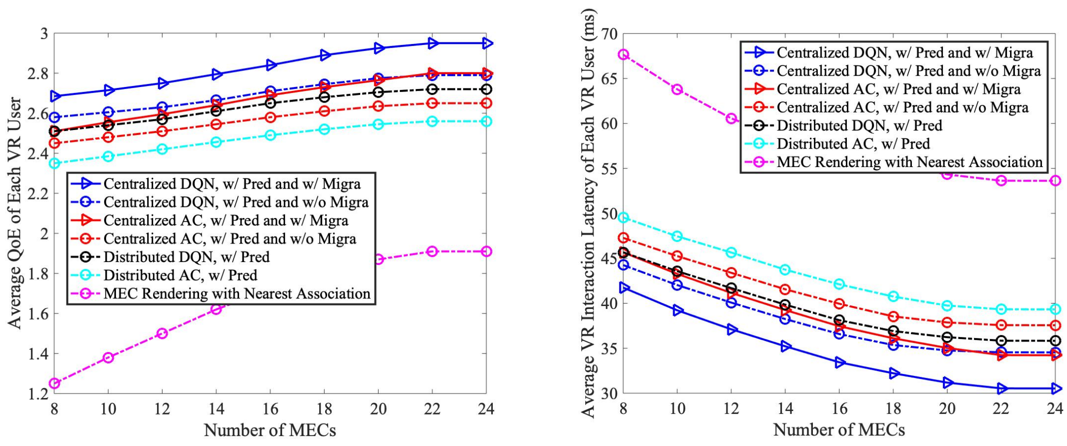


Fig. 3.12 Average QoE and VR interaction latency of MEC rendering with prediction with/without migration schemes via centralized/distributed DQN/AC learning algorithms with increasing number of VR users.

Fig. 3.11 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 the MEC rendering scheme achieves much lower latency (about 40 ms) compared to the 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. 3.12 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 number of VR users, the average QoE of VR users first decreases then becomes nearly stable as shown in Fig. 3.12 (a), whereas the average VR interaction latency first increases, then becomes nearly stable as shown in Fig. 3.12 (b). This is because with increasing number of VR users, more MECs are activated to provide downlink transmission for more requested FoVs, which increase the interference among those transmissions. As the number of VR users becomes too large, all MECs become active to serve all VR users to render most of FoVs, and 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. 3.12 (a) and (b). This is because the MECs with higher computing



(a) (b)

Fig. 3.13 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.

ability will be selected to render the same required FoV for migration, which decreases the rendering latency. Importantly, all learning-based MEC rendering with prediction schemes substantially outperform the conventional non-learning-based MEC rendering with the nearest association scheme.

Fig. 3.13 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 number of MECs, the average QoE of VR users first increases, then becomes nearly stable as shown in Fig. 3.13 (a), whereas the average VR interaction latency first decreases, then becomes nearly stable as shown in Fig. 3.13 (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.

3.5 Conclusions

In this chapter, a decoupled learning strategy was developed to optimize real-time VR video streaming in wireless networks, which considered FoV prediction and rendering MEC association. Specifically, based on GRU architecture, an RNN model was used to predict the 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 show 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 decreased the VR interaction latency.

Chapter 4

Learning-based Prediction, Rendering and Transmission for Interactive VR in RIS-Assisted Terahertz Networks

4.1 Introduction

Wireless virtual reality (VR) can be a potential solution in breaking geographical boundaries, providing the VR users with a sense of total presence and immersion under VR interaction latency, and may unleash plenty of novel VR applications [25]. To achieve this vision, we still face unique challenges, including how to support real-time VR interaction under low interaction latency (in the order of tens of milliseconds), high resolution 360-degree VR video transmission under high data rates, seamless connectivity for moving VR users even under unstable wireless channels, and satisfy the asymmetric and coupled uplink and downlink requirements [69].

4.1.1 Motivation

To address the above challenges, terahertz (THz) communication can be a promising enabler for high rate, high reliability, and low VR interaction latency [16]. However, the THz transmission suffers from severe propagation attenuation and water-molecular absorption loss because of its high frequency, which limits the propagation distance [64]. That is the reason why the THz communication system is usually deployed in an indoor scenario. Note that the indoor environment can be complex with physical obstacles, such as walls and furniture, which may block the line-of-sight (LoS) communication links [58, 108].

To address the severe path attenuation of THz and support transmission for users in non-line-of-sight (NLoS) areas, reconfigurable intelligent surface (RIS) can be an effective approach to create a second virtual LoS path and enhance the coverage [24, 112, 114, 113, 158]. The RIS is a planar surface that consists of a number of small-unit reflectors, and is equipped with a low-cost sensor and controlled with a simple processor. Each reflecting element of the RIS can reflect incident electromagnetic waves independently with an adjustable phase shift. Through deploying the RIS in the THz network and smartly adjusting the phase shift of all elements, the THz signals between the transmitters and receivers can be reconfigured flexibly to support the THz transmission for the users in NLoS areas [148, 71, 137].

It is important to note that the VR interaction latency, which is composed of the uplink transmission latency, the rendering latency, and the downlink transmission latency, is also one of the key requirements in VR service. Violating the VR interaction latency constraint can lead to motion sickness and discomfort [69]. Rendering real-time high-quality VR videos via a computing unit with high processing capabilities can be a potential solution to reduce the VR interaction latency. To do so, mobile edge computing (MEC) can be introduced to shift the heavy VR video computation load from the VR device to the MEC server [101].

4.1.2 Contributions

Motivated by the above, in this chapter, we focus on optimizing the QoE of VR users in a MEC-enabled and RIS-assisted THz VR network in an indoor scenario, and we develop a novel learning strategy to efficiently optimize the long-term QoE in THz VR systems. The main contributions are summarized as follows:

- We propose a MEC-enabled and RIS-assisted THz VR network in an indoor scenario, taking into account the uplink transmission, MEC rendering, and downlink VR video transmission.

- In the uplink, we use a two-ray uplink transmission to deliver the actual viewpoints or learning models to the MEC. Based on the historical and current viewpoints of the VR user from real VR datasets [4], we propose two methods, which are referred to as the centralized online Gated Recurrent Unit (GRU) algorithm and the distributed Federated Averaging (FedAvg) algorithm, to predict the dynamical viewpoint preference of VR users over time. By doing so, the predicted field of view (FoV) can be rendered and transmitted in advance, with the aim to reduce the VR interaction latency.

- We also propose an algorithm that integrates online Long-Short Term Memory (LSTM) and Convolutional Neural Networks (CNN) to predict new positions of VR users based on their historical positions delivered via the uplink, which are then used to predict the LoS or the NLoS statuses of VR users.
- With the predicted viewpoint and the LoS/NLoS status of each VR user as inputs, we propose a constrained Deep Reinforcement Learning (C-DRL) algorithm to optimize the long-term QoE of VR users under VR interaction latency constraints, by selecting the optimal phase shifts of the RIS reflecting elements. Through comparison with non-learning-based methods, we show that our proposed ensemble learning architecture with GRU, LSTM, CNN, and C-DRL achieves near-optimal QoE similar to that offered by an exhaustive-search algorithm, and enhances about two times the QoE compared to the random phase shift selection scheme.

4.1.3 Organization

The rest of this chapter is organized as follows. The system model and problem formulation are proposed in Section 4.2. The learning algorithms for THz VR systems are presented in Section 4.3. The simulation results and conclusions are described in Section 4.4 and Section 4.5, respectively.

4.2 System Model and Problem Formulation

We consider an indoor scenario, where an RIS that comprises N reflecting elements is deployed to assist the uplink and downlink transmission between a MEC and K^{VR} VR users, as shown in Fig. 4.1. The MEC operating over THz frequency is equipped with M antennas[1] and each VR user is equipped with a single antenna, respectively. The indoor scenario is assumed to be a square with length W of each side. The RIS is connected to a smart controller that communicates with the MEC via a wired link for cooperative transmission and information exchange, such as channel state information (CSI), and phase shifts control of all reflecting elements [147]. Due to the substantial path loss in THz transmission, we only consider the THz signal reflected by the RIS for the first time and ignore the signals that are reflected twice or more times following [144].

4.2.1 VR User Mobility

We present a mobility model based on the VR user movements in the indoor VR scenario, which is the so-called virtual reality mobility model (VRMM) [152]. The VRMM includes

[1]Physically, the MEC and SBS are co-located in one location.

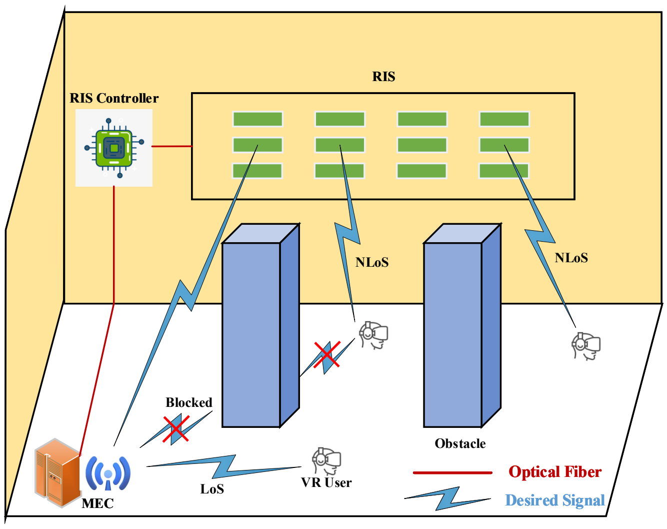


Fig. 4.1 Wireless VR system in THz network.

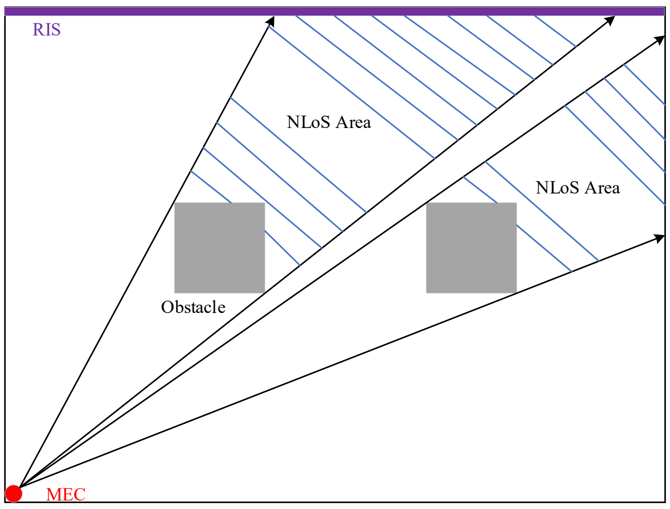


Fig. 4.2 Illustration of THz network in the presence of obstacles.

the following parameters: start location, destination location, speed, and moving direction. We assume that there are four directions for the VR user to select, namely, up, down, left, and right. We split the indoor area into $W \times W$ grids. When the VR user is at the start location, it sets its destination location, speed, and moving direction, and transmits its current location at each time slot to the MEC server through uplink transmission. Note that the location of the VR user for the next time slot is determined by the location of the current time slot rather than the locations in previous time slots, so that the mobility of the VR user in the indoor area follows the Markov property. When the VR user arrives at the destination location, it sets a new destination location and moves forward to it with a given speed.

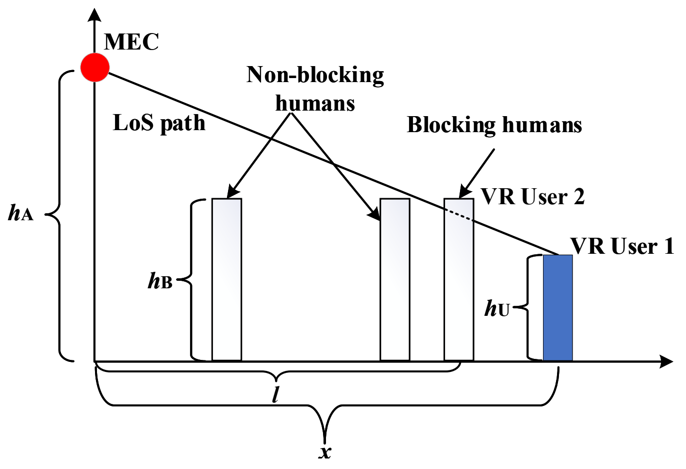


Fig. 4.3 Illustration of a single THz transmission link in the presence of blocker via other VR user with higher height.

4.2.2 Indoor Blockage

Due to the severe signal attenuation and narrow wave spread in THz frequencies, the THz transmission is very sensitive to the presence of obstacles [149]. When the VR users are moving in an indoor scenario, the blockage between the MEC and the kth VR user can be caused by an obstacle and the other VR users with higher heights that are located near the kth VR user. For simplicity, we map the 3D indoor scenario into a 2D image. In Fig. 4.2, when VR users are behind the obstacle, they are directly blocked by the obstacle. As shown in Fig. 4.3, we assume that the height of the MEC is h_A, the height of the VR user 2 is h_B ($h_B < h_A$), the height of the VR user 1 is h_U ($h_U < h_B$), the distance between the VR user 2 with height h_B and the MEC is l, and the distance between the VR user 1 with height h_U and the MEC is x.

Definition 1: When the MEC server, the VR user 2, and the VR user 1 are located in the same line in the 2D plane, the VR user 1 will be blocked by the VR user 2 if their distance in the 2D plane is less than $\frac{(h_A-h_U)l}{h_A-h_B}$.

Proof. According to Fig. 4.3, the coordinates of the MEC, the VR user 2, and the VR user 1 in the 2D plane are denoted as $(0,h_A)$, (l,h_B), and (x,h_U), respectively. Applying slope-intercept equation, we can calculate the line equation across the point $(0,h_A)$ and (l,h_B) as

$$y = \frac{h_B - h_A}{l}x + h_A. \tag{4.1}$$

For $y = h_U$, x is computed as $\frac{(h_A-h_U)l}{h_A-h_B}$. Thus, the distance between the VR user 2 and the VR user 1 is computed as $\frac{(h_B-h_U)l}{h_A-h_B}$. □

Due to the blockage caused by the obstacles, such as pillars, walls, or other VR users, the THz transmission between the MEC server and the VR users can be enhanced by the RIS, where each passive reflecting element can change the phase shift of the THz wave [98]. In our model, we define the MEC-VR user link as the line-of-sight (LoS) link, and the MEC-RIS-VR user link as the non-LoS (NLoS) link. It is important to note that through obtaining the current and historical locations and LoS/NLoS statuses of the VR users, the MEC server can predict the LoS/NLoS statuses of the VR users at each time slot.

