Federated Deep Reinforcement Learning for Internet of Things With Decentralized Cooperative Edge Caching

Xiaofei Wang, *Senior Member, IEEE*, Chenyang Wang, *Student Member, IEEE*, Xiuhua Li, *Member, IEEE*, Victor C. M. Leung, *Fellow, IEEE*, and Tarik Taleb, *Senior Member, IEEE*

***Abstract*—Edge caching is an emerging technology for addressing massive content access in mobile networks to support rapidly growing Internet-of-Things (IoT) services and applications. However, most current optimization-based methods lack a self-adaptive ability in dynamic environments. To tackle these challenges, current learning-based approaches are generally proposed in a centralized way. However, network resources may be overconsumed during the training and data transmission process. To address the complex and dynamic control issues, we propose a federated deep-reinforcement-learning-based cooperative edge caching (FADE) framework. FADE enables base stations (BSs) to cooperatively learn a shared predictive model by considering the first-round training parameters of the BSs as the initial input of the local training, and then uploads near-optimal local parameters to the BSs to participate in the next round of global training. Furthermore, we prove the expectation convergence of FADE. Trace-driven simulation results demonstrate the effectiveness of the proposed FADE framework on reducing the performance loss and average delay, offloading backhaul traffic, and improving the hit rate.**

***Index Terms*—Cooperative caching, deep reinforcement learning (DRL), edge caching, federated learning, hit rate, Internet of Things (IoT).**

Manuscript received December 16, 2019; revised March 8, 2020; accepted March 24, 2020. Date of publication April 9, 2020; date of current version October 9, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2101901, Grant 2018YFC0809803, Grant 2018YFF0214700, and Grant 2018YFB2100100; in part by the China NSFC under Grant 61702364 and Grant 61902044; in part by the Chongqing Research Program of Basic Research and Frontier Technology under Grant cstc2019jcyj-msxmX0589; in part by the Chinese National Engineering Laboratory for Big Data System Computing Technology; in part by the Canadian NSERC; in part by the European Unions Horizon 2020 Research and Innovation Program through the MonB5G Project under Grant 871780; in part by the Academy of Finland 6Genesis Project under Grant 318927; and in part by the Academy of Finland CSN Project under Grant 311654. This article was presented in part at the IEEE Wireless Communications and Networking Conference (WCNC), April 2019, Marrakesh, Morocco. *(Corresponding author: Xiuhua Li.)*

Xiaofei Wang and Chenyang Wang are with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China (e-mail: xiaofeiwang@tju.edu.cn; chenyangwang@tju.edu.cn).

Xiuhua Li is with the State Key Laboratory of Power Transmission Equipment and System Security and New Technology, School of Big Data and Software Engineering, and Key Laboratory of Dependable Service Computing in Cyber Physical Society, Ministry of Education, Chongqing University, Chongqing 401331, China (e-mail: lixiuhua1988@gmail.com).

Victor C. M. Leung is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China, and also with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: vleung@ieee.org).

Tarik Taleb is with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland, also with the Information Technology and Electrical Engineering, Oulu University, 90570 Oulu, Finland, and also with the Department of Computer and Information Security, Sejong University, Seoul 05006, South Korea (e-mail: tarik.taleb@aalto.fi).

Digital Object Identifier 10.1109/JIOT.2020.2986803

I. Introduction

With the rapid enhancements of wireless access technology and the Internet of Things (IoT), massive Internet services and applications are gradually migrating to mobile networks. Due to the extensive access of sensors, massive interconnections via IoT devices (e.g., smart thermostat, temperature sensors, and smartwatches) are embedded in people's daily lives. For instance, as reported in [1], the number of IoT devices will surpass ten billion by 2020, and higher quality of real-time services (e.g., heart rate monitor, step count, and outdoor video live) from these devices is required. Facing the tremendous growth in network traffic load and Quality of Service/Experience (QoS/QoE) of user demands, enormous challenges have emerged for mobile networks and the IoT [2]–[7].

In particular, due to the integration of powerful sensoring and computing functions, IoT devices are equipped with intelligent identification, behavior tracking, daily management, etc. [8]–[10]. Meanwhile, the application of short-range communication technology enables these IoT devices to form a variety of *ad hoc* network application scenarios (e.g., content airdrops and Apple edge cache service). For these scenarios, the reliable content transmission may fail to be provided due to the performance fluctuation of nearby IoT devices. Thus, it is feasible to cache the content on multiple nearby IoT devices with certain storage capabilities [11]–[13].

