Intelligent and Converged Networks
2020, 1(3): 243–257

ISSN 2708-6240
DOI: 10.23919/ICN.2020.0020

Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks

G. M. Shafiqur Rahman*, **Tian Dang, and Manzoor Ahmed**

Abstract: Fog Radio Access Networks (F-RANs) have been considered a groundbreaking technique to support the services of Internet of Things by leveraging edge caching and edge computing. However, the current contributions in computation offloading and resource allocation are inefficient; moreover, they merely consider the static communication mode, and the increasing demand for low latency services and high throughput poses tremendous challenges in F-RANs. A joint problem of mode selection, resource allocation, and power allocation is formulated to minimize latency under various constraints. We propose a Deep Reinforcement Learning (DRL) based joint computation offloading and resource allocation scheme that achieves a suboptimal solution in F-RANs. The core idea of the proposal is that the DRL controller intelligently decides whether to process the generated computation task locally at the device level or offload the task to a fog access point or cloud server and allocates an optimal amount of computation and power resources on the basis of the serving tier. Simulation results show that the proposed approach significantly minimizes latency and increases throughput in the system.

Key words: fog radio access networks; computation offloading; mode selection; resource allocation; distributed computation; low latency; deep reinforcement learning

1 Introduction

The rapid development of Internet of Things (IoT) has enabled the emergence of various latency-sensitive and computation-intensive applications, such as augmented reality, virtual reality, and natural language processing[1,2]. To achieve the full benefits of IoT, sufficient networking and computation infrastructure are mandatory to support instantaneous response and low-latency-based IoT applications. However, IoT devices with constrained computing capability pose a challenge in terms of meeting the computation demand of these applications[3]. Furthermore, the fifth-generation cellular systems have enabled the explosive growth of IoT devices, whose number can reach approximately 24 billion by 2020. Such number of devices not only causes the growth of explosive data but also generates massive computation demand for next-generation wireless networks. Although cloud computing provides a flexible configuration of hardware resources and allows computation-intensive tasks to be uploaded to the cloud for processing in a minimum time. It is usually deployed far away from the users; moreover, it imposes a heavy burden on the fronthaul and generates an intolerable transmission delay that degrades overall system performance[4]. As a remedy to the above limitations, Fog Radio Access Networks (F-RANs) have emerged as a promising architecture with embedded storage and computing capacity to support IoT devices[5,6].

F-RANs extend cloud computing to the network edge and have been integrated into the IoT operating environment to support the demands of the users

• G. M. Shafiqur Rahman, Tian Dang, and Manzoor Ahmed are with the Key Laboratory of Universal Wireless Communications (Ministry of Education), Beijing University of Posts and Telecommunications, Beijing 100876, China. E-mail: shafiq.it@hotmail.com; tiandang@bupt.edu.cn; manzoor.achakzai@gmail.com.

∗ To whom correspondence should be addressed.

Manuscript received: 2020-10-17; revised: 2020-11-02; accepted: 2020-11-28

with real-time response and high automation[7]. By exploiting edge computing and caching, the burden on constrained fronthaul is significantly alleviated, and latency is shortened[8]. In terms of supporting intensive computation demand in F-RANs, each IoT device can select a proper mode, which includes the local, edge, and cloud modes. In the local mode, the task is executed locally at the device level; for edge and cloud modes, each User Equipment (UE) offloads the task either to a Fog Access Point (F-AP) or a remote cloud server. In the cloud mode, UE is served by Remote Radio Heads (RRHs) with centralized signal processing and resource allocation strategies; in the edge mode, the task is executed through the F-APs[9]. The F-RAN paradigm has been improved significantly and enabled processing of computation tasks in the vicinity of the UE; however, latency performance is still unsatisfactory and thus must be further optimized.

Recently, Machine Learning (ML), specifically Deep Reinforcement Learning (DRL), has appealed to the research community and is considered an effective technique for solving many classification challenges and complex problems under high-dimensional state space[10]. By exploiting a Deep Neural Network (DNN), DRLs can estimate precise value function and provide accurate regression in Reinforcement Learning (RL) problems[11, 12]. DRL has already been widely applied in many applications, such as voice recognition, image recognition, large-scale data mining, and transactional behavior analysis, and has shown promising outcomes due to its extraordinary learning capability[13].

Driven by the advancements and significant contributions of DRL, in this work we study a joint DRL-based mode selection, distributed computation resource allocation, and power allocation to achieve low latency in F-RANs. DRL learns the optimal policy and makes an intelligent decision in selecting a proper mode; based on the selected mode, it allocates a precise amount of resources.

1.1 Related work

Computation offloading is a promising solution for low-computation-capability devices running on power batteries. With respect to fog computing, many studies have investigated computation offloading and resource allocation strategies. For instance, in Refs. [14, 15], the nonorthogonal multiple access technique for optimal and suboptimal resource allocation in F-RANs was investigated. In Ref. [15], the main problem was decomposed into subproblems and then solved with matching and sequential convex programming algorithms. A hierarchical fog architecture was considered in Ref. [16], where user devices can offload their tasks to either a fog node or a remote cloud. In Ref. [17], a joint problem of computation and radio resource allocation was studied for offloading the computation task, and an iterative algorithm was proposed to address the problem. In Ref. [18], the multistage stochastic programming approach for offloading computational expensive tasks was investigated to meet the demands of IoT-eHealth for low latency and real-time monitoring in F-RANs. In Refs. [19, 20], the mixed fog/cloud system was considered to offload the computation-intensive task to minimize latency. In Ref. [21], the distributed game methodology in crowd sensing was studied to ensure maximum resource utilization.

However, the above research mainly considered migrating computation tasks from user devices to either edge nodes or cloud-computing servers and utilized less efficient offloading optimization approaches. As a key technique for enabling Artificial Intelligence (AI), ML has recently been used extensively in wireless networks to solve complex problems without explicit programming[22]. In Ref. [11], an actor-critic DRL-based scheme was studied in consideration of joint computation offloading, radio resource allocation, and content caching to minimize end-to-end latency in F-RANs. In Ref. [23], a DRL scheme for IoT edge computing was proposed in consideration of joint computation offloading and multiuser scheduling algorithm to minimize the long-term average weighted sum of delay and power consumption under stochastic traffic arrival. In contrast to Refs. [11, 23], Ref. [24] studied the double Deep Q-Network (DQN) for efficient computation offloading in ultradense Mobile Edge Computing (MEC) networks. Similarly, in Ref. [25], a DRL-based binary computation offloading approach was considered in IoT systems for processing non-partitionable simple tasks. In Ref. [26], a vehicle-assisted DRL-based offloading scheme was investigated under latency constraints in MEC to find the optimal policy and maximize system utility.

In the context of intelligent resource allocation, power allocation problems were studied considering DRL approaches in Refs. [27–31]. In Ref. [27], a DRL for decision-making was used to find the optimal power level for transmission without requiring global information. Meanwhile, for tackling the challenges of network dynamics, resource diversity, and the coupling of resource management with mode selection in F-RAN, a DRL-based joint mode selection and resource management scheme was investigated in Ref. [28]. In Ref. [29], the DRL framework was used to control power in multiuser wireless communication cellular networks. Transmission rate optimization was examined in Refs. [30, 31] to gain a high throughput in the network where DRL is used to allocate an optimal amount of power resources.

1.2 Motivation and contributions

Most previous studies sought to optimize computation resource allocation, mode selection, and power allocation separately. Furthermore, in terms of computation offloading, several works migrate the computation task from the UE to either edge nodes or cloud-computing servers without considering the optimization of resources[15–20]. The aforementioned works did not focus on the distributed solution at the edge, and majority of the tasks were sent to the cloud-computing tier for computation. Given that fog nodes are introduced with limited resources, distributed computation among the F-APs can be promising in tackling computation-intensive tasks at the edge. Therefore, designing an efficient AI-based computation offloading and resource allocation scheme is of considerable interest.