4.2.3 THz Uplink Transmission

At the start of each time slot, the VR user transmits its actual viewpoint and location to the MEC via uplink transmission. Because of the mobility of the VR user, it may enter the LoS or NLoS region. To guarantee the reliability of the uplink transmission, we consider a two-ray uplink transmission. One ray is the LoS link, and the other is the NLoS link. For the VR user in the LoS region, the received signals include the ones from LoS and NLoS links. While for the VR user in the NLoS region, the received signal only includes the signal from the NLoS link. For the kth VR user, the transmitted two-ray signals through the uplink transmission at the tth time slot are denoted as

$$\begin{aligned} y_k^{\mathrm{up}}(t) = & \mathbf{u}_k^H(t)\mathbf{h}_k^{\mathrm{up}}(t)x_k^{\mathrm{up}}(t) + \\ & \mathbf{u}_k^H(t)\mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_k^{\mathrm{up}}(t)x_k^{\mathrm{up}}(t) + \\ & \sum_{i=1,i\neq k}^{K^{\mathrm{VR}}} \mathbf{u}_k^H(t)\mathbf{h}_i^{\mathrm{up}}(t)x_i^{\mathrm{up}}(t) + \\ & \sum_{i=1,i\neq k}^{K^{\mathrm{VR}}} \mathbf{u}_k^H(t)\mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_i^{\mathrm{up}}(t)x_i^{\mathrm{up}}(t) + n^{\mathrm{up}}(t), \end{aligned} \tag{4.2}$$

where $\mathbf{u}_k^H(t) \in \mathbb{C}^{1\times M}$ is the beamforming vector of the kth VR user at the tth time slot, which can be denoted as $\frac{\mathbf{h}_k^{\mathrm{up}}(t)+\mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_k^{\mathrm{up}}(t)}{\|\mathbf{h}_k^{\mathrm{up}}(t)+\mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_k^{\mathrm{up}}(t)\|}$ [145]. In (4.2), $\mathbf{h}^{\mathrm{up}}(t) \in \mathbb{C}^{M\times 1}$ is the channel vector between the MEC and the kth VR user at the tth time slot, $x_k^{\mathrm{up}}(t)$ is the transmitted data symbol of the kth VR user, and is set as discrete random variable with zero mean and unit variance, $\mathbf{g}_k^{\mathrm{up}}(t) \in \mathbb{C}^{N\times 1}$ is the channel matrix between the kth VR user and the RIS, $\mathbf{G}^{\mathrm{up}}(t) \in \mathbb{C}^{M\times N}$ is the channel matrix between the RIS and the MEC, and $n^{\mathrm{up}}(t)$ is the additive white Gaussian noise with zero mean and $\hat{\sigma}^2$ variance. Meanwhile, $\sum_{i=1,i\neq k}^{K^{\mathrm{VR}}} \mathbf{u}_k^H(t)\mathbf{h}_i^{\mathrm{up}}(t)x_i^{\mathrm{up}}(t)$ and $\sum_{i=1,i\neq k}^{K^{\mathrm{VR}}} \mathbf{u}_k^H(t)\mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_i^{\mathrm{up}}(t)x_i^{\mathrm{up}}(t)$ are the interferences from the LoS and NLoS links of other VR users, respectively. Let $\boldsymbol{\theta} = [\theta_1, ..., \theta_N]$ denote the selected phase shift set of N reflection elements, where $\theta_n \in [0, 2\pi]$ denotes the phase shift of the nth reflecting element of the RIS, which can be carefully adjusted by an

RIS controller. Here, the reflection coefficients matrix $\mathbf{\Theta}^{\mathrm{up}}(t)$ is presented as

$$\mathbf{\Theta}^{\mathrm{up}}(t) = \mathrm{diag}(e^{j\theta_1^{\mathrm{up}}}(t), ..., e^{j\theta_N^{\mathrm{up}}}(t)). \tag{4.3}$$

For practical implementation, we assume that the phase shift of each element of the RIS can only take a finite number of discrete values. We set b as the number of bits used to indicate the number of phase shift levels $\hat{L}$, where $\hat{L} = 2^b$. For simplicity, we assume that such discrete phase-shift values can be obtained by uniformly quantizing the interval $[0, 2\pi)$. Thus, the set of discrete phase shift values at each element is given by

$$\mathcal{F} = \{0, \triangle\theta, ..., (\hat{L}-1)\triangle\theta\}, \tag{4.4}$$

where $\triangle\theta = 2\pi/\hat{L}$ [144]. Now, the uplink transmission rate of the kth VR user at the tth time slot is calculated as

$$R_k^{\mathrm{up}}(t) = \log_2\left(1 + \frac{|\mathbf{u}_k^H(t)(\mathbf{h}_k^{\mathrm{up}}(t) + \mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_k^{\mathrm{up}}(t))|^2}{I_k^{\mathrm{up}}(t) + \hat{\sigma}^2}\right), \tag{4.5}$$

and

$$I_k^{\mathrm{up}}(t) = \sum_{i=1, i\neq k}^{K^{\mathrm{VR}}} |\mathbf{u}_k^H(t)(\mathbf{h}_i^{\mathrm{up}}(t) + \mathbf{G}^{\mathrm{up}}(t)\mathbf{\Theta}^{\mathrm{up}}(t)\mathbf{g}_i^{\mathrm{up}}(t))|^2. \tag{4.6}$$

According to (4.5), the uplink transmission rate of the VR user in the LoS area is determined by both the LoS and NLoS links. The uplink transmission rate of the kth VR user in the NLoS area is only affected by the NLoS link, and $\mathbf{u}_k^H(t)\mathbf{h}_k^{\mathrm{up}}(t) = 0$.

4.2.4 VR Viewpoint Prediction

By predicting the viewpoint preference of the VR user with a centralized online GRU algorithm or distributed FedAvg algorithm, the corresponding FoV can be rendered and transmitted in advance in order to decrease the VR interaction latency. When the VR user watches the VR video frames, the viewpoint is determined by three degrees of freedom, which are X, Y, and Z axes. Thus, predicting the viewpoint of the VR user can be converted to predicting the X, Y, and Z angles. We consider a sliding window to predict the viewpoint of the VR user in continuous time slots. The future viewpoint of the VR user can be predicted according to the current and past rotation statuses. To guarantee prediction accuracy, we use the online GRU algorithm to predict the viewpoint of the VR user at each time slot. Specifically, we use Mean Square Error (MSE) as a cost function in each training step to update the parameters of the online GRU model, which is calculated as

$$\mathrm{MSE}_t^k = \frac{1}{K^{\mathrm{VR}}} \sum_{k=1}^{K^{\mathrm{VR}}} (\hat{V}_t^k - V_t^k)^2, \tag{4.7}$$

where $\hat{V}_t^k$ can be $\hat{X}_t^k$ or $\hat{Y}_t^k$ or $\hat{Z}_t^k$ and V_t^k can be X_t^k or Y_t^k or Z_t^k are the predicted and actual viewpoints of the kth VR user at the tth time slot, respectively. At the $(t-1)$th time slot, the MEC or the VR device will predict the viewpoint $\hat{V}_t^k$ of the kth VR user for the tth time slot. Then, the kth VR user will transmit its actual viewpoint V_t^k or the learning model to the MEC via uplink transmission. By comparing it with the predicted viewpoint, the parameters of the online GRU algorithm are updated, which can further improve the prediction accuracy.

4.2.5 MEC Rendering

When the VR users enjoy the VR video frames, the corresponding portion of the sphere is rendered at the MEC based on the predicted viewpoint. Through equirectangular projection (ERP) mapping [8], a stitched 2D image with RGB color mode is rendered into the required FoV. We assume that the resolution of the FoV is $N_p \times N_v$, and the size of each pixel is 8 bits. The size of the FoV in RGB mode is calculated as

$$C = 3 \times 8 \times N_p \times N_v \times V, \tag{4.8}$$

where 3 represents the red, green, and blue color in RGB mode, and $V = 2$ is the number of viewpoints for two eyes. We assume that the execution ability of the GPU of the MEC is F_{MEC}, and the number of cycles required for processing one bit of input data in the MEC is f_{MEC}. The MEC rendering latency is calculated as

$$T_{\mathrm{render}} = \frac{f_{\mathrm{MEC}} C}{F_{\mathrm{MEC}}}. \tag{4.9}$$

From (4.9), we can obtain that the rendering latency for all VR users is the same.

4.2.6 THz Downlink Transmission

In the THz downlink transmission, it is possible that VR users may be blocked by the obstacles or VR users with higher heights, as shown in Fig. 4.2 and Fig. 4.3. For the VR users that are not blocked by obstacles and other VR users, the MEC directly performs transmission in the LoS channel, otherwise, it is served by the NLoS channel aided by the RIS [72, 59, 106].

We consider a multi-input single-output (MISO) THz channel. We use the sets $\mathcal{V}_{\text{LoS}}$ and $\mathcal{V}_{\text{NLoS}}$ to denote the LoS and NLoS VR user groups, respectively. For the kth VR user in the LoS group, the received signal from the MEC at the tth time slot is denoted as

$$\begin{aligned} y_k^{\text{LoS}}(t) = \mathbf{h}_k^H(t)\mathbf{v}_k^{\text{LoS}}(t)x_k^{\text{LoS}}(t) + \\ \sum_{i\neq k, i\in\mathcal{V}_{\text{LoS}}} \mathbf{h}_k^H(t)\mathbf{v}_i^{\text{LoS}}(t)x_i^{\text{LoS}}(t) + \\ \sum_{j\in\mathcal{V}_{\text{NLoS}}} \mathbf{h}_k^H(t)\mathbf{v}_j^{\text{NLoS}}(t)x_j^{\text{NLoS}}(t) + n(t), \end{aligned} \tag{4.10}$$

where $\mathbf{h}_k(t) \in \mathbb{C}^{M\times 1}$ is the channel vector between the MEC and the kth VR user, $\mathbf{v}_k^{\text{LoS}}(t) \in \mathbb{C}^{M\times 1}$ and $\mathbf{v}_j^{\text{NLoS}}(t) \in \mathbb{C}^{M\times 1}$ are the beamforming vectors of the kth VR user in the LoS group, and the jth VR user in the NLoS group, respectively. In (4.10), $\mathbf{v}_k^{\text{LoS}}(t)$ can be denoted as $\frac{\mathbf{h}_k(t)}{\|\mathbf{h}_k(t)\|}$, and $x_k^{\text{LoS}}(t)$ and $x_j^{\text{NLoS}}(t)$ indicate the transmitted data symbol for the kth VR user in the LoS group and the jth VR user in the NLoS group, respectively, and are defined as discrete random variable with zero mean and unit variance. We assume that $x_k^{\text{LoS}}(t)$ and $x_j^{\text{NLoS}}(t)$ are independent from each other. Meanwhile, $\sum_{i\neq k, i\in\mathcal{V}_{\text{LoS}}} \mathbf{h}_k^H(t)\mathbf{v}_i^{\text{LoS}}(t)x_i^{\text{LoS}}(t)$ and $\sum_{j\in\mathcal{V}_{\text{NLoS}}} \mathbf{h}_k^H(t)\mathbf{v}_j^{\text{NLoS}}(t)x_j^{\text{NLoS}}(t)$ are the interferences from the MEC. In addition, $n(t) \sim \mathcal{CN}(0,\sigma^2)$ is the additive white Gaussian noise at the kth VR user in the LoS group. The transmission rate between the MEC and the kth VR user in the LoS group at the tth time slot is expressed as

$$R_k^{\text{LoS}}(t) = \log_2\left(1 + \frac{|\mathbf{h}_k^H(t)\mathbf{v}_k^{\text{LoS}}(t)|^2}{I_k^{\text{LoS}}(t) + \sigma^2}\right), \tag{4.11}$$

where

$$I_k^{\text{LoS}}(t) = \sum_{i\in\mathcal{V}_{\text{LoS}}, i\neq k} |\mathbf{h}_k^H(t)\mathbf{v}_i^{\text{LoS}}(t)|^2 + \sum_{j\in\mathcal{V}_{\text{NLoS}}} |\mathbf{h}_k^H(t)\mathbf{v}_j^{\text{NLoS}}(t)|^2. \tag{4.12}$$

For the VR users in the NLoS group, the signal between the MEC and the bth VR user at the tth time slot is presented as

$$\begin{aligned} y_b^{\text{NLoS}}(t) = \mathbf{g}_b^H(t)\mathbf{\Theta}^{\text{down}}(t)\mathbf{G}^{\text{down}}(t)\mathbf{v}_b^{\text{NLoS}}(t)x_b^{\text{NLoS}}(t) + \\ \sum_{j\neq b, j\in\mathcal{V}_{\text{NLoS}}} \mathbf{g}_b^H(t)\mathbf{\Theta}^{\text{down}}(t)\mathbf{G}^{\text{down}}(t)\mathbf{v}_j^{\text{NLoS}}(t)x_j^{\text{NLoS}}(t) + n(t), \end{aligned} \tag{4.13}$$

where $\mathbf{G}^{\text{down}}(t) \in \mathbb{C}^{N\times M}$ is the channel matrix between the MEC and the RIS, $\mathbf{g}_b(t) \in \mathbb{C}^{N\times 1}$ is the channel matrix between the RIS and the bth VR user, $\mathbf{v}_b^{\text{NLoS}}(t) \in \mathbb{C}^{M\times 1}$ is the precoding matrix for the bth VR user in the NLoS group, which can be written as $\frac{\mathbf{G}^{\text{down}}(t)^H\mathbf{\Theta}^{\text{down}}(t)\mathbf{g}_k(t)}{\|\mathbf{G}^{\text{down}}(t)^H\mathbf{\Theta}^{\text{down}}(t)\mathbf{g}_k(t)\|}$, $x_b^{\text{NLoS}}(t)$ is the transmitted data for the bth VR user, $\mathbf{\Theta}^{\text{down}}(t)$ is the

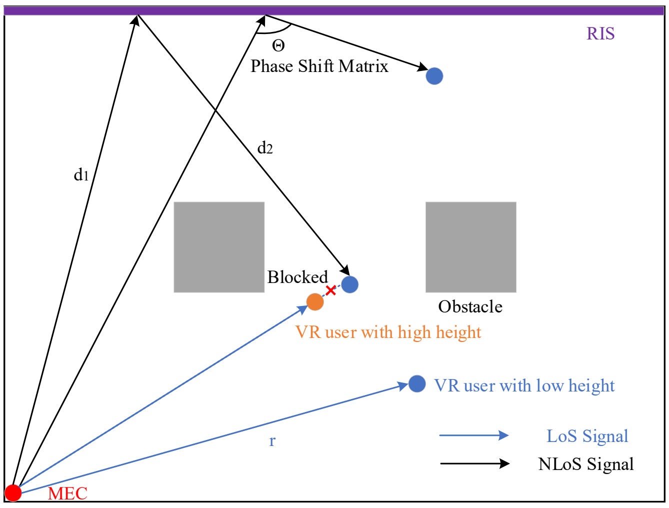


Fig. 4.4 Illustration of THz channel model in the presence of obstacles.