Moreover, edge computing has been regarded as a promising technology that can bring computation and caching services in proximity to the network edges [e.g., in base stations (BSs) and IoT devices] from the mobile network operator (MNO) or the cloud. This article considers that IoT devices are handled by people to enable operations (e.g., request or receive messages or sensoring). We consider a general cooperative edge caching-supported IoT architecture illustrated in Fig. 1. To improve the QoS/QoE and provide the best performance for IoT devices

Fig. 1. Cooperative edge caching-supported IoT architecture.

with content service requirements (e.g., system update package) from the content providers (CPs), related applications exist. For instance, Apple Inc. recently launched the Apple edge cache service [14], enabling the delivery of Apple content services directly to the equipment within CP partner networks. Caching the requested contents in edge nodes (e.g., BSs) is similar to that in IoT devices, which improves the efficiency of content access compared to excessive downloading via backhaul links [15]–[17].

Many efforts have been devoted to addressing the resource allocation issues in traffic offloading for massive IoT devices (especially for mobile devices). The studies in [15] and [18]–[20] investigated the architectures of collaborative edge caching in mobile networks. Golrezaei *et al.* [21], Woo *et al.* [22], Li *et al.* [23], and Chae *et al.* [24] optimized content access delay by collaborative caching among BSs, thereby improving the QoS of users. Other existing schemes have been proposed to design and optimize edge caching framework from the perspectives of energy consumption [25] and context awareness [26].

Recently, learning-based approaches have also been widely utilized to design and optimize edge caching [27]–[30]. For instance, deep reinforcement learning (DRL) was considered for comprehensive resource allocation, as in [31]. This approach maximizes the long-term reward of energy consumption and requires no prior knowledge of the considered networks. An assumption was made that the devices are sufficiently powerful to train the DRL agents independently. However, IoT devices can only support lightweight neural networks with small-scale data processing. In addition, most of the traditional DRL algorithms train the data in BSs or a data center by sharing the original data, leading to a large amount of network resource consumption during the data transmission process [32], [33].

To cope with the dynamic environment and ensure data-localized training for IoT devices, we aim to optimize the edge caching problem in a long term and decentralized fashion. Motivated by the aforementioned, we propose a federated deep-reinforcement-learning-based cooperative edge caching algorithm (FADE), which enables IoT devices [or user equipment (UE)][1] to cooperatively learn a shared model while keeping all the training data on the individual device. FADE is performed in a decentralized manner. First, a UE obtains the initial training model from the local BS, improves it by learning from the local data in the device, and summarizes the update. Then, only the update is sent to the BS, where all the updates from participating UEs are averaged to improve the shared model.

The main contributions of this article are summarized as follows.

1) We investigate the issue of federated DRL for IoT with decentralized cooperative edge caching. Particularly, we model the content replacement problem as a Markov decision process (MDP) and propose a federated learning framework based on double deep Q-network (DDQN) to address the problem of data sampling in the noncontinuous huge spaces.
2) We propose the FADE framework, which can enable fast training and decouple the learning process from the data stored in the cloud in a distributed-centralized way by keeping the data training in the local UEs. In addition, we prove that FADE is L-smooth and μ-strong and derives its expectation convergence.
3) Trace-driven simulation results show that compared with existing methods, the proposed FADE framework can effectively reduce the performance loss and average delay, offload backhaul traffic, and improve the hit rate.

The remainder of this article is organized as follows. Section II summarizes the previous work. Section III introduces the system model and formulates the problem. The framework design of the proposed FADE is presented in Section IV. Trace-driven simulation results evaluate the effectiveness of the proposed framework in Section V. Finally, Section VI concludes this article.

II. Related Work

For caching-supported IoT networks, existing studies can be divided into the following two categories.

The first category is to utilize traditional methods based on convex optimization or probability modeling to address the content placement problem for IoT networks. For instance, Li *et al.* [34] focused on maximizing traffic offloading and reducing the system costs by designing a hierarchical edge caching strategy. The work in [35] considered the problem of the optimal bandwidth allocation and minimized the average transmission delay by deploying a greedy algorithm of cooperative edge caching. Vural *et al.* [36] proposed a content replacement strategy with multiattribute joint optimization in terms of content lifetime, the request rate of users, and the hop information between the source user and the destination. To efficiently optimize the caching resources of IoT devices, Zhang and Liu [37] exploited probabilistic caching in heterogeneous IoT networks in order to improve traffic offloading. The content caching problem in IoT networks requires continuous optimization. In other words, various attributes (e.g., content popularity and user mobility) in IoT networks are constantly evolving. However, these strategies are often difficult to

[1]In this article, we use UE to denote the IoT device hereafter.

adapt to dynamic environments and hard to deploy in practice due to the global information required.

The second classification is based on learning algorithms such as machine learning/deep learning, which learns key attribute features (e.g., user request behavior, content popularity, and user mobility distribution) in the network to optimize the content caching strategy. The literature of [38] and [39] showed that reinforcement learning (RL) has great potential for the utilization in the scheme design of content caching in BSs. Specifically, Gu *et al.* [38] proposed a cache replacement strategy based on Q-learning to reduce traffic load in future cellular networks, and RL was also employed for cache placement [39] by using a multiarmed bandit (MAB) algorithm. Chen *et al.* [40] proposed a popularity-based caching strategy for IoT networks by deploying deep neural networks to predict the near future popularity of the IoT data. However, most centralized learning algorithms are prone to posing high cache diversity and storage utilization, which lead to excessive network communication resource consumption. On the other hand, the distributed learning method requires much cache and action space, which will also cause the above problems.