Based on the above observations, this study proposes an efficient and low-complexity distributed DRL framework for computation offloading, jointly considering mode selection, computation resource allocation, and power allocation. The main contributions of this study are summarized as follows:

- An uplink F-RAN-based IoT environment is presented to evaluate the performance in terms of latency. In this architecture, three modes (i.e., local, fog, and cloud modes) are considered. The system allows each UE to select only one mode to execute the generated computation task. Given that cloud computing is aware of the state of the fog nodes, the DRL controller is incorporated into the cloud zone, which is responsible for precise mode selection and resource allocation.
- In the proposed framework, a joint problem of mode selection, computation resource allocation, and power allocation is formulated to minimize latency under the constraints of computing resources and fronthaul capacity, thereby guaranteeing the Quality of Service (QoS), power consumption, and strict mode for each IoT device. Then, this nonconvex problem is transformed as a Markov Decision Process (MDP) problem. The DRL technique is applied to solve the MDP problem and achieve a suboptimal solution. Furthermore, the fixed target network and replay memory are used for the stable training process of neural networks.
- The effects of DRL-based mode selection and resource allocation are illustrated. Numerical results demonstrate the performance gain of the proposal by comparing it with the Q-learning, fixed, and random approaches. Based on the presented solution, our proposed mechanism outperforms the benchmark schemes, distinctly enhances the throughput of the network, and significantly reduces the delay in processing of tasks by approximately 35%–67%.

1.3 Paper organization

The remainder of this paper is organized as follows: Section 2 presents the system model. Section 3 provides the analytical formulation of the latency optimization problem. Section 4 analyzes the DRL-based joint computation offloading and resource allocation scheme. Section 5 presents the simulation results to validate the performance improvement of the proposed DRL scheme. Section 6 concludes the study.

2 System model

Figure 1 depicts the considered system model for

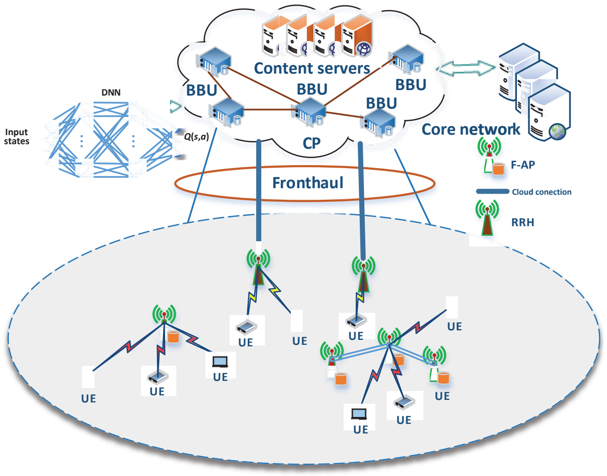

Fig. 1 F-RAN architecture with DRL for computation offloading. Here, CP represents cloud server and BBU represents baseband unit.

computation offloading and resource allocation in uplink F-RANs, where the DRL approach is adopted to accelerate the performance. The system architecture consists of a set of F-APs $\mathcal{L} = \{1, 2, \ldots, L\}$, a set of RRHs $\mathcal{J} = \{1, 2, \ldots, J\}$, and several UEs $\mathcal{K} = \{1, 2, \ldots, K\}$. In the proposed model, a set of IoT devices, namely, smartphones and laptops, are considered to be the UEs. UEs have processing and cache capacities and thus can process the requested computation tasks locally. Each UE k generates computation task $\phi_k = \{M_k, C_k\}$, where M_k is the size of input data for computation measured in bytes, and C_k denotes the required computation resources measured with Central Processing Unit (CPU)-cycle frequency (Hz) to accomplish a computing task ϕ_k[3]. The generated computation task ϕ_k is delay sensitive and can be processed either locally, at the edge tier or on the cloud-computing servers to achieve minimal latency in the system. In terms of task execution, the DRL controller in the cloud makes a precise execution decision based on available execution modes, resource capacity, channel capacity, and transmission rate. Mode selection can be represented as $v_k^m \in \{0, 1\}$, where $m = \{\text{local}, \text{edge}, \text{cloud}\}$ is the offloading mode for each UE k. Subsequently, based on the selected mode, DRL allocates an optimal amount of resources. If the edge is selected as the execution mode, the primary F-AP, which is scheduled with the UE, splits the task into subtasks and sends them to the nearest available assistive F-APs to execute the task in a distributed manner. We assume that each F-AP, RRH as well as the UE, are equipped with a single antenna. In this model, we use the partial frequency multiplexing scheme, and interference is only considered among the UEs.

2.1 Communication model

In this subsection, we present the communication model considered in the study. When the generated computation task ϕ_k cannot be served locally, the UE k offloads the task either to the edge or the cloud server via the wireless interface of the user. We assume the UE offloads the task to the primary F-AP l with the decision of $v_k^{\text{edge}} = 1$ for processing at the edge, where $v_k^{\text{edge}} = 1$ represents that the UE k selects the edge as a suitable mode to offload the task. The obtained signal at the primary F-AP l from user k is expressed as follows:

$$y_{k,l} = \boldsymbol{h}_{k,l}^{\mathrm{H}} p_{k,l} u_k + \sum_{i \neq k, i \in \mathcal{K}} \boldsymbol{h}_{k,l}^{\mathrm{H}} p_{i,l} u_i + n_l, \forall k, l \quad (1)$$

where $\boldsymbol{h}_{k,l}^{\mathrm{H}}$ is the Channel State Information (CSI) matrix, $p_{k,l}$ and $p_{i,l}$ are the uplink transmission power of users k and i, respectively; n_l denotes the additive Gaussian noise received at F-AP l, which is distributed as $(0, \sigma_l^2)$, and u_k is the message of user k.

The Signal-to-Interference-plus-Noise Ratio (SINR) at F-AP l from user k is represented as follows:

$$\mathrm{SINR}_{k,l} = \frac{|\boldsymbol{h}_{k,l}^{\mathrm{H}}|^2 p_{k,l}}{\sum\limits_{i \neq k} |\boldsymbol{h}_{i,l}^{\mathrm{H}}|^2 p_{i,l} + \sigma_l^2}, \forall k, l \quad (2)$$

Moreover, the optimal transmission rate propels the system to ensure the QoS and minimize the transmission delay. Then the data rate, which is achieved at the F-AP l, can be expressed as follows:

$$R_{k,l} = B \log_2(1 + \mathrm{SINR}_{k,l}) \quad (3)$$

where $R_{k,l}$ represents the uplink data rate from UE k to F-AP l, and B denotes the bandwidth for the channel in the network. Given the constrained computation resources of fog nodes, executing all the generated tasks at the edge is impossible. Consequently, the network controller selects the cloud as suitable mode ($v_k^{\text{cloud}} = 1$) to offload the computation tasks to the cloud server through the RRH. The received signal at RRH j from user k can be stated as follows:

$$y_{k,j} = \boldsymbol{h}_{k,j}^{\mathrm{H}} p_{k,j} u_k + \sum_{i \neq k, i \in K} \boldsymbol{h}_{k,j}^{\mathrm{H}} p_{i,j} u_i + n_j, \forall k, j \quad (4)$$

where $p_{k,j}$ is the transmitted power from user k to RRH j, $\boldsymbol{h}_{k,j}^{\mathrm{H}}$ is the CSI matrix for RRH, and n_j is the Gaussian noise at RRH j.

Similarly, the SINR and achievable transmission rate from user k to RRH j can be represented as follows:

$$\mathrm{SINR}_{k,j}=\frac{|\boldsymbol{h}_{k,j}^{\mathrm{H}}|^2 p_{k,j}}{\sum\limits_{i\neq k}|\boldsymbol{h}_{i,j}^{\mathrm{H}}|^2 p_{i,j}+\sigma_j^2},\forall k,j \tag{5}$$

$$R_{k,j}=B\log_2(1+\mathrm{SINR}_{k,j}) \tag{6}$$

where $R_{k,j}$ denotes the uplink data rate from the UE k to RRH j.

The target SINR is defined as the $\gamma_{\min}$ for achieving high QoS in the system. Therefore, Eqs. (2) and (5) suggest that for the selection of the communication mode regardless of edge or cloud, the threshold $\gamma_{\min}$ must be satisfied.

2.2 Delay model

The generated delay in the proposed model is classified as computing and transmission delays. Computation delay is generated because tasks are executed in different tiers, namely, local computation delay, computation delay at the edge, and computation delay at the cloud-computing zone. By contrast, transmission delay is generated for uploading the tasks either from UE to edge or UE to the cloud-computing tier. To download the processed data, the system also generates transmission delay. We present the details of the delay model in the following subsections.

2.2.1 Local computation delay

When the DRL controller is satisfied with the processing capacity of the IoT device, it makes the decision to process the computation task ϕ_k locally ($v_k^{\mathrm{local}}=1$). The CPU of an IoT device is the primary engine for local computation, and the performance of the CPU is characterized by the CPU-cycle frequency[32]. Therefore, local computation delay can be represented as follows:

$$D_k^{\mathrm{local}}=\frac{C_k}{f_k^{\mathrm{local}}} \tag{7}$$

where f_k^{local} is the maximum processing capacity of UE k that can be used to execute the computation task ϕ_k.