reflection coefficients matrix of the RIS. Note that $\mathbf{\Theta}^{\text{down}}(t)$ is written as

$$\mathbf{\Theta}^{\text{down}}(t) = \text{diag}(e^{j\theta_1^{\text{down}}}(t), ..., e^{j\theta_N^{\text{down}}}(t)). \tag{4.14}$$

In (4.13), $\sum_{j\neq b, j\in\mathcal{V}_{\text{NLoS}}} \mathbf{g}_b^H(t)\mathbf{\Theta}^{\text{down}}(t)\mathbf{G}^{\text{down}}(t)\mathbf{v}_j^{\text{NLoS}}(t)x_j^{\text{NLoS}}(t)$ is the interference from the RIS. Then, the downlink transmission rate of the bth VR user in the NLoS group is written as

$$R_b^{\text{NLoS}}(t) = \log_2\left(1 + \frac{|\mathbf{g}_b^H(t)\mathbf{\Theta}^{\text{down}}(t)\mathbf{G}^{\text{down}}(t)\mathbf{v}_b^{\text{NLoS}}(t)|^2}{I_b^{\text{NLoS}}(t) + \sigma^2}\right), \tag{4.15}$$

where

$$I_b^{\text{NLoS}}(t) = \sum_{j\neq b, j\in\mathcal{V}_{\text{NLoS}}} |\mathbf{g}_b^H(t)\mathbf{\Theta}^{\text{down}}(t)\mathbf{G}^{\text{down}}(t)\mathbf{v}_j^{\text{NLoS}}(t)|^2. \tag{4.16}$$

4.2.7 THz Channel Model

The THz channel model in the presence of obstacles is shown in Fig. 4.4. In THz communication, the power of the scattering component is generally much lower than that of LoS component, since the signal power of THz wave becomes very weak when it is reflected or scattered two or more times. Thus, we ignore the scattering component, and the LoS channel is expressed as

$$\tilde{\mathbf{h}}_k(t) = h_{f,d_k}^{\text{LoS}}(t)\mathbf{a}_{k,\phi_k}^{\text{LoS}}(t), \tag{4.17}$$

where $\tilde{\mathbf{h}}_k(t) = \{\mathbf{h}_k^{\text{up}}(t), \mathbf{h}_k(t)\}$, LoS channel function $h_{\text{LoS}}(f, d_k)$ consists of a spreading loss function and a molecular absorption loss function, which is presented as

$$h_{f,d_k}^{\text{LoS}}(t) = \frac{c}{4\pi f d_k} e^{-\frac{\tau(f) d_k}{2}} e^{-j2\pi f \delta_{\text{LoS,k}}(t)}, \tag{4.18}$$

where c is the speed of light. Assuming that the RIS can be installed on the wall or ceiling of the indoor scenario with height H, the location of the reflecting unit can be presented as $L_{\text{RIS}} = [X_{\text{RIS}}, Y_{\text{RIS}}, H_{\text{RIS}}]$. The location of the MEC can be denoted as $L_{\text{MEC}} = [X_{\text{MEC}}, Y_{\text{MEC}}, H_{\text{MEC}}]$. The location of the kth VR user can be written as $L_k = [X_k, Y_k, H_k]$. The distance between the MEC and the kth VR user is denoted as d_k, which is calculated as

$$d_k = \sqrt{(X_{\text{MEC}} - X_k)^2 + (Y_{\text{MEC}} - Y_k)^2 + (H_{\text{MEC}} - H_k)^2}, \tag{4.19}$$

f is the carrier frequency, and $\delta_{\text{LoS,k}}(t) = \frac{d_k}{c}$ is the time-of-arrival of the LoS propagation of the kth VR user. $\tau(f)$ is the frequency-dependent medium absorption coefficient that depends on the molecular composition of the transmission medium, namely, the type and concentration of molecules found in the channel as defined in [63]. In addition, $\mathbf{a}_{k,\phi_k}^{\text{LoS}}(t)$ is the normalized antenna array response vector at the MEC with M antenna elements, which is written as

$$\mathbf{a}_{k,\phi_k}^{\text{LoS}}(t) = \frac{1}{\sqrt{M}} [1, e^{j\frac{2\pi}{\lambda}\sin(\phi_k)}, \ldots, e^{j\frac{2\pi}{\lambda}(M-1)\sin(\phi_k)}]^H, \tag{4.20}$$

where M is the number of antennas equipped in the MEC, λ is the wavelength, and ϕ_k denotes the angles of departure/arrival (AoD/AoA).

For the NLoS transmission, the THz channels between the MEC and the RIS are denoted as

$$\mathbf{G}^{\text{up}}(t) = \eta G_{f,d_{\text{M-I}}}^{\text{NLoS}}(t) \mathbf{a}_{\phi_{\text{MEC}}}^{\text{NLoS}}(t) \mathbf{a}_{\phi_{\text{RIS}}}^{\text{NLoS}}(t)^H, \tag{4.21}$$

and

$$\mathbf{G}^{\text{down}}(t) = \eta G_{f,d_{\text{M-I}}}^{\text{NLoS}}(t) \mathbf{a}_{\phi_{\text{RIS}}}^{\text{NLoS}}(t) \mathbf{a}_{\phi_{\text{MEC}}}^{\text{NLoS}}(t)^H, \tag{4.22}$$

where η is the path-loss compensation factor written as

$$\eta = \frac{2\sqrt{\pi} f G_{\text{RIS}} N}{c}. \tag{4.23}$$

In (4.23), N is the number of elements on the RIS, and G_{RIS} is the RIS element gain. The channel function $G_{f,d_{\text{M-I}}}^{\text{NLoS}}(t)$ is written as

$$G_{f,d_{\text{M-I}}}^{\text{NLoS}}(t) = \frac{c}{4\pi f d_{\text{M-I}}} e^{-\frac{\tau(f) d_{\text{M-I}}}{2}} e^{-j2\pi f \delta_{\text{NLoS,M-I}}(t)}, \tag{4.24}$$

where $d_{\text{M}-\text{I}}$ is the distance between the MEC and the RIS, $\delta_{\text{NLoS,M-I}}(t) = \frac{d_{\text{M}-\text{I}}}{c}$ is the time-of-arrival of the NLoS propagation between the MEC and the RIS. The normalized antenna array response vectors $\mathbf{a}_{\phi_{\text{RIS}}}^{\text{NLoS}}(t)$ of the RIS and $\mathbf{a}_{\phi_{\text{MEC}}}^{\text{NLoS}}(t)$ of the MEC are written as

$$\mathbf{a}_{\phi_{\text{RIS}}}^{\text{NLoS}}(t) = \frac{1}{\sqrt{N}}[1, e^{j\frac{2\pi}{\lambda}\sin(\phi_{\text{RIS}})}, ..., e^{j\frac{2\pi}{\lambda}(N-1)\sin(\phi_{\text{RIS}})}]^H, \tag{4.25}$$

and

$$\mathbf{a}_{k,\phi_{\text{MEC}}}^{\text{NLoS}}(t)=\frac{1}{\sqrt{M}}[1, e^{j\frac{2\pi}{\lambda}\sin(\phi_{\text{MEC}})}, ..., e^{j\frac{2\pi}{\lambda}(M-1)\sin(\phi_{\text{MEC}})}]^H, \tag{4.26}$$

respectively. In (4.25) and (4.26), ϕ_{RIS} and ϕ_{MEC} are AoD or AoA, repectively. The THz channel between the RIS and the bth VR user is given by

$$\tilde{\mathbf{g}}_b(t) = g_{f,d_b}^{\text{NLoS}}(t)\mathbf{a}_{\phi_b}^{\text{NLoS}}(t), \tag{4.27}$$

where $\tilde{\mathbf{g}}_b(t) = \{\mathbf{g}_b^{\text{up}}(t), \mathbf{g}_b(t)\}$, the channel function $g_{f,d_b}^{\text{NLoS}}(t)$ is written as

$$g_{f,d_b}^{\text{NLoS}}(t) = \frac{c}{4\pi f d_b} e^{-\frac{\tau(f)d_b}{2}} e^{-j2\pi f \delta_{\text{NLoS},b}(t)}, \tag{4.28}$$

d_b is the distance between the RIS and the bth VR user, and $\mathbf{a}_{\phi_b}^{\text{NLoS}}(t)$ is written as

$$\mathbf{a}_{\phi_b}^{\text{NLoS}}(t) = \frac{1}{\sqrt{N}}[1, e^{j\frac{2\pi}{\lambda}\sin(\phi_b)}, ..., e^{j\frac{2\pi}{\lambda}(N-1)\sin(\phi_b)}]^H. \tag{4.29}$$

4.2.8 Quality of Experience Model

The QoE of the wireless VR video frame streaming can be affected by several factors, including video quality, VR interaction latency, and smoothness of the VR video frame [9]. The success of the uplink transmission will further affect the prediction of the viewpoint and the status of LoS or NLoS of the VR user. We use the unit-impulse function $\hat{\delta}_k(t)$ to denote the success of the viewpoint prediction, which is expressed as

$$\hat{\delta}_k(t) = \begin{cases} 1, & \text{if } \hat{V}_t^k = V_t^k; \\ 0, & \text{otherwise.} \end{cases} \tag{4.30}$$

where $\hat{V}_t^k = (\hat{X}_t^k, \hat{Y}_t^k, \hat{Z}_t^k)$ and $V_t^k = (X_t^k, Y_t^k, Z_t^k)$ are the predicted and actual viewpoint of the kth VR user at the tth time slot, respectively. In (4.30), if $\hat{V}_t^k = V_t^k$, $\hat{\delta}_k(t) = 1$, otherwise, $\hat{\delta}_k(t) = 0$. According to [150] and [100], the QoE of the kth VR user at the tth time slot is denoted as

$$\text{QoE}_k(t) = \hat{\delta}_k(t)(q(R_k(t)) - |q(R_k(t)) - q(R_k(t-1))|), \tag{4.31}$$

where $q(R_k(t))$ is the VR video transmission quality metrics. Here, due to [100], $q(R_k(t))$ is presented as

$$q(R_k(t)) = \log\left(\frac{R_k^{\text{down}}(t)}{R_{\text{th}}^{\text{down}}}\right), \tag{4.32}$$

where $R_{\text{th}}^{\text{down}}$ is the downlink transmission threshold, and $|q(R_k(t)) - q(R_k(t-1))|$ is the transmission quality variation, which indicates the magnitude of the changes in the transmission quality from the $(t-1)$th time slot to the tth time slot. Note that the QoE model in (4.31) guarantees the seamless, continuous, smoothness and uninterrupted experience of the VR user.

4.2.9 Optimization Problem

To ensure that the requested FoV is rendered and transmitted within the VR interaction latency, we aim to maximize the long-term QoE of the RIS-aided THz transmission system by optimizing the phase shift of the RIS reflecting element under VR interaction latency constraint. At the tth time slot, the VR interaction latency T_{VR} consists of T_{uplink}, T_{render}, and T_{downlink} [69], which is written as

$$T_{\text{VR}}(t) = T_{\text{uplink}}(t) + T_{\text{render}}(t) + T_{\text{downlink}}(t), \tag{4.33}$$

where $T_{\text{uplink}}(t)$ is the uplink transmission latency, and $T_{\text{render}}(t)$ is the MEC rendering latency. For the centralized online GRU, the size of the uplink data is small, and the uplink transmission latency is negligible. It is important to know that the size of FoV does not change for different viewpoints, thus, the rendering latency remains the same. Therefore, the VR interaction constraint condition can be converted to downlink transmission latency constraint. The proposed THz VR system aims at maximizing the long-term total QoE under the downlink transmission latency constraint in 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. We formulate the optimization as

$$\max_{\pi(A_t|S_t)} \sum_{i=t}^{\infty}\sum_{k=1}^{K} \gamma^{i-t}\text{QoE}_k(i), \tag{4.34}$$

$$s.t.\ T_{\text{downlink}}^{k}(i) \le T_{\text{th}}^{\text{downlink}}, \tag{4.35}$$

where $\gamma \in [0,1)$ is the discount factor which determines the weight of the future QoE, and $\gamma = 0$ means that the agent only considers the immediate reward. In (4.35), $T_{\text{downlink}}^{k}(t)$ is the downlink transmission latency of the kth VR user at the tth time slot, and $T_{\text{th}}^{\text{downlink}}$ is the downlink transmission latency constraint. Note that (4.35) guarantees the VR interaction latency in each time slot under the VR interaction latency constraint.

Due to the fact that the mobility of the VR user is Markovian in continuous time slots, the dynamics of the THz VR system is a Partially Observable Markov Decision Process (POMDP) problem, which is generally intractable. Here, the partial observation refers to that the MEC server can only know the previous viewpoints and locations of the VR users in the environment, while it is unable to know all the information of the environment, including, but not limited to, the channel conditions, and the viewpoint in the current time slot. Meanwhile, the selected policy also needs to satisfy the VR interaction threshold constraint. Thus, the problem in (4.34) is a constrained MDP (C-MDP) problem that can be transformed into the following form

$$\min_{\omega \geq 0} \max_{\pi} \sum_{i=t}^{\infty} \sum_{k=1}^{K} \gamma^{i-t} \mathrm{QoE}_k(i) - \omega (T_{\mathrm{downlink}}^{k}(i) - T_{\mathrm{th}}^{\mathrm{downlink}}), \tag{4.36}$$

where ω is the Lagrangian multiplier, and π is the policy. Due to the fact that the number of combinations of the phase shift increases exponentially with the number of phase shift levels of the RIS, the problem in (4.36) is the generalization of large dimension C-MDP. To address this issue, we deploy constrained deep reinforcement learning (C-DRL) to solve this problem in (4.36) in Section 4.3.

4.3 Learning Algorithms for THz VR System

The deep neural network is one of the most popular non-linear approximation functions, and C-DRL can effectively solve the C-MDP problem [89]. To solve the optimization problem in (4.36), we propose a novel learning architecture based on online GRU, online LSTM, CNN, and C-DRL, as shown in Fig. 4.5. In particular, the online GRU and online LSTM are integrated with CNN to predict the viewpoint preference and LoS or NLoS status of each VR user in continuous time slots, respectively. Using this information as inputs, the C-DRL is deployed to select an optimal reflection coefficient matrix for THz downlink transmission.