III. System Model and Problem Formulation

In this section, we first introduce the topology of cooperative edge caching-supported IoT systems and then discuss the delay model. Next, the cache replacement model is presented. Finally, we formulate the optimization problem of edge caching for IoT systems. Some key parameters are listed in Table I.

TABLE I
KEY PARAMETERS

Notation	Meaning
F	Total number of contents
N_b	Number of BSs
N_u	Number of UEs
C	Cache size
$\mathcal{F}$	Library of popular contents
D_f	Size of content f
$(P_f)_{F\times 1}$	Global popularity of content f
$\mathcal{M}$	Wireless channels
$d_n^c,\ d^P,\ d_b$	Transmissions delay between BS n to UE, BS and CPs, BS and BS, respectively
$v_{u,n}$	Downlink data rate between BS n and UE u
$P_{u,f}$	Preference of UE u for content f
$\mathbf{s}_{i,n}^{\mathrm{c}}$	The set of content caching state in BS n with each decision epoch i
$\boldsymbol{\chi}_i$	The state of BS during each decision epoch i
$\boldsymbol{\Phi}(\boldsymbol{\chi}_i)=\{\mathbf{a}_i^{\mathrm{local}},\mathbf{a}_i^{\mathrm{co-BS}},a_i^{\mathrm{CP}}\}$	System action with the state $\boldsymbol{\chi}_i$
$\mathcal{R}(\boldsymbol{\chi},\boldsymbol{\Phi})$	Reward function
$Q(\boldsymbol{\chi},\boldsymbol{\Phi};\boldsymbol{w}_i)$ and $\hat{Q}(\boldsymbol{\chi},\boldsymbol{\Phi};\widehat{\boldsymbol{w}}_{\boldsymbol{i}})$	Q values of mainNet and TargetNet, respectively
$L(\boldsymbol{w}_i)$	Loss function of Double DQN
$F_j(w)$	Loss function of Federated DRL

A. Topology of Cooperative Edge Caching-Supported IoT Systems

The topology of a cooperative edge caching-supported IoT network is illustrated in Fig. 2. Particularly, in the considered IoT network, a high number of geographically distributed UEs (e.g., smartwatches) are served by some BSs via wireless cellular links, and BSs are connected via wired optical cables. Here, each BS is deployed with an edge server for computation and caching, and thus, each BS can cache various contents to satisfy the demands of content services for IoT devices. As a result, UEs can fetch their requested contents either locally from the edge servers or directly by downloading the contents from CPs (in the cloud) to the BSs via the MNO core.

Consider a hierarchical IoT network in which $\mathcal{N}=\{1,2,\ldots,N_b\}$ fully connected BSs with a finite cache size of C and $\mathcal{U}=\{1,2,\ldots,N_u\}$ UEs are distributed in the service area. Denote $\mathcal{F}=\{1,2,\ldots,F\}$ as a library of contents that are supported by the CPs which all UEs may access in the system for a relatively long time. Denote $D_f, f\in\mathcal{F}$ as the size of content f.[2]

Let $(P_f)_{F\times 1}$ be the global popularity, which indicates the probability distribution of content f requested by all UEs in the network, and let p_{nf} be the local popularity of content f under BS n. We consider that $P_f=\sum_{n\in\mathcal{N}}p_{nf}$, and $(P_f)_{F\times 1}$ follows the Mandelbrot–Zipf (MZipf) distribution[3] [43] as

$$P_f=\frac{\left(I_f+\tau\right)^{-\beta}}{\sum_{i\in\mathcal{F}}(I_i+\tau)^{-\beta}}\quad\forall f\in\mathcal{F} \tag{1}$$

where I_f is the rank of content f in descending order of content popularity and τ and β denote the plateau factor and skewness factor, respectively.

B. Delay Model

We consider the content access delay for a UE as the round-trip time to receive the requested content. From Fig. 2, d^b denotes the transmission delay of the BSs' cooperation and d^p is the delay between the BS and CPs. The wireless transmission delay d^{c} can be regarded as the period of a UE obtaining the content from the local BS. Considering $\mathcal{M}=\{1,2,\ldots,M\}$, wireless channels are deployed, and $a_u\in\mathcal{M}$ is the channel that is assigned to UE u by BS. Similar to [44], we can obtain the downlink data rate between BS n and UE u as follows:

$$v_{u,n}=B\log_2\left(1+\frac{q_u g_{u,n}}{\sigma^2+\sum_{v\in\mathcal{U}\setminus\{u\}:a_v=a_u}q_v g_{v,n}}\right) \tag{2}$$

where B denotes the channel bandwidth, σ^2 represents the background noise power, q_u is the power consumption of BS n transmission to UE u, and the channel gain $g_{u,n}$ can be determined by the distance $l_{u,n}$ between BS n and UE u.