2.2.2 Offloading delay at the edge

IoT devices have limited computation resources; thus, when the computation resource requirement C_k cannot be satisfied locally with the decision of the DRL controller ($v_k^{\mathrm{edge}}=1$), the UE k, $k\in\mathcal{K}$, offloads the tasks to the edge through a wireless link. The generated delay at the edge can be divided into transmission delay and computation delay. The transmission delay from UE k to primary F-AP l can be characterized as follows:

$$D_{k,l}^{\mathrm{upload}}=\frac{M_k}{R_{k,l}} \tag{8}$$

where M_k is the size of offloaded data to the F-AP l.

In the context of accomplishing the computation task at the edge, we consider the distributed computation phenomenon, where the available assistive F-APs participate in the computation process with the primary F-AP in a distributed manner. Once the DQN selects the edge as the suitable offloading mode, the primary l-th F-AP splits the task of UE k into $\{1,2,\ldots,N\}$ subtasks, which are equal to the number of participating assistive F-APs N. The set of F-APs, which serves the UE k to process the computation subtasks, can be expressed as $L_l=\{l,l_{k,1},l_{k,2},\ldots,l_{k,N}\}\subseteq L$, in which $l_{k,n}$ represents that l-th primary F-AP incorporated the n-th assistive F-AP to accomplish the subtask n. At this time, the computation resource requirement C_k is divided into $N+1$ parts which can be denoted as $\{C_{k,0},C_{k,1},\ldots,C_{k,N}\}$. Each part of the computation demand is distributedly executed at the edge by the participating F-APs. Thus, the processing delay should be the largest one, i.e.,

$$D_k^{\mathrm{edge}}=\max\left\{\frac{C_{k,0}}{f_{l,k}},\frac{C_{k,1}}{f_{1,k}},\ldots,\frac{C_{k,n}}{f_{n,k}},\ldots,\frac{C_{k,N}}{f_{N,k}}\right\}+\max(M_{l,n}^{\mathrm{input}}+M_{n,l}^{\mathrm{output}})\tau \tag{9}$$

where $f_{l,k}$ and $f_{n,k}$ represent the computation resources of primary F-AP l and assistive F-AP n, which are allocated to the UE k, respectively; and $M_{l,n}^{\mathrm{input}}$ and $M_{n,l}^{\mathrm{output}}$ are the small chunks of input and output data for assistive F-AP n, respectively. τ represents the transmission delay between the primary F-AP l and assistive F-AP n, $(l,n\in\mathcal{L})$, where the longest delay is considered the acceptable transmission delay.

We consider that the primary F-AP l and assistive F-AP n communicate over the wireless interface. In terms of selecting the assistive F-AP, the transmission delay τ

between the primary F-AP l and assistive F-AP n must be less than the threshold delay $\delta_{l,n}$ $(\tau \leqslant \delta_{l,n})$.

Moreover, to make the system model more realistic, we consider the downloading transmission latency for processed data at UE k. Therefore, the downloading delay from the primary F-AP l to UE k can be designated as follows:

$$D_{l,k}^{\text{download}} = \frac{M_k^{\text{processed}}}{R_{l,k}} \tag{10}$$

where $M_k^{\text{processed}}$ is the processed data at the edge, and $R_{l,k}$ is the downlink data rate.

2.2.3 Offloading delay at the cloud

The fog node is introduced with limited computing resources; hence, in some conditions, even the distributed computation scheme is not enough to tackle the computing demand of UE. In this context, the controller selects the cloud mode ($v_k^{\text{cloud}} = 1$) to offload the task ϕ_k. Similarly, in the edge mode, latency is generated to process the task and transmit the input data. Furthermore, the uploading latency for the cloud is generated in two forms, the latency from UE k to the RRH j and RRH j to the cloud server[17]. The transmission latency from UE k to the CP tier through RRH can be expressed as follows:

$$D_{k,\text{CP}}^{\text{upload}} = \frac{M_k}{R_{k,j}} + \frac{M_k}{R_{j,\text{CP}}} \tag{11}$$

where $R_{k,j}$ is the uplink data rate from user k to RRH j and $R_{j,\text{CP}}$ denotes the transmission rate from RRH to cloud-computing tier.

The processing latency at the cloud computing zone for user k can be expressed as follows:

$$D_k^{\text{cloud}} = \frac{C_k}{f_{\text{CP},k}^{\text{cloud}}} \tag{12}$$

where $f_{\text{CP},k}^{\text{cloud}}$ is the maximum processing capacity of cloud servers, which are allocated to UE k to accomplish the computational task.

We consider that the amount of uploaded data is larger than that of processed data. Given that the cloud servers are located thousands of miles away and connected via fiber and core networks, the downloading latency for sending back the result of processed data from the cloud server to UE is also non-negligible[3]. The downloading latency from CP to UE through RRH is represented as follows:

$$D_{\text{CP},k}^{\text{download}} = \frac{M_k^{\text{processed}}}{R_{\text{CP},j}} + \frac{M_k^{\text{processed}}}{R_{j,k}} \tag{13}$$

where $R_{\text{CP},j}$ and $R_{j,k}$ represent the downlink rate from CP to RRH j and RRH j to UE k, respectively. The uplink and downlink data rates vary.

Hereafter, we summarize the overall generated latency in the proposed architecture. The total latency for each layer is calculated as follows:

$$D_{\text{Total}}^{\text{local}} = \frac{C_k}{f_k^{\text{local}}} \tag{14}$$

$$D_{\text{Total}}^{\text{edge}} = \frac{M_k}{R_{k,l}} + \max\left\{\frac{C_{k,0}}{f_{l,k}}, \frac{C_{k,1}}{f_{1,k}}, ..., \frac{C_{k,n}}{f_{n,k}}, ..., \frac{C_{k,N}}{f_{N,k}}\right\} + \max(M_{l,n}^{\text{input}} + M_{n,l}^{\text{output}})\tau + \frac{M_k^{\text{processed}}}{R_{l,k}} \tag{15}$$

$$D_{\text{Total}}^{\text{cloud}} = \frac{M_k}{R_{k,j}} + \frac{M_k}{R_{j,\text{CP}}} + \frac{C_k}{f_{\text{CP},k}^{\text{cloud}}} + \frac{M_k^{\text{processed}}}{R_{\text{CP},j}} + \frac{M_k^{\text{processed}}}{R_{j,k}} \tag{16}$$

where $D_{\text{Total}}^{\text{local}}$, $D_{\text{Total}}^{\text{edge}}$, and $D_{\text{Total}}^{\text{cloud}}$ represent the generated total delay at the local, edge, and cloud-computing tiers, respectively. Therefore, the overall system delay can be represented as follows:

$$D^{\text{system}} = \sum_{k=1}^{K} v_k^{\text{local}} \frac{C_k}{f_k^{\text{local}}} + \sum_{k=1}^{K} v_k^{\text{edge}} \left[\max\left(\frac{C_{k,0}}{f_{l,k}}, \frac{C_{k,1}}{f_{1,k}}, \ldots, \frac{C_{k,n}}{f_{n,k}}, \ldots, \frac{C_{k,N}}{f_{N,k}}\right) + \max(M_{l,n}^{\text{input}} + M_{n,l}^{\text{output}})\tau + \frac{M_k^{\text{processed}}}{R_{l,k}}\right] + \sum_{k=1}^{K} v_k^{\text{cloud}} \left[\frac{M_k}{R_{k,j}} + \frac{M_k}{R_{j,\text{CP}}} + \frac{C_k}{f_{\text{CP},k}^{\text{cloud}}} + \frac{M_k^{\text{processed}}}{R_{\text{CP},j}} + \frac{M_k^{\text{processed}}}{R_{j,k}}\right] \tag{17}$$

The computation demand of each UE k can be served by only one mode at a time.