4.3.1 Viewpoint Prediction

We use the centralized online GRU and distributed FedAvg to predict the viewpoints of VR users over time. The input of the learning model is the actual viewpoints of previous time slots, and the output is the predicted viewpoint of the VR user for the next time slot. For the centralized online GRU, the VR user directly transmits its actual viewpoint to the

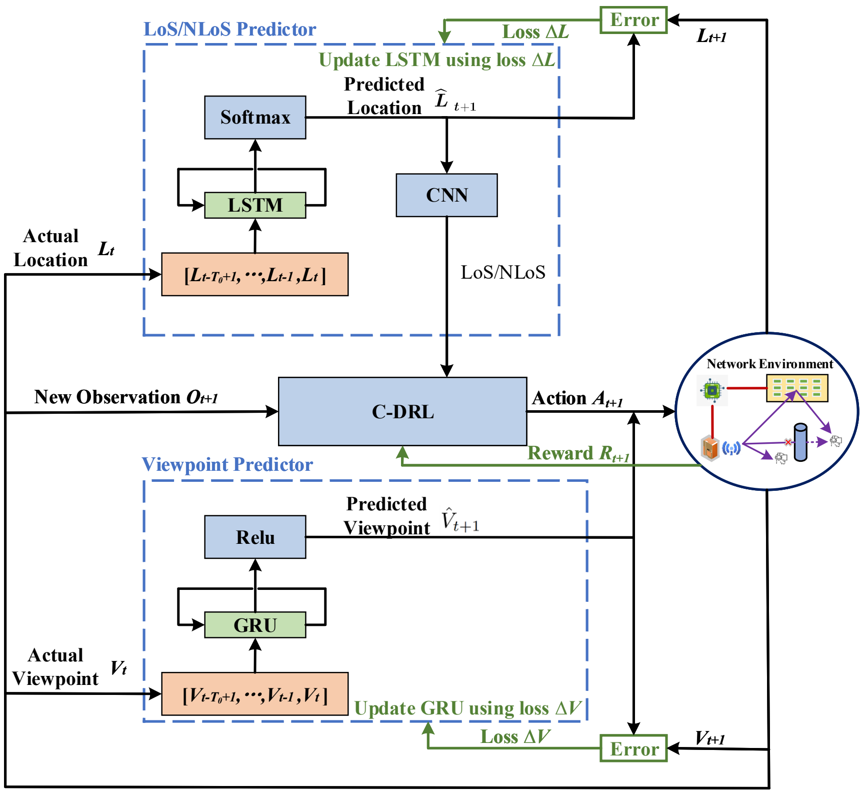


Fig. 4.5 Learning strategy for MEC-enabled and RIS-assisted THz VR networks.

MEC through uplink transmission at each time slot, and then the viewpoint is predicted based on the current and previous time slots. The centralized online GRU for viewpoint prediction has already been introduced in Section IV of [93] in detail.

For the federated learning among distributed VR devices, each VR user predicts the viewpoint in its VR device in continuous time slots based on the online GRU algorithm. At each time slot, the updated learning model of each VR user is delivered to the MEC for model aggregation via uplink transmission, and the model aggregation at the MEC at the tth time slot can be denoted as

$$\bar{\boldsymbol{\theta}}_t^{\mathrm{GRU}} = \sum_{k=1}^{K} p_k \boldsymbol{\theta}_{k,t}^{\mathrm{GRU}}, \text{ and } p_k = \frac{n_k}{n}, \tag{4.37}$$

where p_k is the percentage of the number of data samples of the kth device in the total number of data samples, n_k is the number of data samples of the kth VR user, and n is the total number of data samples, which is calculated as $n = \sum_{k=1}^{K} n_k$. Then, the aggregated model is transmitted to each VR user through downlink transmission to predict the viewpoint, and the predicted viewpoint is delivered to the MEC for VR video frame rendering. Since the size of the learning model is much larger than that of the viewpoint, the VR interaction latency may be increased.

4.3.2 LoS and NLoS Prediction

When VR users move in the indoor scenario following the VRMM mobility model, they may be blocked by the VR users with higher heights or obstacles. To predict the LoS or NLoS status of each VR user in continuous time slots, we first employ an RNN model based on LSTM to predict the position of the VR user [90]. Then, we map the indoor scenario into a 2D image, label the positions of the MEC, the VR users, and the obstacles with different colors, and deploy the CNN to predict the LoS or NLoS status of each VR user.

Long-short Term Memory

To capture the dynamics in mobility of the VR user for the $(t+1)$th time slot, both of the most recent observation $O_t = \{O_t^1, O_t^2, ..., O_t^K\}$ and the previous observations $H_t = \{O_{t-T_o+1}, ..., O_{t-2}, O_{t-1}\}$ given a memory window T_o are required, where $O_t^k = L_t^k = [X_t^k, Y_t^k, H_t^k]$ is the actual location of the kth VR user at the tth time slot. In the VRMM, we assume that there are four moving directions, which are up, down, left and right, and can be denoted as D_{u}, D_{d}, D_{l}, and D_{r}, respectively. To detect the the moving direction of the VR user over time, an RNN model with parameters $\boldsymbol{\theta}^{\mathrm{RNN}}$, and a LSTM architecture in particular, is leveraged, where $\boldsymbol{\theta}^{\mathrm{RNN}}$ consists of both the LSTM internal parameters and weights of each layer.

The online LSTM layer has multiple standard LSTM units and receives current and previous observations at each time slot via the two-ray uplink transmission, and is connected to an output layer, which consists of a Softmax non-linearity activation function with four output values. The four output values represent the predicted probabilities $\mathcal{P}\{D_t = \{D_{\mathrm{u}}, D_{\mathrm{d}}, D_{\mathrm{l}}, D_{\mathrm{r}}\} | [O_t^1, O_t^2, ..., O_t^K], \boldsymbol{\theta}^{\mathrm{RNN}}\}$ of the moving directions for the tth time slot with historical observations $[O_t^1, O_t^2, ..., O_t^K]$.

To update the model parameter $\boldsymbol{\theta}^{\mathrm{RNN}}$, a standard Stochastic Gradient Descent (SGD) [117] via BackPropagation Through Time (BPTT) [141] is used. At the $(t+1)$th time slot, the parameters $\boldsymbol{\theta}^{\mathrm{RNN}}$ are updated as

$$\boldsymbol{\theta}_{t+1}^{\mathrm{RNN}} = \boldsymbol{\theta}_t^{\mathrm{RNN}} - \lambda^{\mathrm{RNN}} \nabla \mathcal{L}^{\mathrm{RNN}}(\boldsymbol{\theta}_t^{\mathrm{RNN}}), \tag{4.38}$$

where $\lambda^{\mathrm{RNN}} \in (0,1]$ is the learning rate, $\nabla \mathcal{L}^{\mathrm{RNN}}(\boldsymbol{\theta}_t^{\mathrm{RNN}})$ is the gradient of the loss function $\mathcal{L}^{\mathrm{RNN}}(\boldsymbol{\theta}_t^{\mathrm{RNN}})$ to train the RNN predictor. Here, $\mathcal{L}^{\mathrm{RNN}}(\boldsymbol{\theta}_t^{\mathrm{RNN}})$ is obtained by averaging the cross-entropy loss as

$$\mathcal{L}_t^{\mathrm{RNN}}(\boldsymbol{\theta}^{\mathrm{RNN}}) = -\sum_{t'=t-T_b+1}^{t} \log\left(\mathcal{P}\{D_{t'} = \hat{D} | O_{t'-T_0}^{t'}, \boldsymbol{\theta}^{\mathrm{RNN}}\}\right), \tag{4.39}$$

where

$$\hat{D}=\{D_{\mathrm{u}},D_{\mathrm{d}},D_{\mathrm{l}},D_{\mathrm{r}}\},\ O_{t'-T_0}^{t'}=[O_{t'-T_0+1},\dots,O_{t'-1},O_{t'}], \tag{4.40}$$

and T_b is the randomly selected mini-batch size.

By predicting the moving direction of each VR user, the MEC is able to know the position of the VR user in the next time slot in advance.

Convolutional Neural Network

Based on the predicted positions of VR users, the MEC, obstacles, and the VR users with different heights are labeled with different colors, and mapped into a 2D image to be the input of the CNN. The CNN predicts the LoS/NLoS status of each VR user based on the input 2D image.

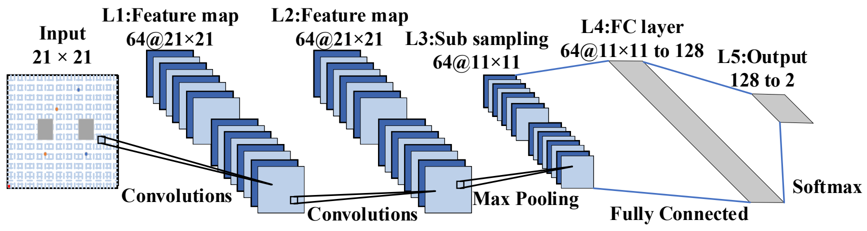


Fig. 4.6 Proposed CNN to classify the LoS or NLoS status of VR users.

The CNN is a multi-layer network evolved from the traditional neural network. The CNN mainly includes the input layer, convolution layer, pooling layer, fully-connected layer, and output layer. It is used for feature extraction and mapping through fast training, and possesses high classification and prediction accuracy. We assume that the proposed CNN model consists of one data input layer, N_c convolution layers, N_p pooling layers, N_f fully-connected layers, and one output layer. Meanwhile, it is assumed that the size of the input image is $N_0 \times N_0$. The detailed description of each layer in the CNN is introduced as follows:

(a) Data Input Layer: The MEC, obstacles, VR users with higher heights, and VR users with lower heights are denoted by different colors in the 2D image. The preprocessed images are used as the input data of the convolution network, and the initial feature extraction is obtained via principal component analysis (PCA) [134], which is usually used for feature extraction. Meanwhile, the images are projected into the characteristic subspace of $N_0 \times N_0 \times 3$, where 3 presents the color of the image is in RGB mode. Therefore, the learning efficiency and computational complexity of the CNN can be decreased by reducing the image dimensionality.

(b) Convolution Layer: The characteristics of the input image are extracted by a randomly initialized filter. It is possible that the input image has various characteristics, thus, multiple filters are used to extract all features in the original image. Zero padding is also used for each convolution layer to keep the size of the features extracted from the input image as $N_0 \times N_0$.

(c) Pooling Layer: It plays an important role in sub-sampling via using the features extracted from the convolution layers. The time complexity can be decreased in the next convolution layer or fully-connected layer by reducing the number of operations in the sub-sampling. The Max-pooling method is usually deployed to extract the largest value in the sliding window for the sub-sampling among all methods used in the pooling layer.

(d) Fully-connected Layer: The features extracted by the convolution and pooling layer are inserted into the neural network. A softmax layer that is often used for the classifications of multiple classes is employed at the end of the fully-connected layer. Meanwhile, the classification result corresponds to a probability that the sum of the probabilities of all classes is equal to 1, and the class with the highest probability is the estimated label for the corresponding input image.

For example, the proposed CNN to classify the LoS or NLoS statuses of VR users is shown in Fig. 4.6, which is also used for the simulations in Section 4.4. In Fig. 4.6, a 21×21 image goes through two convolution layers with 64 filters, and one max-pooling layer. The kernel size and pooling size are 2×2. Then, the extracted features pass through a fully-connected layer with 128 filters. The activation function in the convolution layer and fully-connected layer is the ReLu function. After passing through the fully-connected layer, the softmax layer is used to determine the LoS or NLoS status corresponding to the input image. Note that Adam Optimizer is also employed while training the proposed CNN model [83].

4.3.3 Downlink RIS Configuration

The main purpose of Reinforcement Learning (RL) is to select the proper reflection coefficient matrix $\mathbf{\Theta}$ given in (4.14) of the RIS for THz downlink transmission for the VR users in NLoS areas. While for the uplink transmission at the $(t+1)$th time slot, it directly uses the selected $\mathbf{\Theta}$ at the tth time slot. This is because the downlink transmission requires a high data rate for the FoV with high resolution, whereas the uplink transmission only transmits the actual position and viewpoint or the learning model of the VR user (e.g., the size of the uplink data is much smaller than that of the FoV). Through a series of action strategies, the MEC is able to transmit the selected $\mathbf{\Theta}$ to the RIS via wired connection, interact with the environment, and obtain rewards based on its actions, which can help improve the action strategy. With enough iterations, the MEC is able to learn the optimal policy that maximizes the long-term reward.

We use $S \in \mathcal{S}$, $A \in \mathcal{A}$, and $R \in \mathcal{R}_e$ to denote the state, action and reward from their corresponding sets, respectively. The purpose of the RL algorithm is to find an optimal policy π to 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 the optimization problem in (4.34) are introduced as follows.

- State: At the tth time slot, the network state is denoted as

$$\begin{aligned} S_t &= (\mathcal{L}_t, \mathcal{I}_t, \widehat{\text{QoE}}_{t-1}) \in \mathcal{S}, \qquad (4.41) \\ \text{with } \mathcal{L}_t &= \{L_t^1, L_t^2, ..., L_t^{K^{\text{VR}}}\}, \\ \mathcal{I}_t &= \{I_t^1, I_t^2, ..., I_t^{K^{\text{VR}}}\} \\ \widehat{\text{QoE}}_{t-1} &= \{\text{QoE}_{t-1}^1, \text{QoE}_{t-1}^2, ..., \text{QoE}_{t-1}^{K^{\text{VR}}}\}, \end{aligned}$$

  where $\mathcal{L}_t$ is the set containing the positions of all VR users at the tth time slot, where $L_t^k = [X_t^k, Y_t^k, H_t^k]$, $I_t^k = \{1, 0\}$ is the predicted LoS or NLoS status of the kth VR user for the tth time slot, where 1 represents LoS, 0 represents NLoS, and QoE_{t-1}^k is the QoE value calculated by (4.31) of the kth VR user for the $(t-1)$th time slot.

- Action: The action space is written as

$$\begin{aligned} A_t &= \{\widetilde{\mathbf{\Theta}}_t\} \in \mathcal{A}, \qquad (4.42) \\ \text{with } \widetilde{\mathbf{\Theta}}_t &= \{\mathbf{\Theta}_t^1, \mathbf{\Theta}_t^2, ..., \mathbf{\Theta}_t^{\hat{L}^N}\}, \end{aligned}$$

  where $\widetilde{\mathbf{\Theta}}_t$ is the set that includes all possible reflection coefficient matrix given the number of reflection elements N of the RIS and the number of phase shift levels $\hat{L}$, with $\mathbf{\Theta}_t^i$ given in (4.14).

- Reward: The immediate reward R_t is designed as

$$R_t(S_t, A_t) = \sum_{k=1}^{K_{\text{VR}}} \text{QoE}_t^k. \qquad (4.43)$$

The performance of the selected action is determined by the position and LoS/NLoS status of the VR user, which can further influence the long-term QoE of the THz VR system. Therefore, we use the observed position, the LoS/NLoS status, and the QoE of the VR user as observation, and use the QoE as a reward. According to the observed environmental state S_t at the tth time slot, the MEC selects specific action A_t from the set $\mathcal{A}$ and obtains reward R_t. Then, the discounted accumulation of the long-term reward is denoted as

$$Q(S,\pi) = \sum_{i=t}^{\infty} (\gamma)^{i-t} R_i(S_i, A_i), \tag{4.45}$$

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

When the number of reflection elements and phase shift levels is small, the RL algorithm can efficiently obtain the optimal policy. However, when a large number of reflection elements and phase shift levels exist, e.g. 10^{50} ($\hat{L} = 10, N = 50$), the state and action spaces will be increased proportionally, which will not only occupy plenty of computation memory of the MEC, but also inevitably result in massive computation latency and degraded 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 the MEC, and solve complex decision-making problems. Meanwhile, the policy should not violate the VR interaction latency threshold. Thus, we use a constrained deep Q network (C-DQN) to solve the optimization problem, which indirectly optimizes the policy by optimizing the value function while satisfying the downlink latency constraint.