[2]We consider $\mathcal{D}=\{D_1, D_2,\ldots,D_f,\ldots,D_{\mathcal{F}}\}$ to be the size of local data sets as well, which will be introduced in Section IV-B.

[3]Note that it is also widely used in mobile IoT scenarios [41], [42].

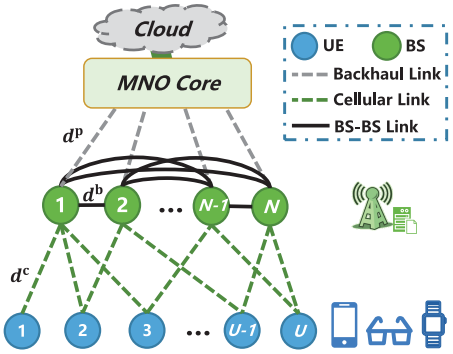


Fig. 2. Topology of the caching-supported IoT system.

Consider that $P_{u,f} = r_{u,f}/R_u, f \in \mathcal{F}$, reflects the preference of UE u for content f, and $\sum_{f\in\mathcal{F}} P_{u,f} = 1$, where $r_{u,f}$ is the number of UE u requests for content f and R_u is the total request number of UE u in the network. Furthermore, we define $P_{u,nf} = P_{u,f}a_{un}$ as the local UE preference for content f under BS n, a_{un} is the association probability of UE u and BS n. Thus, the wireless transmission delay d_n^c can be obtained as

$$d_n^c = \sum_{u\in\mathcal{U}}\sum_{f\in\mathcal{F}} P_{u,nf}\frac{D_f}{v_{u,n}}. \tag{3}$$

C. Cache Replacement Model

We model the process of content cache replacement in a BS as an MDP [45]. The state of the cache and request, system action, and feedback reward are given as follows.

1) Cache and Request States: During each decision epoch i, we define the content cache state as $\boldsymbol{s}_{i,n}^c := \{s_{i,n,f}^c\}, n \in \mathcal{N}$, $f \in \mathcal{F}$. Here, $s_{i,n,f}^c$ is the cache state in BS n for content $f \in \mathcal{F}$, and $s_{i,n,f}^c = 1$ represents that BS n caches the content f, $s_{i,n,f}^c = 0$ otherwise. Furthermore, we use $\boldsymbol{s}_{i,u}^r := \{s_{i,u,f}^r\}, u \in \mathcal{U}, f \in \mathcal{F}$ to denote the request state from UE u, where $s_{i,u,f}^r$ is the request state of u for content f. Similarly, $s_{i,u,f}^r = 1$ means that UE u sends a request for content f, $s_{i,u,f}^r = 0$ is the opposite. Thus, we derive the cache and request state during each decision epoch i as

$$\boldsymbol{\chi}_i = \left(\boldsymbol{s}_{i,u}^r, \boldsymbol{s}_{i,n}^c\right) \in \mathcal{X} \stackrel{\text{def}}{=} \{1, 2, \ldots, F\} \times \left\{\times_{f\in\mathcal{F}}\mathbb{P}\right\}. \tag{4}$$

2) System Action: Adapting to the continuous changes in the dynamic environment, BSs can choose which contents to be replaced and decide where the requests are processed (via local BS, BS cooperation, or CPs). We denote $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ as the system action with the state $\boldsymbol{\chi}_i$, and the action space for all cooperative BSs is defined as

$$\boldsymbol{\Phi}(\boldsymbol{\chi}_i) = \{\boldsymbol{a}_i^{\text{local}}, \boldsymbol{a}_i^{\text{co-BS}}, a_i^{\text{CP}}\} \tag{5}$$

where there exist three different types of system action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$, shown as follows.

a) Local processing action: We denote $\boldsymbol{a}_i^{\text{local}} \stackrel{\text{def}}{=} [a_{i,0}^{\text{local}}, a_{i,1}^{\text{local}}, \ldots, a_{i,F}^{\text{local}}]$ as the local processing action when the cache state controlled by the local BS is available, where $a_{i,f}^{\text{local}} \in \{0, 1\}, f \in \mathcal{F}$, and $a_{i,f}^{\text{local}} = 1$ indicates that content f needs to be replaced by the current requested content, while $a_{i,f}^{\text{local}} = 0$ is the opposite. In this case, the content request is processed locally.