3 Problem formulation

In this section, the optimization problem of computation offloading is formulated for the uplink F-RAN scenario. The objective of this problem is to minimize the overall system latency by jointly optimizing mode selection, computation resource allocation, and power allocation under the constrains of computing resources, fronthaul capacity, limited transmitting power, and QoS demand of per IoT device. The problem statement can be represented as follows:

$$\begin{aligned}
&\underset{\{v_k^m, f_{l,k}, p_k\}}{\text{minimize}} \; D^{\text{system}} \\
\text{s.t.}\;\; &\text{C1}: \; v_k^{\text{local}} + v_k^{\text{edge}} + v_k^{\text{cloud}} = 1, \; v_k^m \in \{0,1\}, \forall k, \\
&\text{C2}: \; \sum_{k=1}^{K} v_k^{\text{edge}} f_{l,k} \leqslant f_l, \forall l, \\
&\text{C3}: \; \sum_{k=1}^{K} v_k^{\text{cloud}} R_{k,j} \leqslant S_j, \forall j, \\
&\text{C4}: \; v_k^{\text{edge}} \text{SINR}_{k,l} + v_k^{\text{cloud}} \text{SINR}_{k,j} \geqslant \gamma_{\min}, \forall k, \\
&\text{C5}: \; v_k^{\text{edge}} p_{k,l} + v_k^{\text{cloud}} p_{k,j} \leqslant P_{\max}, \forall k
\end{aligned} \tag{18}$$

where the constraint C1 implies that each user k can select only one mode at a time to process their generated computation task. The constraint C2 limits the number of user scheduling to each F-AP l due to its bounded computation resources. The constraint C3 represents that a limited number of UEs can be supported by each RRH j with its constrained fronthaul link capacity S_j. The QoS requirement is guaranteed by the constraint C4 regardless of edge or cloud mode. The constraint C5 illustrates that the uplink transmission rate is bounded by the allocated power of $P_{\max}$. The mode selection v_k^m, computing resource allocation $f_{l,k}$, and power allocation p_k are considered to be the objective variables to solve the formulated optimization problem. p_k is the set of $P_{k,i}$ where $\{1, 2, \ldots, i\}$ is the receiving node of either F-AP l or RRH j, which is determined on the basis of the selected mode.

The stated optimization problem becomes nontrivial and difficult to solve with traditional approaches because of the strict requirement for low latency. Furthermore, time-varying user demand makes Formula (18) more challenging. Generally, traditional exhaustion optimization methods can be applied to solve the formulated problem. However, its computational complexity is considerably high. Moreover, the orchestration of precise mode selection, computing resource allocation, and power allocation provide another level of difficulty to the system. Thus, DRL with low computational complexity is presented to tackle the computational offloading problem with joint mode selection and multitier resource allocation in this uplink F-RAN scenario.

4 DRL-based computation offloading and resource allocation

Formula (18) is a nonconvex multivariate problem, which is challenging to solve with conventional approaches. However, this offloading problem can be modeled under the MDP and can be efficiently solved with the DQN. In this reinforcement learning based problem, Q-learning in the cloud-computing tier acts as a learning agent to achieve the optimal offloading policy after extensively training the system with numerous offloading interactions[2]. We first evaluate the performance with the traditional Q-learning approach and then realize the solution with DRL.

The MDP model can be defined as the tuple of $\{\mathcal{S}, \mathcal{A}, \mathcal{P}(s_{t+1}|s_t, a_t), \mathcal{R}(s_t, a_t)\}$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ represents a set of possible actions, $\mathcal{P}(s_{t+1}|s_t, a_t)$ depicts the state transition probability, and $\mathcal{R}(s_t, a_t)$ is the received reward after performing the action a_t on state s_t. However, state transition probability is hardly obtained in many practical problems. To overcome this hurdle, Q-function $Q(s_t, a_t)$ is utilized as the key parameter. The Q-function is responsible for returning the maximum expected reward by following a policy π that is expressed as follows[11]:

$$Q^{\pi}(s, a) = \mathrm{E}\left[\sum_{t=0}^{T} \xi^t r_t | s_0 = s, a_0 = a, \pi\right] \tag{19}$$

where ξ is a discount factor, r_t represents the obtained reward at time t, s_0 is the initial state, a_0 is an action, and E_{t+1} is the expected return of a trajectory at time $t+1$. The optimal Q-function ensures the maximum cumulative reward, which can be stated as follows:

$$Q^{*}(s, a) = \max_{\pi} Q^{\pi}(s, a) \tag{20}$$

By following the Bellman criterion, the optimal Q-function can be estimated as follows:

$$Q^{*}(s_t, a_t) = \mathrm{E}_{t+1}\left[r_t + \xi \max_{a_{t+1}} Q^{*}(s_{t+1}, a_{t+1})\right] \tag{21}$$

Generally, the Q-function is achieved recursively by exploiting the information of (s, a, r, s') current state, action, immediate reward, and the transition state at the next time span $(t+1)$. Subsequently, the Q-function can be updated as follows:

$$Q_{t+1}(s,a)=\alpha\left(r_t+\xi\max_{a'\in A}Q\left(s_{t+1},a'\right)\right)+ \\ (1-\alpha)Q(s_t,a_t) \tag{22}$$

where α is the learning rate. By utilizing the proper learning rate, the iteration algorithm guarantees that $Q_t(s,a)$ will be converged to optimal $Q^*(s,a)$[31].

4.1 MDP-based computation offloading model

In this subsection, we model the MDP-based computation offloading and resource allocation problem in the following manner.

4.1.1 State space

The system state is designed with the currently available offloading mode for UE k, the distributed computing resources at the edge, the CSI, and the discretized power resources of each IoT device. The system state for time cycle t is stated as $\boldsymbol{s}_t=\{m,f_l,\boldsymbol{h}_{k,i}^{\mathrm{H}},p_k^n\}$. The meaning of each element of the system state is elaborated in the following manner:

- $m=\{\text{local},\text{edge},\text{cloud}\}$ denotes the available modes for UE k to execute the computation task.
- f_l $(l=1,2,\ldots,L)$ represents the available computation resources of F-APs at the edge that can be allocated to the UE k. The serving F-APs have been categorized into primary and assistive F-APs.
- $\boldsymbol{h}_{k,i}^{\mathrm{H}}$ $(i=1,2,\ldots)$ is the CSI vector where i denotes the i-th receiver of either F-AP l or RRH j.
- p_k^n $(n=1,2,\ldots)$ depicts the discretized unit of available transmitting power (in dBm) of UE k. The range of power is represented as $[p_{k,i}^{\min},p_{k,i}^{\max}]$.

4.1.2 Action space

Theoretically, the agent can perform numerous actions. However, to take numerous actions, the system requires massive computation, which produces a huge amount of delay and degrades the system performance. Thus, to avoid computation complexity, the agent considers only one IoT device at each decision epoch t. The action at time slot t is denoted as $a_t\in\mathcal{A}$, where $\mathcal{A}$ is a finite value space $[1,2,\ldots,A]$. The action space for solving the offloading problem can be designed as $\mathcal{A}=\{v_k^m, v_k^{\text{edge}}p_{k,l}+v_k^{\text{cloud}}p_{k,j},f_{l,k}\}$. Each action of the action space is described as follows:

- Based on the given state $\boldsymbol{s}_t$, the agent performs action v_k^m for selecting a mode to execute the generated computation task of UE k.
- After selecting the mode, the agent performs action $p_{k,i}$ for allocating the optimal amount of transmitting power to upload data M_k either to the edge or the cloud. If the agent selects the local mode ($v_k^{\text{local}}=1$), then the uploading transiting power $p_{k,l}=0$ and $p_{k,j}=0$. However, if the agent selects either the edge or the cloud tier to offload the task, the action can be represented on the basis of the selected mode as $v_k^{\text{edge}}p_{k,l}+v_k^{\text{cloud}}p_{k,j}$, where $v_k^m\in\{0,1\}$ and $m\in\{\text{edge},\text{cloud}\}$. If the agent selects the edge as a suitable mode ($v_{k=1}^{\text{edge}}$), then it allocates power $p_{k,l}$ to offload the task from the user k to F-AP l, $v_k^{\text{edge}}=1$. Otherwise, when $v_k^{\text{cloud}}=1$, the agent selects the cloud as an offloading mode and allocates power $p_{k,j}$ for offloading the task from the user k to RRH j. The possible actions for allocating power can be the discrete amount of total power as $\{p_{k,i}^1,p_{k,i}^2,\ldots,p_{k,i}^n\}$. For power allocation in the cloud mode, we only consider allocating power between the UE k and RRH j.
- The action $f_{l,k}$ represents that the computing resources of F-AP l is allocated to the UE k if the controller selects the edge ($v_k^{\text{edge}}=1$) as a suitable mode for offloading the computation task ϕ_k. The action vector can be represented as $\{f_{1,k},f_{2,k},\ldots,f_{L,k}\}$, where $\{1,2,\ldots,L\}$ is the combination of primary F-AP l and assistive F-APs n, $(l,n)\in\mathcal{L}$.