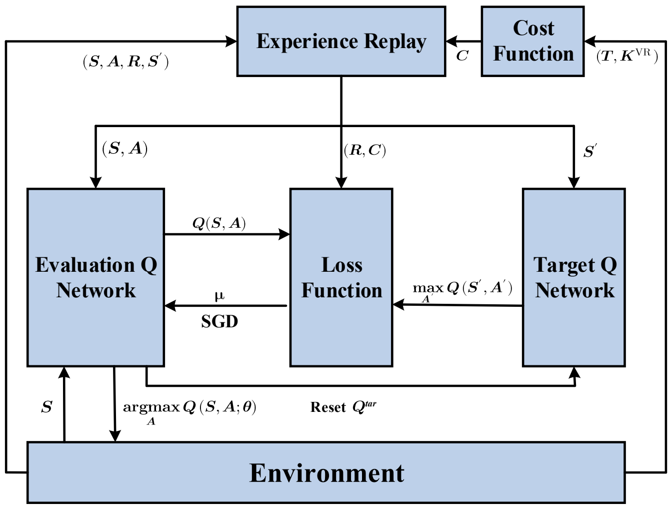


Fig. 4.7 The C-DRL diagram of the THz transmission scheme.

As shown in Fig. 4.7, C-DQN is a value-based DRL algorithm, it combines a neural network with Q-learning and optimizes the state-action value function through a deep neural network (DNN). The C-DQN uses a neural network to store state and action information $Q^{\pi}(S,A)$. It also applies the experience replay to train the learning model, and experiences in the experience replay are partially selected to learn to improve the learning efficiency of the neural network and break the correlation among the training samples. In addition, the distribution of the training samples can be smoothed via averaging the selected samples, which can further avoid the training divergence.

The objective of C-DQN is to find the optimal policy $\pi^{\star}$, and obtain optimal state-action value $Q^{\star}(S,A)$, which is expressed as

$$\pi^{\star}(S) = \arg\max_{A} Q^{\star}(S,A). \tag{4.46}$$

In the proposed optimization problem, C-DQN should satisfy the VR downlink transmission latency constraint. Thus, the state-action value is calculated as

$$Q(S,A) = R + \gamma_{\text{CDQN}} \max_{A'} Q(S',A') - \omega C^{\text{down}}, \tag{4.47}$$

where S' is the next state, A' is the next action, and γ_{CDQN} is the discount factor, which determines the balance between the current state-action value and future state-action value. In (4.46), C^{down} is the downlink transmission cost due to the constraint in (4.35) at each time slot, which is calculated as

$$C^{\text{down}} = \frac{\sum_{k=1}^{K^{\text{VR}}} T^{k}_{\text{downlink}}}{K^{\text{VR}}} - T^{\text{downlink}}_{\text{th}}, \tag{4.48}$$

where K^{VR} is the number of the VR users, $T^{k}_{\text{downlink}}(t)$ is the downlink transmission latency of the kth VR user at the tth time slot, and $T^{\text{downlink}}_{\text{th}}$ is the downlink transmission latency constraint. According to (4.46), the Q evaluation network in C-DQN is used to estimate $Q(S,A)$. Note that the target Q network does not change in each time slot and is updated after several time slots. To update the evaluation state-action value, Bellman Equation is applied, which is denoted as

$$Q^{e}(S,A) = (1-\alpha_{\text{CDQN}})Q^{e}(S,A) + \alpha_{\text{CDQN}} Q^{tar}(S,A), \tag{4.49}$$

where Q^{e} and Q^{tar} are the output of Q evaluation and target network, respectively. In (4.48), α_{CDQN} is the learning rate. The loss function is calculated as $(Q^{tar} - Q^{e})$, which is used to update the weights of the Q evaluation network. We can obtain the optimal policy and Q value when the C-DRL converges. Our detailed C-DRL algorithm is presented in Algorithm 4.

4.3.4 Computational Complexity Analysis of Learning Algorithms

For the computation complexity of the RNN based on the GRU and LSTM architecture, it is 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 computation complexity of the CNN is written as $O(\sum_{i=1}^{L_{\text{CNN}}} \hat{m}_i^2 \hat{n}_i^2 c_{\text{in}} c_{\text{out}})$, where $\hat{m}$ is the length of the output feature map of the Convolution kernel, $\hat{n}$ is the length of the Convolution kernel, c_{in} is the number of the input channels, c_{out} is the number of

Table 4.1 Simulation Parameters of THz VR Network

Indoor scenario size	20 m $\times$ 20 m $\times$ 3 m
Number of MEC	1
Location of MEC	[0, 0, 3 m]
Number of VR users	5
Height of VR user	[1.2 m, 1.8 m]
Location of obstacle (X axis)	[4 m, 8 m], [12 m, 16 m]
Location of obstacle (Y axis)	[8 m, 12 m]
Height of obstacle (Z axis)	3 m
Center Location of RIS	[10 m, 20 m, 3 m]
Speed of Light	3×10^8 m/s
THz center frequency	300 GHz
Number of phase shift elements	20
FoV resolution	4k
MEC execution ability F_{MEC}	5 GHz
Number of cycles processing one bit f_{MEC}	1000 Cycles/bit
Number of antennas of MEC	30
Downlink transmission latency	12 ms
White Gaussian noise σ^2	-110 dBm
LSTM memory size	10
Minibatch size	64
RNN learning rate	0.005
Discount Factor γ	0.9
Number of C-DRL layer	2
Number of C-DRL units of each layer	128
C-DRL Learning rate α_{CDQN}	0.05
Time slots	300

the output channels, and L_{CNN} is the number of CNN layers [46]. The computational complexity of the C-DRL algorithm is given by $O(\bar{m}\bar{n}\log\bar{n})$. Here, $\bar{m}$ is the number of layers, and $\bar{n}$ is the number of units per learning layer [49].

To compare with the C-DRL algorithm, we use an exhaustive algorithm to select the optimal phase shift of the RIS, and the computational complexity of the exhaustive algorithm is denoted as $O(\hat{L}^N)$ [98], where $\hat{L}$ and N are the number of phase shift levels and the number of reflecting elements of the RIS, respectively.

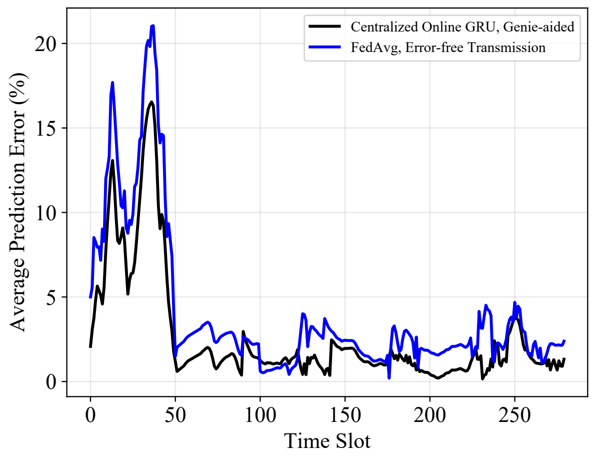


Fig. 4.8 Average prediction error of centralized online GRU and distributed FedAvg algorithms in continuous time slots.

4.4 Simulation Results

In this section, we examine the effectiveness of our proposed learning architecture in Fig. 4.5. The simulation parameters are summarized in Table 4.1.

4.4.1 Viewpoint Prediction

The VR dataset obtained from [4] includes 16 clips of VR videos with 153 VR users, and 969 data samples of the motion in three dimensions, pitch, yaw, and roll, namely, X, Y, and Z viewing angles. The viewpoint ranges of X, Y and Z angles are (-50°, 50°), (-150°, 150°), and (-50°, 50°), respectively. According to [22], the motion of the VR user has strong short-term auto-correlations in all three dimensions. Due to the fact that auto-correlations are much stronger than the correlation between these three dimensions, the angles in each direction can be trained independently and separately. In addition, the range of Y angle distribution is much larger than that of X and Z. Therefore, for simplicity, we use online

GRU to predict the Y angle of VR users in this section, however, our algorithms can also be used for the prediction of X and Z angles. In the simulation, we use the viewpoint samples of the historical ten time slots (1 second) to predict the viewpoint of the next time slot (0.1 seconds).

Fig. 4.8 plots the average prediction error of the centralized online GRU and distributed FedAvg algorithms in continuous time slots. In the Genie-aided scheme, the learning algorithms are trained with the known correct actual viewpoint of each VR user by the MEC at each time slot. In the Error-free Transmission scheme, there are no transmission errors in the uplink and downlink transmission. We can see that the performance of the centralized online GRU is better than that of the FedAvg. This is because the FedAvg depends on the local data of each VR user while minimizing the loss function, and this biases the learning model to be fit for the specific VR user [104], whereas the centralized online GRU can learn from global data of all VR users, so that the learning model can be appropriated for all VR users. It is also noted that at the beginning, there are large fluctuations in the performance of the learning algorithms. This is because the parameters in the learning algorithms should be modified to capture the viewpoint preference of the VR user. In addition, more simulation results of viewpoint prediction have already been described in Section V of [93] in detail.

4.4.2 LoS and NLoS Prediction

During the LoS and NLoS prediction, the algorithm that integrates online LSTM and CNN is deployed to predict the mobility of VR users and judge the LoS/NLoS status of the VR user in continuous time slots. We first use Python to simulate the mobility of VR users in the indoor scenario, and label the moving direction based on the corresponding mobility data. Then, we use the created mobility dataset to train parameters of the LSTM, which can be further used for the online mobility prediction. Fig. 4.9 (a) plots the loss of the LSTM of each epoch. It is seen that the LSTM converges after 60 epochs. Fig. 4.9 (b) plots the prediction error of the LSTM via different number of moving periods. It is noted that when we use the mobility data of the previous 10 time slots to predict the mobility direction for the next time slot, we can obtain the minimum prediction error [93].

Fig. 4.10 (a) plots the loss of CNN for LoS or NLoS prediction of each VR user of each epoch. It is obtained that the CNN converges after 150 epochs. Fig. 4.10 (b) plots the prediction accuracy of CNN via different number of VR users. We observe that the prediction accuracy is about 95% when the number of VR users is smaller than 15, but decreases with increasing number of VR users. This can be explained by the fact that when more VR users exist in the indoor scenario, the features of the input 2D image become

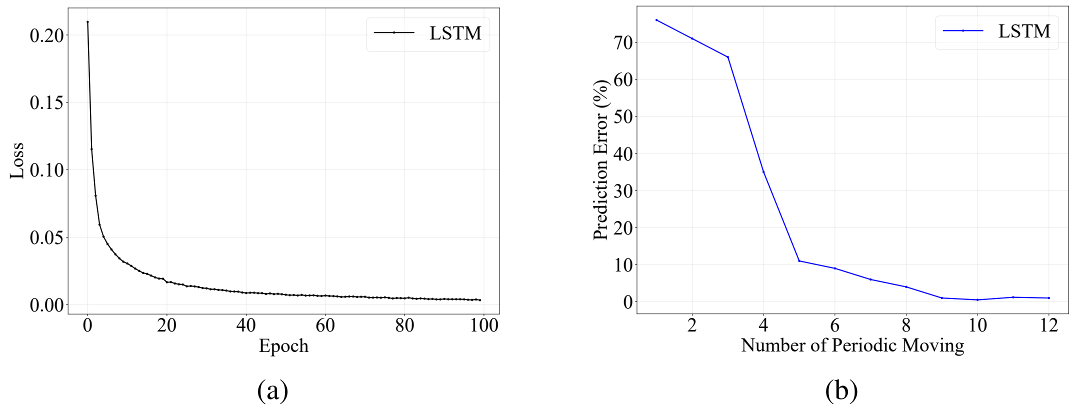


Fig. 4.9 (a) Loss of LSTM for VR user mobility prediction of each epoch. (b) Prediction error of LSTM algorithm via different number of moving periods.

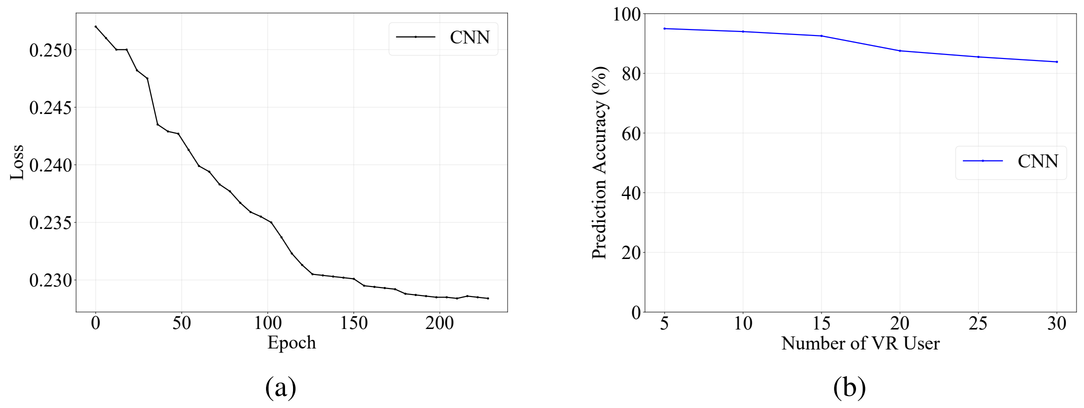


Fig. 4.10 (a) Loss of CNN for LoS or NLoS prediction of each VR user of each epoch. (b) Prediction accuracy of CNN algorithm via different number of VR users.

more complex, and make it difficult for CNN to extract the features with the given structure, which reduces the prediction accuracy.

4.4.3 RIS Configuration of THz Transmission

For the downlink THz transmission, we deploy C-DRL to select the proper phase shift of the RIS to reflect the THz signals for the VR users in NLoS areas. For simplicity, we use "w/ Pred" to present "with prediction". In the Genie-aided scheme, the online learning algorithms are trained with the known correct actual viewpoint and position of each VR user at each time slot, which is the upper bound of the online learning algorithm and can hardly be achieved in the practical wireless VR systems. To compare with the proposed

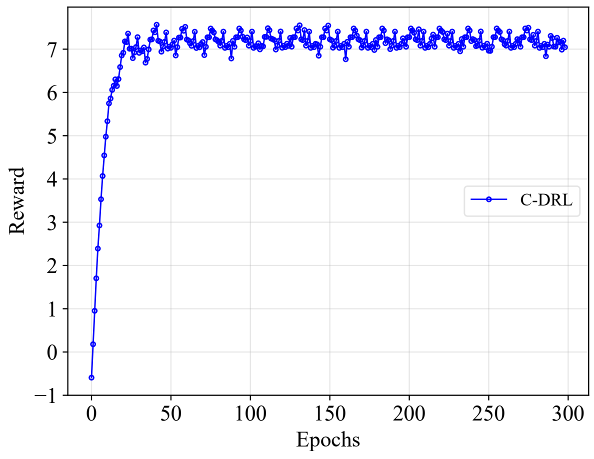


Fig. 4.11 Reward of the MEC-enabled and RIS-assisted THz VR network of each time slot via C-DRL.

learning architecture, an exhaustive algorithm is deployed to select the optimal phase shift of the RIS in downlink transmission at each time slot.