b) Cooperation processing action: If the requested content f is not cached in the local BS, the UE's request needs to be routed to its neighbor BS. We define $\boldsymbol{a}_i^{\text{co-BS}} \stackrel{\text{def}}{=} [a_{i,1}^{\text{co-BS}}, \ldots, a_{i,N}^{\text{co-BS}}]$ as the cooperation processing action, where $a_{i,n}^{\text{co-BS}} \in \{0, 1\}$, and $a_{i,n}^{\text{co-BS}} = 1$ denotes that BS n is selected to address the current UE's request.

c) Remote processing action: If the UE cannot obtain the requested content f from either the local BS or its neighboring BSs, the local BS decides whether to forward the request to CPs, denoted as $a_i^{\text{CP}} \in \{0, 1\}$, where $a_i^{\text{CP}} = 1$ represents the request is handled by CPs. In this case, the UE should obtain the requested content f directly from the remote CPs.

3) System Reward: When the local BS takes action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ upon state $\boldsymbol{\chi}_i$, it will obtain the feedback reward. To satisfy the QoS of UEs, our goal is to minimize the average content access latency of the system.

Because of fiber communication, d_n^c may be far greater than d^b and d^P. Based on (3), to achieve the maximum system reward and guarantee the objective of minimizing the average content access delay, we use the negative exponential function to normalize the reward function. Thus, we derive the reward function as

$$R_n(\chi_i, \Phi(\chi_i)) = \begin{cases} p_{nf}e^{-\xi_1 d_n^c}, & \text{Cellular Service} \\ p_{nf}e^{-\left(\xi_1 d_n^c + \xi_2 d^b\right)}, & \text{BS} - \text{BS Cooperation} \\ p_{nf}e^{-\left(\xi_1 d_n^c + \xi_3 d^P\right)}, & \text{Backhaul Service} \end{cases} \tag{6}$$

where $\xi_1 + \xi_2 + \xi_3 = 1, \xi_1 \ll \xi_2 < \xi_3$, and $p_{nf}e^{-\xi_1 d_n^c}$ is the reward that a UE obtains content f from BS only via cellular service; $p_{nf}e^{-(\xi_1 d_n^c + \xi_2 d^b)}$ means the UE is served by the BS–BS cooperation. When a UE has to be served by the MNO core via backhaul links, the reward is $p_{nf}e^{-(\xi_1 d_n^c + \xi_3 d^P)}$.

D. Problem Formulation

Based on (6), our optimization objective is to maximize the expected long-term reward based on an arbitrary initial state $\boldsymbol{\chi}_1$ as

$$R^{\text{long}} = \max \mathbb{E}_{\boldsymbol{\Phi}}\left[\lim_{I\to\infty}\frac{1}{I}\sum_{i=1}^{I}\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))|\boldsymbol{\chi}_1 = \boldsymbol{\chi}\right] \tag{7}$$

where $\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$ is the sum of $R_n(\chi_i, \Phi(\chi_i))$.

Moreover, a single-agent infinite-horizon MDP with a discounted utility (8) can be generally utilized to approximate the expected infinite-horizon undiscounted value, especially when $\gamma \in [0, 1)$ approaches 1

$$V(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathbb{E}_{\Phi}\left[\sum_{i=1}^{\infty}(\gamma)^{i-1}\cdot\mathcal{R}\left(\boldsymbol{\chi}_i, \boldsymbol{\Phi}\left(\boldsymbol{\chi}_i\right)\right)|\boldsymbol{\chi}_1 = \boldsymbol{\chi}\right]. \tag{8}$$

Each BS is expected to learn an optimal control policy, denoted as Φ^*, for maximizing $V(\boldsymbol{\chi}, \boldsymbol{\Phi})$ with a random initial state $\boldsymbol{\chi}$. Then, we can describe the optimal control policy Φ^* as

$$\Phi^* = \underset{\boldsymbol{\Phi}}{\text{argmax}}\, V(\boldsymbol{\chi}, \boldsymbol{\Phi}) \quad \forall \boldsymbol{\chi} \in \mathcal{X}. \tag{9}$$

Thus, we formulate the corresponding edge caching problem to maximize the value function and obtain the optimal control policy, which can be expressed as

$$\begin{aligned} \max \quad & V(\boldsymbol{\chi}, \boldsymbol{\Phi}) \\ \text{s.t.} \quad & \sum_{f \in \mathcal{F}} s^{c}_{i,n,f} D_f \le C, \quad n \in \mathcal{N} \\ & s^{c}_{i,n,f} \in \{0,1\}, \quad n \in \mathcal{N} \text{ and } f \in \mathcal{F} \\ & s^{r}_{i,u,f} \in \{0,1\}, \quad u \in \mathcal{U} \text{ and } f \in \mathcal{F} \end{aligned} \tag{10}$$

where $i = \{1, 2, 3, \ldots\}$ is the decision epoch index and all the constraints are used to promise the available cache size at each BS, and (10) subjects to the BS cache size C.