In the actions of mode selection, power allocation, and computation resource allocation, all the constraints C1–C5 have been rigorously considered.

4.1.3 Reward function

The precise reward function helps to find the optimal action policy[33]. Hence, the reward must be defined appropriately for an efficient learning process. After performing a series of actions, mode selection v_k^m, power allocation $v_k^{\text{edge}}p_{k,l}+v_k^{\text{cloud}}p_{k,j}$, and computation resource allocation $f_{l,k}$, we calculate the overall system delay from Eq. (23). Based on the performed action, if the system minimizes delay, the agent receives a positive reward; otherwise, the agent is penalized with a negative reward. The reward function can be defined as follows:

$$r_t = -\Bigg\{ \sum_{k=1}^{K} v_k^{\text{local}} \frac{C_k}{f_k^{\text{local}}} + \sum_{k=1}^{K} v_k^{\text{edge}} \Bigg[\max\Bigg(\frac{C_{k,0}}{f_{l,k}}, \frac{C_{k,1}}{f_{1,k}}, \ldots, \frac{C_{k,n}}{f_{n,k}}, \ldots, \frac{C_{k,N}}{f_{N,k}} \Bigg) + \frac{M_k^{\text{processed}}}{R_{l,k}} + \max(M_{l,n}^{\text{input}} + M_{n,l}^{\text{output}})\tau \Bigg] + \sum_{k=1}^{K} v_k^{\text{cloud}} \Bigg[\frac{M_k}{R_{k,j}} + \frac{M_k}{R_{j,\text{CP}}} + \frac{C_k}{f_{\text{CP},k}^{\text{cloud}}} + \frac{M_k^{\text{processed}}}{R_{\text{CP},j}} + \frac{M_k^{\text{processed}}}{R_{j,k}} \Bigg] \Bigg\} \quad (23)$$

where r_t is the immediate reward. The long-term accumulative reward is considered the minimizing delay at a longer period T. The controller performs the best action by exploring and exploiting each possibility to maximize future accumulative rewards. The long-term cumulative reward is defined as $R_t = \sum_{t=0}^{T} \xi^t r_t$ with a discount factor $\xi \in [0, 1]$. ξ determines the effect of future reward based on current mode selection, power allocation, and computation resource allocation decision. The lower the value of ξ, the more emphasis on immediate rewards. We are attempting to minimize latency; thus, the overall rewards are always negative of latency.

The Q-learning method has been increasingly exploited for solving the RL problem but shows infeasibility for numerous state-action scenarios because in practice, when the state-action pair is sufficiently large, traversing each step with all the samples stored in a Q-table is challenging[33]. This behavior inherently limits the traditional RL with fully observed low-dimensional state space. To overcome the drawbacks of Q-learning, DRL is proposed in this work.

4.2 DRL-based computation offloading and resource allocation

Herein, we propose the DRL-based computation offloading scheme to improve computation performance by accelerating the learning process. DRL can find the optimal policy without explicit prior knowledge of the network. In DRL, the Neural Network (NN) can be trained directly without exploiting the handcrafted features, thus considerably reducing system complexity[3, 34]. Moreover, the replay memory leverages the system performance by finding the optimal policy with few interactions. To avoid the drawbacks of Q-learning in large state and action space problems, the DNN is used as a function approximator to approximate the actionvalue function $Q(s, a; \omega) \approx Q^*(s, a)$, which immensely enhances the learning capability[34]. The entire training process and computation offloading strategy are illustrated in Fig. 2 and Algorithm 1.

Figure 2 and Algorithm 1 represent that the DNN takes the current state $\boldsymbol{s}_0$ as an input, which consists of available offloading modes, computation resources at the edge, CSI, and maximum power resource of IoT device $\boldsymbol{s}_t = \{m, f_l, \boldsymbol{h}_{k,i}^{\text{H}}, p_k^n\}$. By corresponding with all the possible actions, the Q-value $Q(s, a; \omega)$ is derived as the output by adjusting the weight of NN parameter ω. For trade-off between exploitation and exploration, the IoT device selects the offloading policy based on the output of DNN according to the ε-greedy policy[32]. The first and foremost action of the agent is to select the precise computation mode v_k^m. Based on the selected mode, the agent performs further actions to allocate the resources. In Algorithm 1, Lines 15–21 show that if the agent selects edge mode ($v_k^{\text{edge}} = 1$), then the agent takes the actions for power allocation and computation resource allocation consecutively. However, if the agent goes with cloud mode ($v_k^{\text{cloud}} = 1$), then it only performs the power allocation action. By contrast, if the agent selects local computation mode ($v_k^{\text{local}} = 1$), then the agent does not perform any further action, and the computation process is accomplished by the local resources. Subsequently, the agent moves to a new state ($\boldsymbol{s}_{t+1}$) and calculates the rewards r_t from Eq. (23) as the negative of total system delay. In the replay memory g, the transition (s_t, a_t, r_t, s_{t+1}) of each time slot t is stored as the experience. The DQN trains the network by randomly sampling this transition as a minibatch $M = 32$ and updates the parameter ω of the Q-network by minimizing the loss function as follows:

$$L(\omega) = \text{E}_{s,a,r,s'} \Bigg[(r_t + \xi \max_{a' \in A} \hat{Q}(s_{t+1}, a'; \hat{\omega}) - Q(s_t, a_t; \omega))^2 \Bigg] \quad (24)$$

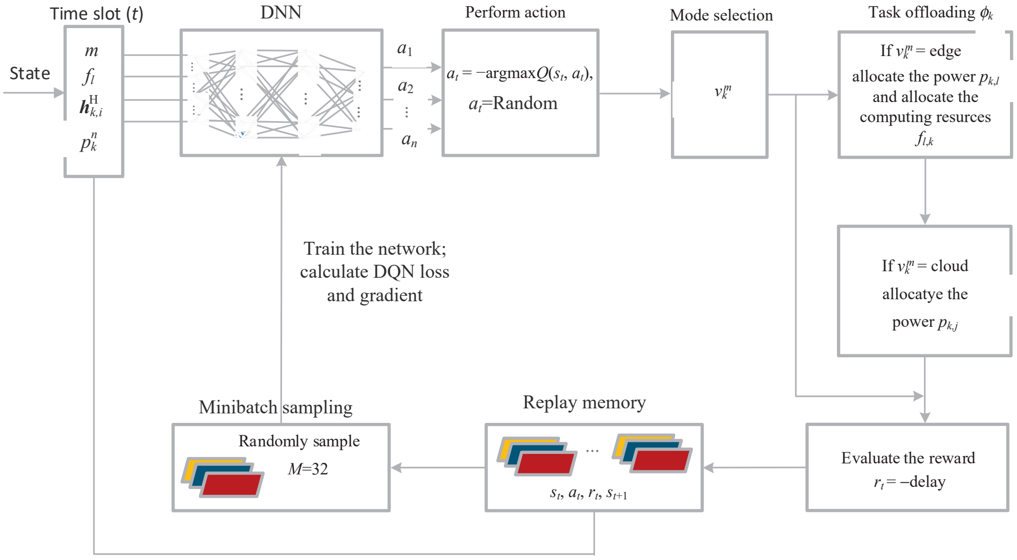
Time slot (t)
State
m
f_l
$\boldsymbol{h}^{\mathrm{H}}_{k,i}$
p^n_k
DNN
a_1
a_2
a_n
Perform action
$a_t = -\mathrm{argmax}Q(s_t, a_t)$,
a_t=Random
Mode selection
v^m_k
Task offloading ϕ_k
If v^m_k = edge allocate the power $p_{k,l}$ and allocate the computing resurces $f_{l,k}$
If v^m_k = cloud allocatye the power $p_{k,j}$
Train the network; calculate DQN loss and gradient
Minibatch sampling
Randomly sample
M=32
Replay memory
s_t, a_t, r_t, s_{t+1}
Evaluate the reward
$r_t = -$delay

Fig. 2 DRL-based learning process and computation offloading for IoT devices.

where $(r_t + \xi \max_{a' \in A} \hat{Q}(s_{t+1}, a'; \hat{\omega})$ is the optimal target Q-function.

After a specific time, the agent sets the weights of the DQN to the target DQN to update the network.

5 Simulation result

In this section, we develop the simulations to evaluate the performance of the proposed DRL-based computation offloading and resource allocation approach. We compare the proposed scheme with three approaches and provide the detailed breakdown of obtained results.