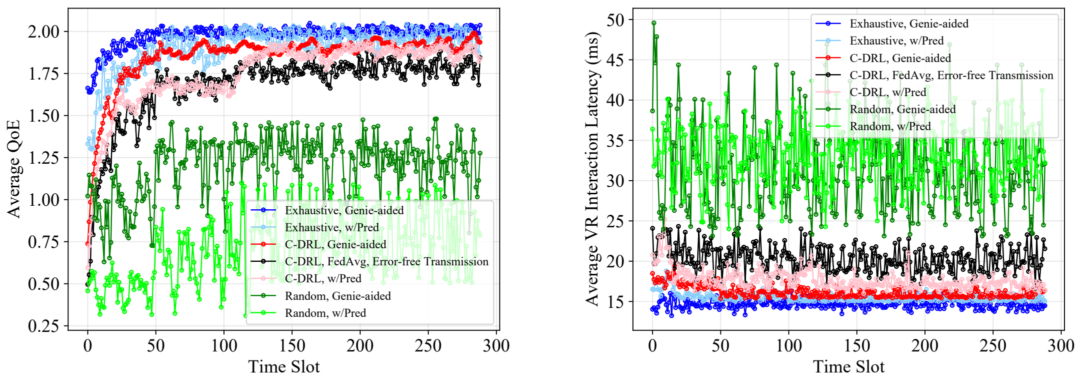


Fig. 4.12 (a) Average QoE of the MEC-enabled and RIS-assisted THz VR network of each time slot via C-DRL with viewpoint and LoS/NLoS prediction. (b) Average VR interaction latency of the MEC-enabled and RIS-assisted THz VR network of each time slot via C-DRL with viewpoint and LoS/NLoS prediction, where the VR interaction latency constraint is 20 ms.

Fig. 4.11 plots the reward of the MEC-enabled and RIS-assisted THz VR network of each time slot via C-DRL. It can be seen that the C-DRL converges after 50 epochs. Fig. 4.12 plots the average QoE and the average VR interaction latency of the MEC-enabled and RIS-assisted THz VR network of each time slot via C-DRL with the uplink viewpoint and LoS/NLoS prediction via GRU compared to that via the exhaustive algorithm, respectively.

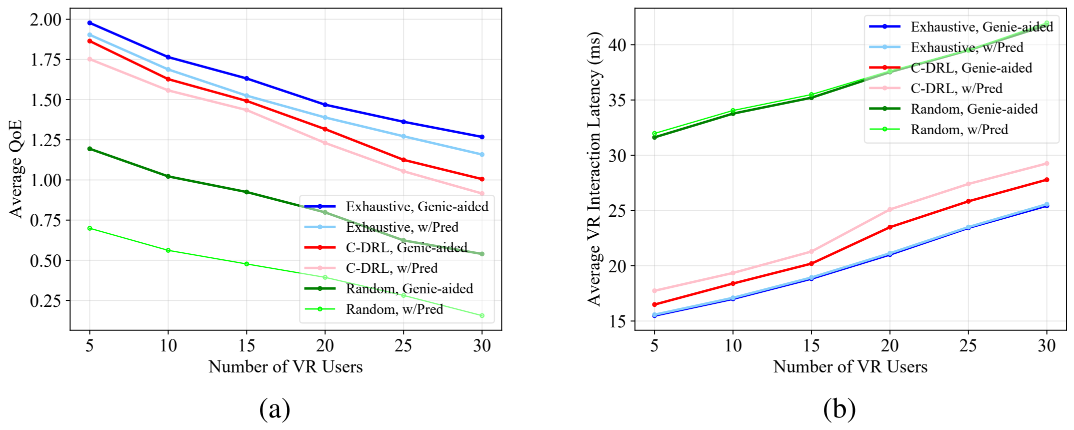


Fig. 4.13 (a) Average QoE of the MEC-enabled and RIS-assisted THz VR network via C-DRL with viewpoint and LoS/NLoS prediction with increasing number of VR users. (b) Average VR interaction latency of the MEC-enabled and RIS-assisted THz VR network via C-DRL with viewpoint and LoS/NLoS prediction with increasing number of VR users, where the VR interaction latency constraint is 20 ms.

It is observed that at the beginning 150 time slots, the average QoE of C-DRL with prediction scheme is worse than that of the C-DRL with the Genie-aided scheme, and both schemes do not violate the VR interaction latency after convergence. This is because in the Genie-aided scheme, the online learning algorithms are directly trained with the known correct actual viewpoint and position of each VR user, so that they are capable of better capturing historical trends of viewpoint preference and mobility of the VR user, which can further improve the prediction accuracy.

Interestingly, we notice that after 150 time slots, the gap between the C-DRL with prediction scheme and the exhaustive with prediction scheme is small. This is due to the experience replay mechanism and randomly sampling in C-DRL, which uses the training samples efficiently and smooths the training distribution over the previous behaviors. Importantly, the performance of all learning-based and exhaustive schemes substantially outperforms the conventional non-learning-based scheme, where the reflection coefficients matrix of the RIS is randomly selected. From Fig. 4.12, we also notice that the performance of the centralized C-DRL with the Genie-aided scheme is better than that of the scheme with FedAvg. This is because in the FedAvg, the learning model needs to be uploaded via uplink transmission for model aggregation, and then the updated global model needs to be transmitted to all VR users through downlink transmission for viewpoint prediction, which leads to extra transmission latency.

Fig. 4.13 plots the average QoE and VR interaction latency of the MEC-enabled and RIS-assisted THz VR network via C-DRL with viewpoint and LoS/NLoS prediction versus the number of VR users compared to that via the exhaustive algorithm, respectively. With

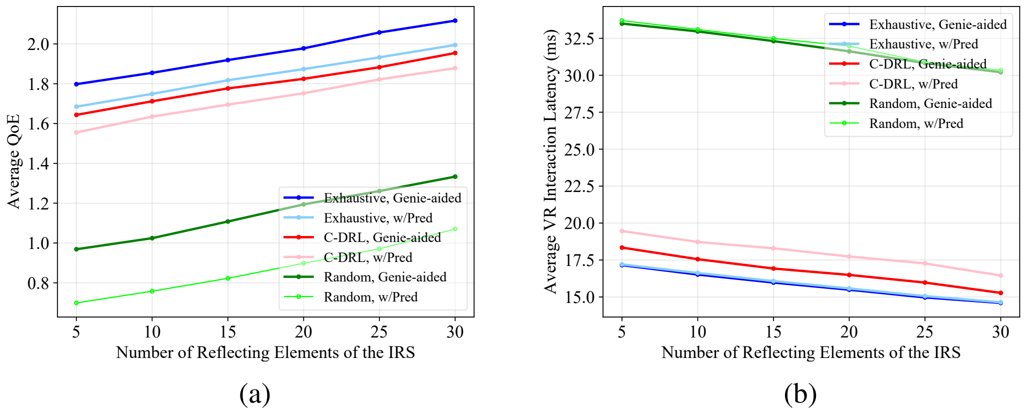


Fig. 4.14 (a) Average QoE of the MEC-enabled and RIS-assisted THz VR network via C-DRL with viewpoint and LoS/NLoS prediction with increasing number of reflecting elements of the RIS. (b) Average VR interaction latency of the MEC-enabled and RIS-assisted THz VR network via C-DRL with viewpoint and LoS/NLoS prediction with increasing number of reflecting elements of the RIS, where the VR interaction latency constraint is 20 ms.

increasing number of VR users, the average QoE of VR users decreases as shown in Fig. 4.13 (a), whereas the average VR interaction latency increases as shown in Fig. 4.13 (b). This is due to the fact that with increasing number of VR users, the interference among the THz transmission increases. When the number of VR user is larger than 15, the gap between the C-DRL and the exhaustive algorithm becomes larger, and the VR interaction latency constraints are violated with increasing number of VR users. This is because the LoS/NLoS prediction accuracy via CNN decreases, which further affects the action selected by the C-DRL.

Fig. 4.14 plots the average QoE and the average VR interaction latency of the MEC-enabled and RIS-assisted THz VR network via C-DRL with viewpoint and LoS/NLoS prediction versus the number of reflecting elements of the RIS compared to that via the exhaustive algorithm, respectively. With increasing number of reflecting elements of the RIS, the average QoE of VR users increases as shown in Fig. 4.14 (a), whereas the average VR interaction latency decreases as shown in Fig. 4.14 (b). This is because as the number of reflecting elements increases, the THz channel gain reflected by the RIS increases [59], which further increases the THz transmission rate for the VR users in NLoS areas. In addition, the VR interaction latency of the non-learning schemes is not influenced by the predicted LoS/NLoS status via CNN.

4.5 Conclusions

In this chapter, a MEC-enabled and RIS-assisted THz VR network was developed to maximize the long-term QoE of real-time interactive VR video streaming in an indoor scenario under VR interaction latency constraints. Specifically, in the uplink, a centralized online GRU algorithm and distributed FedAvg were used to predict the viewpoints of VR users over time, to determine the corresponding FoV to be rendered at the MEC. An algorithm that integrates online LSTM and CNN was also designed to predict the locations of VR users and determine the LoS or NLoS statuses in advance. Then, a C-DRL algorithm was developed to select the optimal phase shifts of the reflecting elements of the RIS to compensate for the NLoS loss in THz transmission. Simulation results have shown that our proposed ensemble learning architecture with online GRU, online LSTM, CNN, and C-DRL algorithms substantially improved the long-term QoE, while satisfying the VR interaction latency constraint, and the QoE performance of our proposed learning architecture was near-optimal compared to the exhaustive algorithm.

Algorithm 4 C-DRL to select the optimal phase shifts of the RIS in THz transmission

```
 1: Initialize replay memory $G$, discount factor $\gamma_{\text{CDQN}} \in [0,1)$, and learning rate $\alpha_{\text{CDQN}} \in$
    $(0,1]$.
 2: Initialize state-action value function $Q(S,A)$, the parameters of evaluation Q network
    and target Q network.
 3: for Iteration = 1,...,$I$ do
 4:     Input the network state $S$.
 5:     for t = 1,...,T do
 6:         Use $\varepsilon$-greedy algorithm to select a random action $A_t$ from the action space $\mathcal{A}$.
 7:         Otherwise, select $A_t = \max_{A \in \mathcal{A}} Q(S_t,A)$.
 8:         The MEC performs downlink transmission according to the selected action $A_t$.
 9:         The MEC observes reward $R_t$, new state $S_{t+1}$ and calculates the cost according to
            (4.47).
10:         Store transition $(S_t,A_t,R_t,C_t^{\text{down}},S_{t+1})$ in replay memory $G$.
11:         Sample random minibatch of transitions $(S_j,A_j,R_j,C_j^{\text{down}},S_{j+1})$ from replay
            memory $G$.
12:         if $j+1$ is terminal then
13:             $y_j^{target} = R_j$.
14:         else
15:             $y_j^{target} = R_{j+1} + \gamma \max_A Q(S_{j+1},A) - \omega C_{j+1}^{\text{down}}$.
16:         end if
17:         Update evaluation Q network.
18:         Update the Lagrangian multiplier with

            $$\omega = \omega + \alpha_{\text{CDQN}} \frac{1}{|G|} \sum_{i=1}^{|G|} C_i^{\text{down}}, \qquad (4.44)$$

            where $|G|$ is the size of the replay memory.
19:         Update target Q network periodically.
20:     end for
21: end for
```

Chapter 5

Conclusions

5.1 Main Conclusions

In this work, we studied machine learning in wireless VR networks. The main conclusions are summarized as follows:

- 5G and THz techniques guarantee high transmission rate and low interaction latency for wireless VR networks. Furthermore, a proactive retransmission scheme in 5G networks and two-ray uplink transmission in THz networks guarantee the reliability of uplink transmission. Also, deploying RIS in THz networks and smartly adjusting the phase shift elements support the THz transmission for VR users in NLoS areas.
- Machine learning algorithms help 5G and THz achieve the QoE requirements of VR users. First, DRL algorithms can select proper MECs to render VR video frames and determine the optimal association between MECs and VR user groups. Second, constrained DRL can select the optimal phase shifts of the RIS under the latency constraint to perform THz transmission for VR users in NLoS areas.
- Online learning algorithms such as linear regression, neural network, LSTM, and GRU can predict the viewpoints of new VR users over time, and LSTM and GRU can capture the time-correlated features of viewpoints in real VR datasets. In addition, federated learning can not only predict the viewpoints of VR users but also guarantee the privacy of VR users.

Although we mainly considered machine learning for wireless VR networks in 5G and THz, the proposed learning and transmission schemes can also be used for traditional video streaming or information transmission in 4G, 5G, mmWave, THz, 6G, and WiFi.

5.2 Summary of Thesis Achievements

As usage of wireless VR networks and machine learning keeps growing, it becomes increasing important to understand synergies between these two fields to improve QoE of VR users and decrease VR interaction latency. Unlike the existing research works mainly deployed traditional optimization methods to optimize wireless networks and did not use real VR datasets, in this thesis, we investigated a number of possibilities to improve wireless VR networks with the help of machine learning.

In Chapter 2, we considered offline and online learning algorithms for uplink wireless VR networks with a proactive retransmission scheme to predict the viewpoints of wireless VR users with real VR datasets. We deployed multiple learning models for multiple VR videos. Also, to generalize these learning models, we considered one learning model for all VR videos. In the online learning algorithms, the online n-order LR, NN, and LSTM/GRU algorithms would update their parameters according to the actual viewpoints delivered from new VR users through uplink transmission, which could further improve the prediction accuracy. Meanwhile, a proactive retransmission scheme was introduced to the online learning algorithms to enhance the reliability of uplink transmission, which can correctly update the parameters and input viewpoints of online learning models.

In Chapter 3, a decoupled learning strategy was developed to optimize real-time VR video streaming in wireless networks, which considered FoV prediction and rendering MEC association. Specifically, based on GRU architecture, an RNN model was used to predict the 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.