IV. Framework Design of FADE

In this section, we first formulate the DRL pretraining process on the local BS and analyze the computation complexity. Furthermore, the federated DRL-based edge caching algorithm is proposed. Finally, we carry out the theoretical convergence analysis of the proposed algorithm.

A. Pretraining Process BS

In our federated learning-based cooperative edge caching architecture, the local BS first carries out the corresponding action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ based on the current state $\boldsymbol{\chi}_i$. Then, the current feedback reward $\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$ is obtained. Finally, the former system state $\boldsymbol{\chi}_i$ transitions into the next new one $\boldsymbol{\chi}_{i+1}$. The pretrained parameters are sent to each UE as the initialization input of all the participating UEs' federated learning process.

The optimal value function $V(\boldsymbol{\chi})$ can be obtained as follows based on the Bellman equation [45]:

$$V(\boldsymbol{\chi}) = \max_{\boldsymbol{\Phi}} \left\{ \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \Pr\{\boldsymbol{\chi}' | \boldsymbol{\chi}, \boldsymbol{\Phi}\} \cdot V(\boldsymbol{\chi}') \right\}. \tag{11}$$

Rewrite the right-hand side of (11) in the form of the Q-function

$$Q(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \Pr\{\boldsymbol{\chi}' | \boldsymbol{\chi}, \boldsymbol{\Phi}\} \cdot V(\boldsymbol{\chi}'). \tag{12}$$

The optimal state value function $V(\boldsymbol{\chi})$ can be simply abstracted as

$$V(\boldsymbol{\chi}) = \max_{\boldsymbol{\Phi}} Q(\boldsymbol{\chi}, \boldsymbol{\Phi}). \tag{13}$$

Incorporating (13), we rewrite (12) as

$$Q(\boldsymbol{\chi}, \boldsymbol{\Phi}) = \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \sum_{\boldsymbol{\chi}'} \Pr\{\boldsymbol{\chi}' | \boldsymbol{\chi}, \boldsymbol{\Phi}\} \cdot \max_{\boldsymbol{\Phi}'} Q(\boldsymbol{\chi}', \boldsymbol{\Phi}'). \tag{14}$$

Finally, we can obtain the Q-function iteration formula as

$$\begin{aligned} Q^{i+1}(\chi, \Phi) = {} & Q^{i}(\chi, \Phi) \\ & + \alpha^{i} \cdot \left(\mathcal{R}(\chi, \Phi) + \gamma \cdot \max_{\Phi'} Q^{i}(\chi', \Phi') \right. \\ & \left. - Q^{i}(\chi, \Phi) \right) \end{aligned} \tag{15}$$

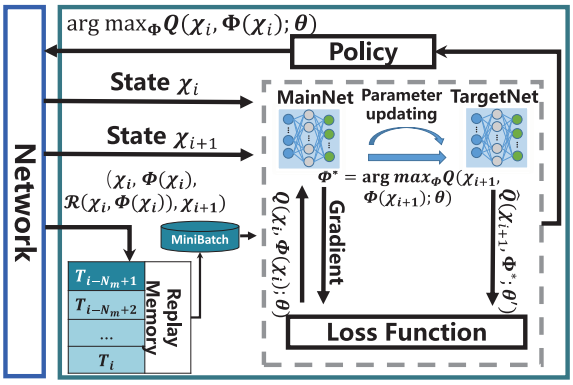


Fig. 3. Caching replacement process of double DQN.

where $\alpha^i \in [0, 1)$ is the learning rate. The current state $\boldsymbol{\chi}_i$ transitions into the next state $\boldsymbol{\chi}_{i+1}$ after local BS taking the system action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$ and obtaining the immediate reward $R(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$.

We use double DQN [46] to deploy the pretraining process in local BSs. The caching replacement process of double DQN is shown in Fig. 3. By updating the parameters $\boldsymbol{w}_i$ of the multiple layer perceptron (MLP), we can obtain the approximate optimal Q-value according to the following equation:

$$Q(\boldsymbol{\chi}, \boldsymbol{\Phi}) \approx Q((\boldsymbol{\chi}, \boldsymbol{\Phi}); \boldsymbol{w}_i). \tag{16}$$

There is an experience replay pool (namely, transition memory) with a finite size N_m in each training agent to store the experienced transitions, denoted as $\mathcal{M} = \{T_{i-N_m+1}, \ldots, T_i\}$, where $T_i = (\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i), \mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)), \boldsymbol{\chi}_{i+1})$. $\mathcal{M}$ is updated by the most recent experienced transitions, and the agent preserves $Q(\boldsymbol{\chi}, \boldsymbol{\Phi}; \boldsymbol{w}_i)$ and $\hat{Q}(\boldsymbol{\chi}, \boldsymbol{\Phi}; \widehat{\boldsymbol{w}}_i)$. The Q network (MainNet) is used to select a system action and the $\hat{Q}$ network (TargetNet) is utilized for evaluating it. Note that the weight parameters $\widehat{\boldsymbol{w}}_i$ in the $\hat{Q}$ network periodically update along with $\boldsymbol{w}_i$ in the Q network.