5.1 Simulation settings

The simulation platform is developed with TensorFlow 1.11.0, Python 3.6, Core i5 @ 1.6 GHz 8 CPU, and Intel UHD graphics 620. The deployment area is 400 × 400 m^2, where 10 F-APs, 5 RRHs, and 30 IoT devices are incorporated to evaluate the system performance. The maximum transmission power for each IoT device is considered to be 18 dBm. We assume that the path loss model is $128 + 37 \times \log_{10} d$, where d denotes distance, the noise power spectral density is 40 dBm/Hz, and the system bandwidth is determined as 10 MHz. To ensure the QoS of the system, the lower bound of the SINR is considered $\gamma_{\min}$. We consider a fully connected DNN. The NN consists of two hidden layers, one input layer and an output layer. A total of 64 and 32 neurons are incorporated for the first and second hidden layers, respectively. The Rectified Linear unit (ReLu) is used as the activation function. The simulation parameters are summarized in Table 1.

5.2 Convergence performance

In this subsection, we present the convergence performance of learning parameters under the DRL approach and compare the convergence performance of the proposed DRL-based computation offloading and resource allocation scheme with other well-studied algorithms.

As shown in Fig. 3, with the batch size $M = 32$, DRL achieves better convergence performance and incurs lower cost compared with batch sizes 8 and 64. The reason is that when DRL chooses batch $M = 8$, the system needs a long time to achieve good policy and incurs a high cost. On the contrary, with a large batch size $M = 64$, the system calculates the gradient more accurately, but the learning process may be trapped in the local optimum and incur higher cost than the batch size $M = 32$. Similarly, Fig. 4 exhibits that with the learning rate of $\alpha = 0.01$, the system shows early convergence and incurs the lowest cost. When the learning rate is too low ($\alpha = 0.001$), the system undergoes a prolonged learning process, whereas when the learning rate is too large ($\alpha = 0.09$), the result may be trapped in the local optimum and incur higher cost[9].

Figure 5 depicts the overall loss and convergence behavior of the DRL algorithm. After approximately 2800 epochs, the DRL scheme achieves minimum

Algorithm 1 DRL-based algorithm for computation offloading and resource allocation in F-RANs

```
1:  **Initialization:**
2:  Initialize $Q$-network $Q(s,a)$ with random weights $\omega$.
3:  Construct a target $Q$-network $\hat{Q}(s,a)$ with weights $\hat{\omega}$.
4:  Initialize the replay memory $g$ with capacity $N_D$.
5:  Size of minibatch $M$.
6:  Maximum training episodes $E_{\max}$.
7:  **Iteration:**
8:  **for** *episode*$1, E_{\max}$ **do**
9:      Reset simulation parameters for the computation offloading environment.
10:     Set the initial state $\boldsymbol{s}_0 = [m, f_l, \boldsymbol{h}_{k,i}^{\mathrm{H}}, p_k^n]$.
11:     **for** *decision step* $t = 1 : T-1$ **do**
12:         Generate a random number $x$ between 0 and 1.
13:         **if** $x \leqslant \varepsilon$ **then**
14:             Perform a random action $a_t$ for selecting mode as $v_k^m$.
15:             **if** $v_k^m ==$ edge **then**
16:                 Take the action $p_{k,l}$ and $f_{l,k}$ based on $\varepsilon$ for allocating the power resources and computation resource consecutively.
17:             **else if** $v_k^m ==$ cloud **then**
18:                 Take the action $p_{k,j}$ based on $\varepsilon$ for allocating transmitting the power resources
19:             **else**
20:                 Execute the task with local resources
21:             **end if**
22:         **else**:
23:             Perform an optimal action $a_t$ as $a_t = \arg\max_{a_t \in A} Q(s_t, a_t; \omega)$
24:             Apply the steps from 15 to 21 with optimal action for mode selection, power allocation and computation resource allocation $[v_k^m, p_k, f_{l,k}]$
25:         **end if**
26:         Execute action $a_t$ and evaluate the delay of the system
27:         Calculate the reward $r_t$ from Eq. (23).
28:         Store the reward $r_t$ together $s_t$, $s_{t+1}$, and $a_t$ as an interaction sample $(s_t, a_t, r_t, s_{t+1})$ into the replay memory $g$
29:         Randomly sample the minibatch with the size $M$ of transitions $(s_t, a_t, r_t, s_{t+1})$ from the replay memory $g$
30:         Train the $Q$-network by minibatch gradient descent on $(r_t + \xi \max_{a' \in A} \hat{Q}(s_{t+1}, a'; \hat{\omega}) - Q(s_t, a_t; \omega))^2$ with $\omega$
31:         Periodically update target $Q$-network with parameter $\omega$ to $\hat{\omega}$
32:     **end for**
33: **end for**
```

loss and shows stability. Figure 6 shows that the proposed approach outperforms the other schemes in convergence performance, thus ensuring a reduced complexity of the DRL algorithm. The reason is that the DRL continuously learns with a DNN and achieves higher learning efficiency than the other schemes. DRL can be considered to achieve minimal latency, which is 24% lower than the random scheme and 11% lower than the Q-learning approach. The random scheme achieves the least efficiency in convergence performance.

Table 1 Summary of simulation parameters.

Parameter	Value
Number of fog access points L	10
Number of remote radio heads J	5
Number of UEs K	30
Noise power (dBm/Hz)	-140
Channel bandwidth B (MHz)	10
Pathloss model	$128 + 37 \times \log_{10} d$
Size of replay memory N_D	2000
Learning rate α	0.01
Size of minibatch M	32
Discount factor ξ	0.9

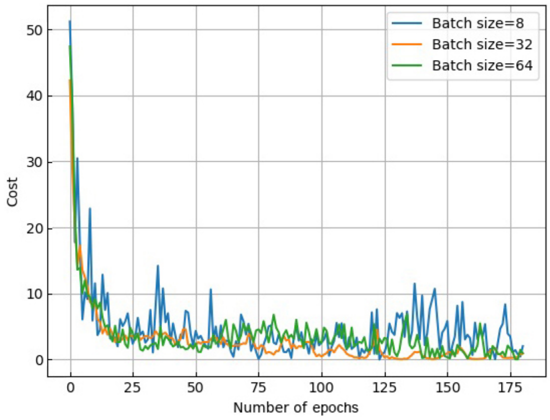

Fig. 3 Evaluation of cost function with the number of epochs under different batch sizes.

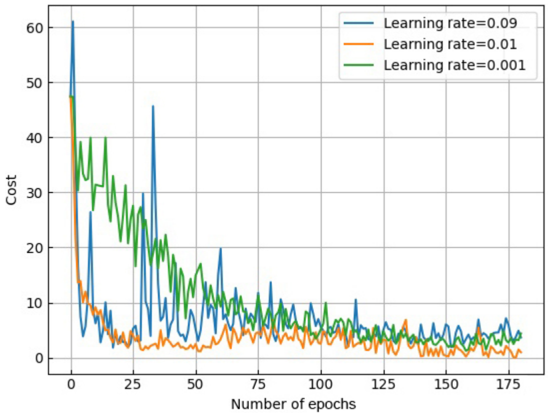

Fig. 4 Evaluation of cost function with the number of epochs under different learning rates.

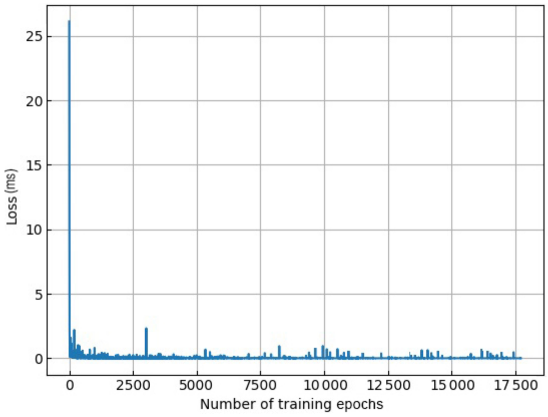

Fig. 5 Total loss during the training process.

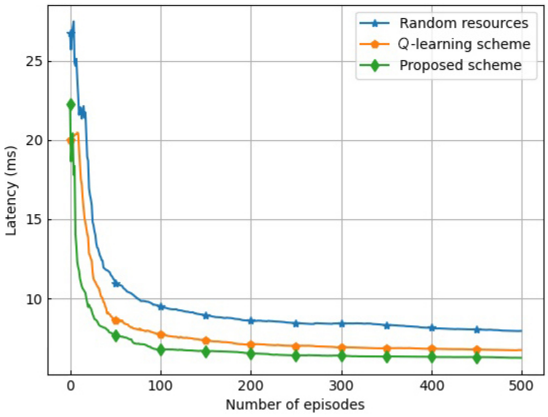

Fig. 6 Evaluation of convergence performance of the algorithms with number of episodes.