Finally, in Chapter 4, a MEC-enabled and RIS-assisted THz VR network was developed to maximize the long-term QoE of real-time interactive VR video streaming in an indoor scenario under VR interaction latency constraints. Specifically, in the uplink, a centralized online GRU algorithm and distributed FedAvg were used to predict the viewpoints of VR users over time, to determine the corresponding FoV to be rendered at the MEC. An algorithm that integrates online LSTM and CNN was also designed to predict the locations of VR users and determine the LoS or NLoS statuses in advance. Then, a C-DRL algorithm was developed to select the optimal phase shifts of the reflecting elements of the RIS to compensate for the NLoS loss in THz transmission.

5.3 Future Works

Various aspects investigated in this thesis can be further developed to improve wireless VR systems. In this section, we detail some of these future works.

- In Chapter 2, 3, and 4, we mainly deployed machine learning algorithms to optimize wireless VR system. However, we do not explain why these machine learning algorithms can improve the QoE of VR users and decrease VR interaction latency. An important research direction could be considering explainable machine learning in wireless VR systems.
- In Chapter 4, through using FedAvg, VR users did not need to transmit local datasets to the MEC, which could guarantee privacy of VR users. However, transmitting learning model can increase transmission latency, especially when the size of learning model is large. Thus, an important research direction is considering adaptive federated dropout in wireless VR networks.

References

[1] (2007). From SISO to MIMO. *MOBILE RADIO Technology*.

[2] (2014). Mobile-edge computing—introductory technical white paper. *White Paper, ETSI*.

[3] (2015). Under the hood: Building 360 video. *https://engineering.fb.com/video-engineering/under-the-hood-building-360-video/*.

[4] (2016). *www.dropbox.com/sh/78ff9djp3v2nv8x/AAACwzDFYwYJzIMrTs8jgM09a*.

[5] (2016). Discussion on HARQ support for URLLC. *R1-1612246, 3GPP TR-RAN1 87*.

[6] (2017). Study on new radio access technology-physical layer aspects. *3GPP TR 38.802 v 14.0.0*.

[7] (2018). Study on scenarios and requirements for next generation access technologies (release 15). *3GPP, TS 38.913 v.15.2.0*.

[8] (2019). 5G; 3GPP virtual reality profiles for streaming applications. *ETSI TS 126 118*.

[9] (2020). 3rd generation partnership project; Technical specification group services and system aspects; Extended reality (XR) in 5G. *3GPP TR 26.928*.

[10] (2020). 5G; NR; User Equipment (UE) radio transmission and reception. *3GPP Ts 38.101-1 v 16.5.0 Release 16*.

[11] (2021). White paper on rf enabling 6G - Opportunities and challenges from technology to spectrum.

[12] Abbas, N., Zhang, Y., Taherkordi, A., and Skeie, T. (2018). Mobile edge computing: A survey. *IEEE Internet Things J.*, 5(1):450 – 465.

[13] Abdullah, D. M. and Ameen, S. Y. (2021). Enhanced mobile broadband (EMBB): A review. *J. Inf. Technol. Informat.*, 1(1):13 – 19.

[14] Agostinelli, F., Hoffman, M., Sadowski, P., and Baldi, P. (2014). Learning activation functions to improve deep neural networks. *arxiv:1412.6830*.

[15] Akyildiz, I. F., Jornet, J. M., and Han, C. (2014a). Terahertz band: Next frontier for wireless communications. *Phys. Commun.*, 12:16 – 32.

[16] Akyildiz, I. F., Jornet, J. M., and Han, C. (2014b). Teranets: ultra-broadband communication networks in the terahertz band. *IEEE Commun. Mag.*, 21(4):130 – 135.

[17] Alliance, N. (2015). 5G white paper.

[18] Alpaydin, E. (1995). Introduction to machine learning. *MIT Press*.

[19] Andrieu, C., Freitas, N. D., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for machine learning. *Machine Learning*, 50(1-2):5 – 43.

[20] Ayoubi, S., Limam, N., Salahuddin, M. A., Shahriar, N., Boutaba, R., Solano, F. E., and Caicedo, O. M. (2018). Machine learning for cognitive network management. *IEEE Commun. Mag.*, 56(1):158 – 165.

[21] Bao, Y., Wu, H., Ramli, A. A., Wang, B., and Liu, X. (2016). Viewing 360 degree videos: Motion prediction and bandwidth optimization. *in Proc. IEEE ICNP*, pages 1161 – 1170.

[22] Bao, Y., Wu, H., Zhang, T., Ramli, A. A., and Liu, X. (2017a). Shooting a moving target: Motion-prediction-based transmission for 360-degree videos. *in Proc. IEEE Int. Conf. Big Data*, pages 1161 – 1170.

[23] Bao, Y., Zhang, T., Pande, A., Wu, H., and Liu, X. (2017b). Motion prediction based multicast for 360-degree video transmissions. *in Proc. IEEE SECON*, pages 1 – 9.

[24] Basar, E., Renzo, M. D., Rosny, J. D., Debbah, M., Alouini, M. S., and Zhang, R. (2019). Wireless communications through reconfigurable intelligent surfaces. *IEEE Access*, 7:116753 – 116773.

[25] Bastug, E., Bennis, M., Medard, M., , and Debbah, M. (2017). Toward interconnected virtual reality: Opportunities, challenges, and enablers. *IEEE Commun. Mag.*, 55(6):110 – 117.

[26] Berg, H. C. (1993). Random walks in biology. *Princeton, NJ, USA: Princeton Univ. Press*.

[27] Bhalla, M. R. and Bhalla, A. V. (2010). Generations of mobile wireless technology: A survey. *Int. J. Comput. Appl.*, 5(4):26 – 32.

[28] Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdiscipl. J. Inf., Knowl. Manage.*, 14:45–76.

[29] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *Proc. 19th Int. Conf. Comput. Statist.*, page 177–186.

[30] Braverman, M., Hazan, E., Simchowitz, M., and Woodworth, B. (2019). The gradient complexity of linear regression. *arxiv:1911.02212*.

[31] Brisimi, T. S., Chen, R., Mela, T., Olshevsky, A., Paschalidis, I. C., and Shi, W. (2018). Federated learning of predictive models from federated electronic health records. *Int. J. Med. Informat.*, 112:59 – 67.

[32] Britz, D., Goldie, A., Luong, M. T., and Le, Q. (2017). Massive exploration of neural machine translation architectures. *arxiv:1703.03906*.

[33] Brown, G. (2016). Exploring the potential of mmwave for 5G mobile access. *Qualcomm White paper*.

[34] Buzzi, S., I, C., Klein, T. E., Poor, H. V., Yang, C., and Zappone, A. (2016). A survey of energy-efficient techniques for 5G networks and challenges ahead. *IEEE J. Sel. Areas Commun.*, 34(4):697 – 709.

[35] Castaneda, M., Chathoth, A., and Nossek, J. A. (2009). Achievable rates in the SIMO-uplink/MISO-downlink of an FDD system with imperfect CSI. *2009 6th International Symposium on Wireless Communication Systems*.

[36] Chaccour, C. et al. (2020a). Risk-based optimization of virtual reality over terahertz reconfigurable intelligent surfaces. *arxiv:2002.09052*.

[37] Chaccour, C. and Saad, W. (2020). On the ruin of age of information in augmented reality over wireless terahertz (THz) networks. *arxiv:2008.09959*.

[38] Chaccour, C., Soorki, M. N., Saad, W., Bennis, M., and Popovski, P. (2020b). Can terahertz provide high-rate reliable low latency communications for wireless VR? *arxiv:2005.00536*.

[39] Chaccour, C., Soorki, M. N., Saad, W., Bennis, M., Popovski, P., and Debbah, M. (2021). Seven defining features of terahertz (THz) wireless systems: A fellowship of communication and sensing. *arxiv:2102.07668*.

[40] Chen, M., Challita, U., Saad, W., Yin, C., and Debbah, M. (2019a). Artificial neural networks-based machine learning for wireless networks: A tutorial. *IEEE Commun. Survey Tuts.*, 21(4):3039 – 3071.

[41] Chen, M., Mozaffari, M., Saad, W., Yin, C., Debbah, M., and Hong, C. S. (2017). Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience. *IEEE J. Sel. Areas Commun.*, 35(5):1046 – 1061.

[42] Chen, M., Saad, W., and Yin, C. (2018). Virtual reality over wireless networks: Quality-of-service model and learning-based resource management. *IEEE Trans. Wireless Comm.*, 66(11):5621 – 5635.

[43] Chen, M., Saad, W., Yin, C., and Debbah, M. (2019b). Data correlation-aware resource management in wireless virtual reality (VR): An echo state transfer learning approach. *IEEE Trans. Comm.*, 67(6):4267 – 4280.

[44] Chen, Z., Lin, T., and Wu, C. (2016). Decentralized learning-based relay assignment for cooperative communications. *IEEE Trans. Veh. Technol.*, 65(2):813 – 826.

[45] Choi, L. U., Ivrla˘c, M. T., Steinbach, E., and Nossek, J. A. (2005). Sequence level models for distortion-rate behaviour of compressed video. *in Proc. IEEE ICIP*, 2:II–486.

[46] Chua, L. O. (1998). *CNN: A Paradigm for Complexity*. Singapore: World Scientific.

[47] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*.

[48] Clay, V., Konig, P., and Konig, S. (2019). Eye tracking in virtual reality. *J. Eye Mov. Res.*, 12(1):284–295.

[49] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, Cambridge, USA.

[50] Courville, A., Goodfellow, I., and Bengio, Y. (2016). Deep learning. *The MIT Press*.

[51] Cuervo, E., Chintalapudi, K., and Kotaru, M. (2018). Creating the perfect illusion : What will it take to create life-like virtual reality headsets? *in Proc. 19th Int. Work. Mob. Comput. Syst. Appl.*, pages 7 – 12.

[52] Dang, T. and Peng, M. (2019). Joint radio communication, caching, and computing design for mobile virtual reality delivery in fog radio access networks. *IEEE J. Sel. Areas Commun.*, 37(7):1594 – 1607.

[53] Deng, Y., Elkashlan, M., Yang, N., Yeoh, P. L., and Mallik, R. K. (2015). Impact of primary network on secondary network with generalized selection combining. *IEEE Trans. Veh. Technol.*, 64(7):3280 – 3285.

[54] Du, J., Yu, F. R., Lu, G., Wang, J., Jiang, J., and Chu, X. (2020). MEC-assisted immersive VR video streaming over terahertz wireless networks: A deep reinforcement learning approach. *IEEE Internet Things J.*, 7(10):9517 – 9529.

[55] Engbert, R., Mergenthaler, K., Breakspear, M., and Pikovsky, A. (2011). An integrated model of fixational eye movements and microsaccades. *Proc. Natl. Acad. Sci.*, 39(108).

[56] Facebook (2022). Oculus rift. *Available: https://www.oculus.com/.*

[57] Fan, C., Lee, J., Lo, W., Huang, C., Chen, K., and Hsu, C. (2017). Fixation prediction for 360 video streaming in head-mounted virtual reality. *NOSSDAV '17*, pages 67–72.

[58] Federici, J. and Moeller, L. (2010). Review of terahertz and subterahertz wireless communications. *J. Appl. Phys.*, 107:111101–1–111101–22.

[59] Feng, K., Wang, Q., Li, X., and Wen, C. K. (2020). Deep reinforcement learning based intelligent reflecting surface optimization for miso communication systems. *IEEE Wireless Commun. Lett.*, 9(5):745 – 749.

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

[61] Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

[62] Guo, C. and Cui, Y. (2019). Optimal multicast of tiled 360 VR video. *IEEE Wireless Commun. Lett.*, 8(1):145 – 148.

[63] Han, C., Bicen, A. O., and Akyildiz, I. F. (2015). Multi-ray channel modeling and wideband characterization for wireless communications in the terahertz band. *IEEE Trans. Wireless Commun.*, 14(5):2402 – 2412.

[64] Han, C. and Chen, Y. (2018). Propagation modeling for wireless communications in the terahertz band. *IEEE Commun. Mag.*, 56(6):96 – 101.

[65] Hastie, T., Tibshirani, R., and Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. *Springer*.

[66] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735 – 1780.

[67] Hoi, S. C. H., Sahoo, D., Lu, J., and Zhao, P. (2018). Online learning: A comprehensive survey. *arXiv:1802.02871.*

[68] Hou, X., Dey, S., Zhang, J., and Budagavi, M. (2018). Predictive view generation to enable mobile 360-degree and VR experiences. *AR/VR Netw. '18*, pages 20–26.

[69] Hu, F., Deng, Y., Saad, W., Bennis, M., and Aghvami, A. H. (2020a). Cellular-connected wireless virtual reality: Requirements, challenges, and solutions. *IEEE Commun. Mag.*, 58(5):105 – 111.

[70] Hu, J., Niu, H., Carrasco, J., Lennox, B., and Arvin, F. (2020b). Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Trans. Veh. Technol.*, 69(12):14413 – 14423.

[71] Huang, C. et al. (2020a). Holographic MIMO surfaces for 6G wireless networks: Opportunities, challenges, and trends. *IEEE Wireless Comm.*, 27(5):118 – 125.

[72] Huang, C., Mo, R., and Yuen, C. (2020b). Reconfigurable intelligent surface assisted multiuser miso systems exploiting deep reinforcement learning. *IEEE J. Sel. Areas Commun.*, 38(8):1839 – 1850.

[73] Huang, C., Yang, Z., Alexandropoulos, G. C., Xiong, K., Wei, L., Yuen, C., and Zhang, Z. (2020c). Hybrid beamforming for RIS-empowered multi-hop terahertz communications: A DRL-based method. *arxiv:2009.09380.*

[74] Huang, C., Yang, Z., Alexandropoulos, G. C., Xiong, K., Wei, L., Yuen, C., Zhang, Z., and Debbah, M. (2021). Multi-hop RIS-empowered terahertz communications: A DRL-based hybrid beamforming design. *IEEE J. Sel. Areas Commun.*, 39(6):1663 – 1677.

[75] Huang, C., Zappone, A., Alexanddropoulos, G. C., Yuen, C., Debbah, M., and Yuen, C. (2019). Reconfigurable intelligent surfaces for energy efficiency in wireless communication. *IEEE Trans. Wireless Commun.*, 18(8):4157 – 4170.

[76] Huq, K. M. S., Busari, S. A., Rodriguez, J., Frascolla, V., Bazzi, W., and Sicker, D. C. (2019). Terahertz-enabled wireless system for beyond-5G ultra-fast networks: A brief survey. *IEEE Netw.*, 33(4):89 – 95.

[77] Jiang, N., Deng, Y., and Nallanathan, A. (2020a). Traffic prediction and random access control optimization: Learning and non-learning based approaches. *arXiv:2002.07759* .

[78] Jiang, N., Deng, Y., Nallanathan, A., and Yuan, J. (2020b). A decoupled learning strategy for massive access optimization in cellular IoT networks. *https://arxiv.org/pdf/2005.01092.pdf.*

[79] Jiang, W., Zhang, Y., Wu, J., Feng, W., and Jin, Y. (2020c). Intelligent reflecting surface assisted secure wireless communications with multiple-transmit and multiple-receive antennas. *IEEE Access*, 8(1):86659–86673.