During the whole process of system training, the agent randomly selects a minibatch $\tilde{\mathcal{M}}$ from transition memory $\mathcal{M}$. Then, it trains the Q network by minimizing the loss function at each epoch as

$$\begin{aligned} L(\boldsymbol{w}_i) = {} & \mathbb{E}_{(\boldsymbol{\chi}, \boldsymbol{\Phi}, \mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}), \boldsymbol{\chi}') \in \tilde{\mathcal{M}}_i} \\ & \times \left[\left(\mathcal{R}(\boldsymbol{\chi}, \boldsymbol{\Phi}) + \gamma \cdot \hat{Q}\left(\boldsymbol{\chi}, \arg\max_{\boldsymbol{\Phi}'} Q(\boldsymbol{\chi}', \boldsymbol{\Phi}'; \boldsymbol{w}_i); \widehat{\boldsymbol{w}}_i \right) \right. \right. \\ & \left. \left. - Q(\boldsymbol{\chi}, \boldsymbol{\Phi}; \boldsymbol{w}_i) \right)^2 \right]. \end{aligned} \tag{17}$$

Moreover, we can obtain the gradient updates of w_i by $\nabla_{w_i} L(w_i)$ as follows:

$$\begin{aligned} \nabla_{w_i} L(w_i) = {} & \mathbb{E}_{(\chi, \Phi, R(\chi, \Phi), \chi') \in M_i} \\ & \times \left[\left(R(\chi, \Phi) + \gamma \right. \right. \\ & \times \hat{Q}\left(\chi, \arg\max_{\Phi'} Q(\chi', \Phi'; w_i); w_i' \right) \\ & \left. \left. - Q(\chi, \Phi; w_i) \right) \cdot \nabla_{w_i} Q(\chi, \Phi; w_i) \right]. \end{aligned} \tag{18}$$

Algorithm 1 Pretraining Process on Local BS

Initialization: (Offline Training Process)

Construct the reward function $\mathcal{R}$.
Initialize transition memory $\mathcal{M}$ with capacity N_m.
Initialize the Q network by a random weight $\boldsymbol{w}$.
Initialize the $\hat{Q}$ network by $\widehat{\boldsymbol{w}}_{\boldsymbol{i}} = \boldsymbol{w}$.
Pretraining the main and target network with $< \boldsymbol{\chi}_i, \Phi(\boldsymbol{\chi}_i) >$ and the corresponding $Q(\boldsymbol{\chi}_i, \Phi(\boldsymbol{\chi}_i); \boldsymbol{w}_i)$.

Iteration: (Online Caching Process)

1: **for** the episode $i = 1$ **to** I **do**
2: BS n receive a request $s^{\mathrm{r}}_{i,u,f}$ from UE u for content f.
3: **if** The cache state of requested content $s^{\mathrm{c}}_{i,n,f} = 1$ **then**
4: End episode.
5: **else**
6: **if** The BS storage $\mathcal{C}$ is not full **then**
7: Cache the requested content f.
8: Update the caching state $\boldsymbol{\chi}_i$ in BS n and end episode.
9: **end if**
10: Receive the caching state $\boldsymbol{\chi}_i$.
11: Select action $\arg\max_{\Phi(\boldsymbol{\chi}_i)} Q(\boldsymbol{\chi}_i, \Phi(\boldsymbol{\chi}_i); \boldsymbol{w}_i)$.
12: Execute action $\Phi(\boldsymbol{\chi}_i)$.
13: Obtain immediate reward $\mathcal{R}(\boldsymbol{\chi}_i, \Phi(\boldsymbol{\chi}_i))$.
14: Get the new state $\boldsymbol{\chi}_{i+1}$.
15: Construct $T_i = (\boldsymbol{\chi}_i, \Phi(\boldsymbol{\chi}_i), \mathcal{R}(\boldsymbol{\chi}_i, \Phi(\boldsymbol{\chi}_i)), \boldsymbol{\chi}_{i+1})$.
16: Save the transition T_i to $\mathcal{M}$.
17: Randomly select a mini-batch of transition $\tilde{M}_i \in \mathcal{M}$.
18: Update $\boldsymbol{\theta}_i$ by minimizing the gradient as in (17).
19: Update $\boldsymbol{w}_i$ of Q network with $\nabla_{w_i} L(w_i)$.
20: Update $\boldsymbol{\theta}_i$ of Q network with the gradient.
21: Update $\widehat{\boldsymbol{w}}_{\boldsymbol{i}}$ of $\hat{Q}$ network periodically.
22: Update the caching state $\boldsymbol{\chi}_i$.
23: **end if**
24: **end for**

The pretraining process on the local BS is shown in Algorithm 1 and has two main procedures.