5.3 Performance analysis of QoS, mode selection, and resource allocation

Figure 7 exposes the effect of different QoS in minimizing latency. To establish the communication link between either the UE and F-AP or UE and RRH, the system must satisfy the threshold $\gamma_{\min}$. The horizontal axis shows the maximum transmitting power for each UE. The highest QoS requirement achieves the lowest latency because for a high QoS, the system assigns additional power resources in the communication mode, thus, increasing the throughput of the network and immensely minimizing latency. Figure 7 depicts that with the highest QoS demand of 300 kbps, the system achieves the lowest delay of approximately 4 ms, whereas the maximum transmitting power is considered to be 18 dBm.

In Fig. 8, we evaluate the offloading performance of 100 computational tasks under different modes. As shown in Fig. 8, the joint computation offloading scheme achieves the lowest latency because with the extensive training of the NN, the DQN has learned the optimal offloading actions and precisely assigned the task to the respective modes. When the IoT devices execute all the tasks by themselves, the system generates the highest delay by comparison. The cloud and edge modes produce relatively lower delay compared with the local mode under the same number of computational tasks.

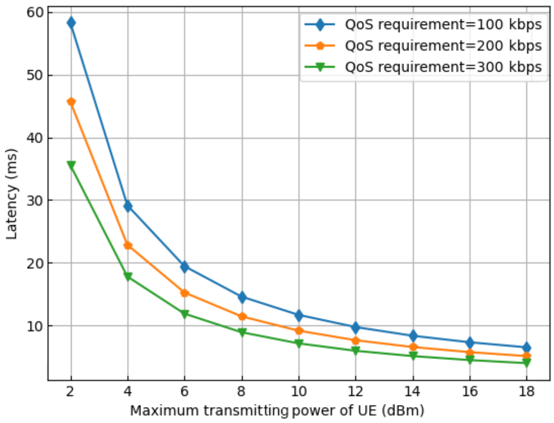

Fig. 7 Evaluation of generated latency under different QoS requirements.

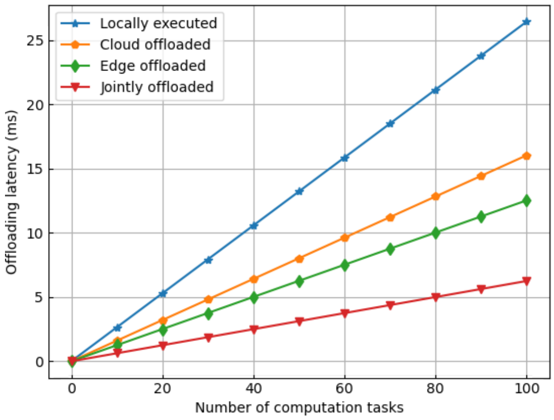

Fig. 8 Offloading delay versus number of computation tasks under different execution modes.

Figure 9 assesses the efficiency of power allocation under different schemes of minimizing latency. With the increase in transmitting power from 2 to 18 dBm, the latency is significantly reduced by the proposed scheme, which outperforms other approaches. The reason is that the DRL efficiently learns the policy on the basis of available resources, channel capacity, and location of the fog nodes and performs a precise action. *Q*-learning and random strategies achieve low efficiency in minimizing latency, and they generate approximately 31% and 51%

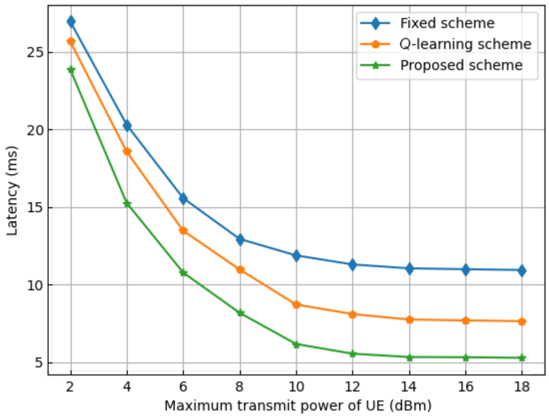

Fig. 9 Performance evaluation of power allocation under different schemes.

higher latency compared with the proposed scheme, respectively.

Figure 10 illustrates the relationship between the number of computation resources and the offloading delay. As shown in Fig. 10, the offloading delay keeps dropping with the increase in the number of computation resources. The DRL-based distributed computation of resource allocation scheme obtains high efficiency in minimizing latency under the same amount of computation resources. The reason is that the proposed scheme incorporates the F-APs wisely on the basis of their location and resource availability. The random approach causes the highest delay. The Q-learning and fixed approach nearly achieve similar efficiency.

5.4 Performance evaluation of the proposed scheme with different benchmarks

We compare the performance of the proposed scheme with three prominent approaches. The evaluation is performed under the same number of computation tasks. Figure 11 shows that the overall offloading delay of the system increases with the growing number of computation tasks from 0 to 100. As shown in Fig. 11, the proposed joint DRL-based computation offloading and resource allocation scheme outperforms the other three schemes. The proposed approach gains approximately 35%–67% lower latency than the other baselines because of the dynamic learning capacity of the DQN. Given the revisiting nature of Q-learning[33], it causes 35% higher latency compared with the proposed scheme. The fixed and random approach shows low efficiency in minimizing latency.

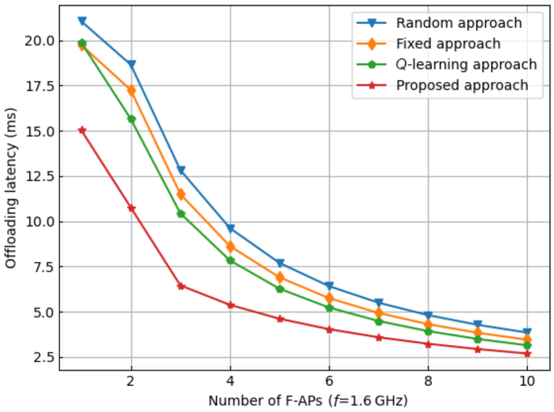

Fig. 10 Performance evaluation of the distributed computation of resource allocation for minimizing latency under different approache.

Table 2 shows the training time and required epochs for evaluating the efficiency of the adopted schemes. The DRL scheme consumes a minimum time of 109.1269 s and takes the least number of epochs (27 000) to accomplish the tasks. Q-learning spends more time than the DRL and takes approximately 29 453 epochs. The random approach consumes the longest time frame, which is almost three times larger than the proposed scheme, whereas the time consumption of the fixed scheme is nearly double than that of the DRL (i.e., 231.6821 s).

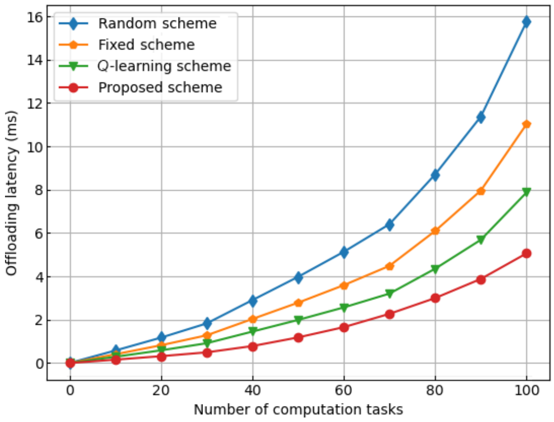

Fig. 11 Comparison of performance of the proposed scheme with different baselines.

Table 2 Comparison of performance among different schemes.

Scheme	Number of epochs	Time (s)
DRL	27 000	109.1269
Q-learning	29 453	153.1838
Fixed	39 000	231.6821
Random	44 301	286.8108

6 Conclusion

Efficient computation offloading is a promising approach to improve the computational capabilities of IoT devices in F-RANs. In this study, we investigated DRL-based joint computation offloading and resource allocation strategies for IoT devices to achieve low latency in F-RANs. A joint mode selection and resource allocation problem is formulated as a nonconvex optimization problem. To solve the problem, a low-complexity DRL scheme, which utilizes the DNN to accelerate the learning speed in the system, is proposed. The DRL can determine the offloading policy efficiently without any explicit assumption about the operating environment and wisely allocate the resources in the network. Furthermore, the distributed computation resource allocation strategy is applied to enhance the computational capacity at the edge. Extensive simulation results demonstrate that the proposed method generates the best policy and outperforms other methods by achieving approximately 35%–67% lower latency.