[80] Kadloor, S., Adve, R. S., and Eckford, A. W. (2012). Molecular communication using brownian motion with drift. *IEEE Trans. Nanobiosci.*, 11(2):89 – 99.

[81] Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *J. Artif. Intell. Res.*, 4(1):237 – 285.

[82] Khan, R., Kumar, P., Jayakody, D. N. K., and Liyanage, M. (2020). A survey on security and privacy of 5G technologies: Potential solutions, recent advancements, and future directions. *IEEE Commun. Surveys Tuts.*, 22(1):196 – 248.

[83] Kingma, D. P. and Jimmy, B. (2014). Adam: A method for stochastic optimization. *arxiv:1412.6980*.

[84] Kuipers, F., Kooij, R., DE, V. D., and Brunnstr¨om, K. (2010). Techniques for measuring quality of experience. *Springer*.

[85] LaValle, S. M., Yershova, A., Katsev, M., and Antonov, M. (2014). Head tracking for the oculus rift. *in Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 187 – 194.

[86] LaViola, J. J. (2003). Double exponential smoothing: an alternative to kalman filter-based predictive tracking. *in Proc. Workshop on Virtual Environments*, pages 199 – 206.

[87] Lee, H., Lee, S. H., and Quek, T. Q. S. (2015). Energy-efficiency oriented traffic offloading in wireless networks: A brief survey and a learning approach for heterogeneous cellular networks. *IEEE J. Sel. Areas Commun.*, 33(4):627 – 640.

[88] Lee, H., Lee, S. H., and Quek, T. Q. S. (2019). Deep learning for distributed optimization: Applications to wireless resource management. *IEEE J. Sel. Areas Commun.*, 37(10):2251 – 2266.

[89] Liang, Q., Que, F., and Modiano, E. (2018). Accelerated primal-dual policy optimization for safe reinforcement learning. *arxiv:1802.06480*.

[90] Lin, Z. (2018). Recurrent neural network models of human mobility. *Ph.D. disserration, Dept. Civil Environ. Eng., Univ. California, Berkeley, Berkeley, CA, USA*.

[91] Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arxiv:1506.00019*.

[92] Liu, X. and Deng, Y. (2021). Learning-based prediction, rendering and association optimization for MEC-enabled wireless virtual reality (VR) networks. *IEEE Trans. Wireless Commun.*, 20(10):6356 – 6370.

[93] Liu, X., Li, X., and Deng, Y. (2021). Learning-based prediction and uplink retransmission for wireless virtual reality (VR) network. *IEEE Trans. Veh. Technol.*, 70(10):10723 – 10734.

[94] Liu, Y., Deng, Y., Elkashlan, M., Nallanathan, A., and Karagiannidis, G. K. (2020). Analyzing grant-free access for URLLC service. *arxiv:2002.07842*.

[95] Long, K., Cui, Y., Ye, C., and Liu, Z. (2019a). Optimal transmission of multi-quality tiled 360 VR video by exploiting multicast opportunities. *IEEE Globecom 2019*.

[96] Long, K., Cui, Y., Ye, C., and Liu, Z. (2019b). Optimal wireless streaming of multi-quality 360 VR video by exploiting natural, relative smoothness-enabled and transcoding-enabled multicast opportunities. *IEEE Trans. Multimedia*, page 1 – 1.

[97] M, V. P. K. and Mahapatra, S. (2018). Quality of experience driven rate adaptation for adaptive HTTP streaming. *IEEE Trans. Broadcast.*, 64(2):602–620.

[98] Ma, X., Chen, Z., Chen, W., Chi, Y., Li, Z., Han, C., and Wen, Q. (2020a). Intelligent reflecting surface enhanced indoor terahertz communication systems. *Nano Commun. Netw.*, 24.

[99] Ma, X., Chen, Z., Chen, W., Li, Z., Chi, Y., Han, C., and Li, S. (2020b). Joint channel estimation and data rate maximization for intelligent reflecting surface assisted terahertz MIMO communication systems. *IEEE Access*, 8:99565 – 99581.

[100] Mao, H., Netravali, R., and Alizadeh, M. (2017a). Neural adaptive video streaming with pensieve. *Proc. Conf. of the ACM Special Interest Group on Data Commun.*, page 197 – 210.

[101] Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017b). A survey on mobile edge computing: The communication perspective. *IEEE Commun. Survey Tuts.*, 19(4):2322–2358.

[102] McMahan, H. B., Moore, E., Ramage, D., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. *In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273 – 1282.

[103] Mehta, H., Patel, D., Joshi, B., and Modi, H. (2014). 0G to 5G mobile technology: A survey. *J. Basic Appl. Eng. Res.*, 1(6):56 – 60.

[104] Mohri, M., Sivek, G., and Suresh, A. T. (2019). Agnostic federated learning. *arxiv:1902.00146.*

[105] Nielsen, M. A. (2014). Neural Networks and Deep Learning. *Determination Press.*

[106] Ning, B., Chen, Z., Chen, W., and Du, Y. (2020). Channel estimation and transmission for intelligent reflecting surface assisted thz communications. *arxiv:1911.04719.*

[107] Ong, H. Y., Chavez, K., and Hong, A. (2015). Distributed deep Q-learning. *arXiv:1508.04186.*

[108] Petrov, V., Pyattaev, A., Moltchanov, D., and Koucheryavy, Y. (2016). Terahertz band communications: Applications, research challenges, and standardization activities. *2016 8th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 183–190.

[109] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., Shyu, M. L., Chen, S. C., and Lyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surveys.*, 52:1–92.

[110] Powell, K. (2019). Nvidia clara federated learning to deliver ai to hospitals while protecting patient data.

[111] Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. *https://arxiv.org/abs/1811.12808*.

[112] Renzo, M. D. et al. (2019). Smart radio environments empowered by reconfigurable AI meta-surfaces: An idea whose time has come. *EURASIP J. Wireless Commun. Netw*.

[113] Renzo, M. D. et al. (2020a). Smart radio environments empowered by reconfigurable intelligent surfaces: How it works, state of research, and the road ahead. *IEEE J. Sel. Areas Commun.*, 38(11):2450 – 2525.

[114] Renzo, M. D., Nltontin, K., et al. (2020b). Reconfigurable intelligent surfaces vs. relaying: Differences, similarities, and performance comparison. *IEEE Open J. Commun. Soc.*, 1:798 – 807.

[115] Roberts, J. A., Wallis, G., and Breakspear, M. (2013). Fixational eye movements during viewing of dynamic natural scenes. *Front. Psychol.*, 4(797).

[116] Rodriguez, J. D., Perez, A., and Lozano, J. A. (2010). Sensitivity analysis of k-fold cross validation in prediction error estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(3):569–575.

[117] Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv:1609.04747*.

[118] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Comput.*, 8(4):14 – 23.

[119] Series, M. (2015). IMT vision-framework and overall objectives of the future development of IMT for 2020 and beyond. *Recommendation ITU*, pages 2083 – 0.

[120] Sharma, S. K. and Wang, X. (2020). Toward massive machine type communications in ultra-dense cellular iot networks: Current issues and machine learning-assisted solutions. *IEEE Commun. Surveys Tuts.*, 22(1):426 – 471.

[121] Shea, K. O. and Nash, R. (2015). An introduction to convolutional neural networks. *arxiv:1511.08458*.

[122] Shirazi, G. N., Kong, P. Y., and Tham, C. K. (2010). Distributed reinforcement learning frameworks for cooperative retransmission in wireless networks. *IEEE Trans. Veh. Technol.*, 59(8):4157 – 4162.

[123] Siddiqi, M. A., Yu, H., and Joung, J. (2019). 5G ultra-reliable low-latency communication implementation challenges and operational issues with iot devices. *Electronics*, 8(981).

[124] Siriwardhana, Y., Porambage, P., Liyanage, M., and Ylianttila, M. (2021). A survey on mobile augmented reality with 5G mobile edge computing: Architectures, applications, and technical aspects. *IEEE Commun. Surveys Tuts.*, 23(2):1160 – 1192.

[125] Song, H., Bai, J., Yi, Y., Wu, J., and Liu, L. (2020). Artificial intelligence enabled internet of things: Network architecture and spectrum access. *IEEE Comput. Intell. Mag.*, 15(1):44 – 51.

[126] Song, H. and Nagatsuma, T. (2011). Present and future of terahertz communications. *IEEE Trans. Terahertz Sci. Technol.*, 1(1):256 – 263.

[127] Song, J., Sheng, M., Quek, T. Q. S., Xu, C., and Wang, X. (2017). Learning-based content caching and sharing for wireless networks. *IEEE Trans. Comm.*, 65(10):4309 – 4324.

[128] Sozinov, K., Vlassov, V., and Girdzijauskas, S. (2018). Human activity recognition using federated learning. *in Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl. Ubiquitous Comput. Commun. Big Data Cloud Comput. Soc. Comput. Netw. Sustain. Comput. Commun*, pages 1103 – 1111.

[129] Staudemeyer, R. C. and Morris, E. R. (2019). Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. *arxiv:1909.09586*.

[130] Stone, P. and Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345 – 383.

[131] Sun, Y., Chen, Z., Tao, M., and Liu, H. (2019). Communications, caching, and computing for mobile virtual reality: Modeling and tradeoff. *IEEE Trans. Comm.*, 67(11):7573 – 7586.

[132] Sung, K. W., Mutafungwa, E., Jantti, R., and *e.l.* (2019). PriMO-5G: making firefighting smarter with immersive videos through 5G. *IEEE 2nd 5GWF*.

[133] Tan, Z., Li, Y., Li, Q., Zhang, Z., Li, Z., and Lu, S. (2018). Supporting mobile VR in LTE networks: How close are we? *in Proc. ACM Meas. Anal. Comput. Syst.*, 2(1):1 – 31.

[134] Turk, M. and Pentland, A. P. (1991). Face recognition using eigenfaces. *In Proc. IEEE Conf. Computer Vision and Pattern Recognition*.

[135] Verma, D., Julier, S., and Cirincione, G. (2018). Federated ai for building ai solutions across multiple agencies. *arxiv:1809.10036*.

[136] Vozmediano, R. M., Montero, R. S., and Llorente, I. M. (2013). Key challenges in cloud computing: Enabling the future internet of services. *IEEE Internet Comput.*, 17(4):18 – 25.

[137] Wan, Z., Gao, Z., Gao, F., Renzo, M. D., and Alouini, M. S. (2020). Terahertz massive MIMO with holographic reconfigurable intelligent surfaces. *arxiv: 2009.10963*.

[138] Wedel, S., Koppetz, M., Skowronek, J., and Raake, A. (2019). Viprovoq: Towards a vocabulary for video quality assessment in the context of creative video production. pages 2387–2395.

[139] Wei, L., Huang, C., Alexanddropoulos, G. C., Yuen, C., Zhang, Z., and Debbah, M. (2021). Channel estimation for RIS-empowered multi-user MISO wireless communications. *IEEE Trans. Commun.*, 69(6):4144 – 4157.

[140] Wei, Y., Yu, F. R., Song, M., and Han, Z. (2018). User scheduling and resource allocation in hetnets with hybrid energy supply: An actor-critic reinforcement learning approach. *IEEE Trans. Wireless Comm.*, 17(1):680–692.

[141] Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550 – 1560.

[142] Wikipedia (2022). Field of view. *available at https://en.wikipedia.org/wiki/ Field of view.*

[143] Wu, G., Yang, C., Li, S., and Li, G. Y. (2015). Recent advances in energy-efficient networks and their application in 5G systems. *IEEE Wireless Comm.*, 22(2):145 – 151.

[144] Wu, Q. et al. (2020a). Beamforming optimization for wireless network aided by intelligent reflecting surface with discrete phase shifts. *arxiv:1906.03165.*

[145] Wu, Q. and Zhang, R. (2019). Intelligent reflecting surface enhanced wireless network via joint active and passive beamforming. *IEEE Trans. Wireless Commun.*, 18(11):5394 – 5409.

[146] Wu, Q. and Zhang, R. (2020a). Beamforming optimization for wireless network aided by intelligent reflecting surface with discrete phase shifts. *IEEE Trans. Commun.*, 68(3):1838 – 1851.

[147] Wu, Q. and Zhang, R. (2020b). Towards smart and reconfigurable environment: Intelligent reflecting surface aided wireless network. *IEEE Commun. Mag.*, 58(1):106 – 112.

[148] Wu, Q., Zhang, S., Zheng, B., You, C., and Zhang, R. (2020b). Intelligent reflecting surface aided wireless communications: A tutorial. *arxiv:2007.02759.*

[149] Wu, Y., Kokkoniemi, J., Han, C., and Juntti, M. (to appear, 2021). Interference and coverage analysis for terahertz networks with indoor blockage effects and line-of-sight access point association. *IEEE Trans. Wireless Commun.*

[150] Yin, X., Jindal, A., Sekar, V., and Sinopoli, B. (2015). A control-theoretic approach for dynamic adaptive video streaming over HTTP. *Proc. 2015 ACM Conf. on Special Interest Group on Data Commun.*, page 325 – 338.

[151] Yu, T., Li, T., Sun, Y., Nanda, S., Smith, V., Sekar, V., and Seshan, S. (2020). Learning context-aware policies from multiple smart homes via federated multi-task learning. *in Proc. IEEE/ACM Int. Conf. Internet Things Design Implement. (IoTDI)*, pages 104 – 115.

[152] Yu, Z., Gong, B., and He, X. (2008). Virtual reality mobility model for wireless ad hoc networks. *J. Syst. Eng. Electron.*, 19(4):819 – 826.

[153] Zhang, C., Patras, P., and Haddadi, H. (2019). Deep learning in mobile and wireless networking: A survey. *IEEE Commun. Surveys Tuts.*, 21(3):2224 – 2287.

[154] Zhang, H., Liu, N., Chu, X., Long, K., Aghvami, A., and Leung, V. C. M. (2017). Network slicing based 5g and future mobile networks: Mobility, resource management, and challenges. *IEEE Commun. Mag.*, 55(8):138 – 145.

[155] Zhang, K., Yang, Z., Liu, H., Zhang, T., and Basar, T. (2018). Fully decentralized multi-agent reinforcement learning with networked agents. *in Proc. Intl. Conf. Machine Learn.*, page 5872– 5881.

[156] Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *J. Internet Services Appl.*, 1(1):7 – 18.

[157] Zheng, H., Kulkarni, S. R., and Poor, H. V. (2021). Attribute-distributed learning: Models, limits, and algorithms. *IEEE Trans. Signal Process.*, 59(1):386 – 398.

[158] Zhou, G., Pan, C., Ren, H., Wang, K., Elkashlan, M., and Renzo, M. D. (2021). Stochastic learning-based robust beamforming design for RIS-aided millimeter-wave systems in the presence of random blockages. *IEEE Trans. Veh. Technol.*, 70(1):1057 – 1061.