References

[1] B. Cao, Y. X. Li, L. Zhang, L. Zhang, S. Mumtaz, Z. Y. Zhou, and M. G. Peng, When internet of things meets blockchain: Challenges in distributed consensus, *IEEE Netw.*, vol. 33, no. 6, pp. 133–139, 2019.

[2] M. H. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. H. Zhuang, Learning-based computation offloading for IoT devices with energy harvesting, *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, 2019.

[3] J. H. Zhao, Q. P. Li, Y. Gong, and K. Zhang, Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks, *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, 2019.

[4] B. Cao, L. Zhang, Y. Li, D. Q. Feng, and W. Cao, Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework, *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 56–62, 2019.

[5] M. G. Peng, S. Yan, K. C. Zhang, and C. G. Wang, Fog-computing-based radio access networks: Issues and challenges, *IEEE Netw.*, vol. 30, no. 4, pp. 46–53, 2016.

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

[7] L. X. Chen, P. Zhou, L. Gao, and J. Xu, Adaptive fog configuration for the industrial internet of things, *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4656–4664, 2018.

[8] H. Y. Xiang, M. G. Peng, Y. H. Sun, and S. Yan, Mode selection and resource allocation in sliced fog radio access networks: A reinforcement learning approach, *IEEE Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4271–4284, 2020.

[9] Y. H. Sun, M. G. Peng, and S. W. Mao, Deep reinforcement learning-based mode selection and resource management for green fog radio access networks, *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1960–1971, 2019.

[10] L. T. Tan and R. Q. Hu, Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning, *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10 190–10 203, 2018.

[11] Y. F. Wei, F. R. Yu, M. Song, and Z. Han, Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning, *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, 2019.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] C. Wu, T. Yoshinaga, Y. S. Ji, T. Murase, and Y. Zhang, A reinforcement learning-based data storage scheme for vehicular ad hoc networks, *IEEE Trans. Veh. Technol.*, vol. 66, no. 7, pp. 6336–6348, 2017.

[14] B. H. Liu, C. X. Liu, M. G. Peng, Y. Q. Liu, and S. Yan, Resource allocation for non-orthogonal multiple access-enabled fog radio access networks, *IEEE Trans. Wirel. Commun.*, vol. 19, no. 6, pp. 3867–3878, 2020.

[15] B. H. Liu, C. X. Liu, and M. G. Peng, Resource allocation for energy-efficient MEC in NOMA-enabled massive IoT networks, *IEEE J. Sel. Areas Commun.*, doi: 10.1109/JSAC.2020.3018809.

[16] L. Q. Liu, Z. Chang, X. J. Guo, S. W. Mao, and T. Ristaniemi, Multiobjective optimization for computation offloading in fog computing, *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, 2018.

[17] Z. Y. Zhao, S. Q. Bu, T. Z. Zhao, Z. P. Yin, M. G. Peng, Z. G. Ding, and T. Q. S. Quek, On the design of computation offloading in fog radio access networks, *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 7136–7149, 2019.

[18] L. Zhang, B. Cao, Y. Li, M. G. Peng, and G. Feng, A multi-stage stochastic programming based offloading policy for fog enabled IoT-eHealth, *IEEE J. Sel. Areas Commun.*, doi: 10.1109/JSAC.2020.3020659.

[19] J. B. Du, L. Q. Zhao, J. Feng, and X. L. Chu, Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee, *IEEE Trans. Commun.*, vol. 66, no. 4, pp. 1594–1608, 2018.

[20] Y. M. Liu, F. R. Yu, X. Li, H. Ji, and V. C. M. Leung, Distributed resource allocation and computation offloading

in fog and cloud networks with non-orthogonal multiple access, *IEEE Trans*. *Veh*. *Technol*., vol. 67, no. 12, pp. 12 137–12 151, 2018.

[21] B. Cao, S. C. Xia, J. W. Han, and Y. Li, A distributed game methodology for crowdsensing in uncertain wireless scenario, *IEEE Trans*. *Mobile Comput*., vol. 19, no. 1, pp. 15–28, 2020.

[22] Y. H. Sun, M. G. Peng, Y. C. Zhou, Y. Z. Huang, and S. W. Mao, Application of machine learning in wireless networks: Key techniques and open issues, *IEEE Commun*. *Surv*. *Tutor*., vol. 21, no. 4, pp. 3072–3108, 2019.

[23] L. Lei, H. J. Xu, X. Xiong, K. Zheng, W. Xiang, and X. B. Wang, Multi-user resource control with deep reinforcement learning in IoT edge computing, arXiv preprint arXiv: 1906.07860, 2019.

[24] X. F. Chen, H. G. Zhang, C. Wu, S. W. Mao, Y. S. Ji, and M. Bennis, Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning, *IEEE Internet Things J*., vol. 6, no. 3, pp. 4005–4018, 2019.

[25] L. Huang, S. Z. Bi, and Y. J. A. Zhang, Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks, *IEEE Trans*. *Mobile Comput*., vol. 19, no. 311, pp. 2581–2593, 2020.

[26] Y. Liu, H. M. Yu, S. L. Xie, and Y. Zhang, Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks, *IEEE Trans*. *Veh*. *Technol*., vol. 68, no. 11, pp. 11 158–11 168, 2019.

[27] H. Ye, G. Y. Li, and B. H. F. Juang, Deep reinforcement learning based resource allocation for V2V communications, *IEEE Trans*. *Veh*. *Technol*., vol. 68, no. 4, pp. 3163–3173, 2019.

[28] R. P. Li, Z. F. Zhao, Q. Sun, C. L. I, C. Y. Yang, X. F. Chen, M. J. Zhao, H. G. Zhang, Deep reinforcement learning for resource management in network slicing, *IEEE Access*, vol. 6, pp. 74 429–74 441, 2018.

[29] F. Meng, P. Chen, L. N. Wu, and J. L. Cheng, Senior, power allocation in multi-user cellular networks: Deep reinforcement learning approaches, *IEEE Trans*. *Wirel*. *Commun*., vol. 19, no. 10, pp. 6255–6267, 2020.

[30] X. J. Li, J. Fang, W. Cheng, H. P. Duan, Z. Chen, and H. B. Li, Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach, *IEEE Access*, vol. 6, pp. 25 463–25 473, 2018.

[31] X. M. He, K. Wang, H. W. Huang, T. Miyazaki, Y. X. Wang, and S. Guo, Green resource allocation based on deep reinforcement learning in content-centric IoT, *IEEE Trans*. *Emerg*. *Top*. *Comput*., vol. 8, no. 3, pp. 781–796, 2020.

[32] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, On reducing IoT service delay via fog offloading, *IEEE Internet Things J*., vol. 5, no. 2, pp. 998–1010, 2018.

[33] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*, 2nd ed. Cambridge, MA, USA: MIT Press, 1998.

[34] H. Zhu, Y. Cao, X. Wei, W. Wang, T. Jiang, and S. Jin, Caching transient data for internet of things: A deep reinforcement learning approach, *IEEE Internet Things J*., vol. 6, no. 2, pp. 2074–2083, 2019.

G. M. Shafiqur Rahman received the MS degree from Beijing University of Posts and Telecommunications (BUPT), China in 2016 and currently is pursuing the PhD degree in the Key Laboratory of Universal Wireless Communications (Ministry of Education) at BUPT. His research interest includes heterogeneous networks, cloud computing based radio access networks, device-to-device communications, deep reinforcement learning, and the applications of stochastic geometry in fog radio access networks.

Tian Dang received the BS degree from Beijing University of Posts and Telecommunications, China in 2015. She is currently pursuing the PhD degree in the Key Laboratory of Universal Wireless Communications (Ministry of Education) at BUPT. Her research interests include the joint radio communication, caching, and computing resource allocation optimization in fog radio access networks.

Manzoor Ahmed received the bachelor degree with distinction from Balochistan University of Engineering and Technology, Khuzdar, Pakistan in 1996 and the master degree from Baluchistan University of Information Technology, Engineering and Management Sciences, Quetta, Pakistan in 2010. He has been pursuing the PhD degree at Beijing University of Posts and Telecommunications, Beijing, China, since 2011. He is serving in National Telecommunication Corporation and has worked in many different technical positions and received several national and international trainings since 2000. His research interests include the non-cooperative and cooperative game theoretic-based resource management in hierarchical heterogeneous networks, interference management in small cell networks, and 5G networks.