1) *Procedure 1 (Offline Training Process):* Initialize the preliminaries of the double DQN training process in the aspects of experience replay memory $\mathcal{M}$, random weights $\boldsymbol{w}$ and $\widehat{\boldsymbol{w}}_{\boldsymbol{i}}$ selection of the main Q network, and target $\widehat{Q}$ network, respectively. Then, pretraining the main and target network with $<\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)>$ and the corresponding $Q(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i); \boldsymbol{w}_i)$.
2) *Procedure 2 (Online Caching Process):* On the premise that there is no requested content f in BS n and the BS storage $\mathcal{C}$ is full (shown as lines 2–9), execute double DQN to train the caching process and update all the parameters (shown as lines 10–20).

Computation Complexity Analysis: In terms of the complexity of DRL, we consider mainly two aspects, namely, transitions and backpropagation. Suppose that there are K transitions into the replay memory; we can obtain the complexity of $\mathcal{O}(K)$. Let a and b denote the numbers of layers and transitions in each layer, respectively. It takes $\mathcal{O}(N_t abk)$ time using backpropagation and gradient descent to train the parameters.

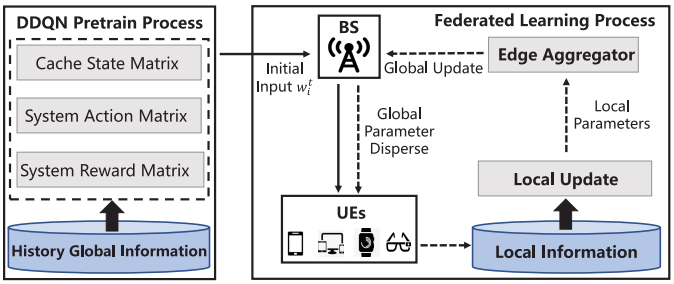


Fig. 4. Overall workflow of the proposed FADE.

Here, N_t is the number of transitions randomly sampled and k denotes the number of iterations. Specifically, the replay memory stores K transitions for which the space complexity is $\mathcal{O}(K)$ and has the space complexity of $\mathcal{O}(ab)$ for dealing with the storage issue of the parameters of DDQN.

Similar to related studies [40], [48], the complexity analysis proves that our proposed algorithm is sufficiently lightweight for IoT devices and easy to deploy.

B. FADE: Federated DRL-Based Edge Caching Algorithm

As mentioned above, DRL can find the optimal strategy dynamically and efficiently. However, it also needs many computing resources. Therefore, the deployment of the DRL agent should be carefully considered.

On the one hand, if the DRL agent does the training, it has three disadvantages as follows.

1) It takes a long time even to train each agent well.
2) It may endanger sensitive data, especially in commercial and industrial scenarios.
3) Although training data can be transformed to protect privacy, the received agent data are less relevant and less targeted among specific UEs.

On the other hand, if we deploy the distributed data training, there are still two shortcomings as follows.

1) Training each DRL agent from scratch would take a long time or even impossible to converge.
2) Individual training by a separate DRL agent will result in an additional waste of energy.

Motivated by the aforementioned reasons, we further propose FADE, a federated DRL framework based on previous DDQN solutions to build a high-quality decentralized model. Federated agents collaboratively learn a shared predictive model, while all training data remain on the individual IoT device (e.g., smartwatches), decoupling machine learning from the data stored in the cloud.

The workflow of FADE is shown in Fig. 4. The BS first disperses the initial input parameters to all the UEs produced by the pretraining process. Then, UE uploads the near-optimal local parameters to the BS to participate in the next round of global training. Repeatedly, the BS aggregates all the updated local parameters, and the improved global model is continuously dispersed to local agents. The detailed mechanism of the federated learning process is shown in Fig. 5. After the dispersion of the initial parameters w^t, each UE computes the local update w_j^{t+1} according to (23). The edge aggregator (i.e.,

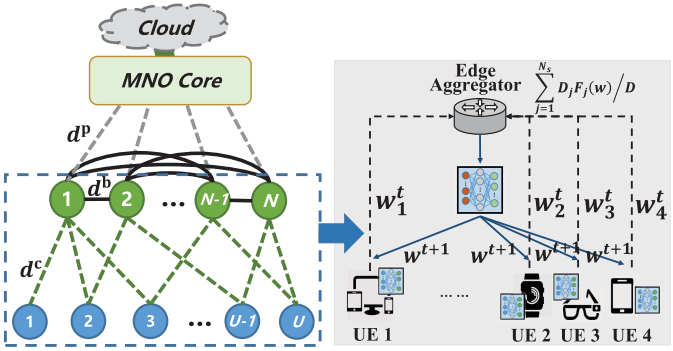


Fig. 5. Mechanism of federated learning.

BS) then obtains the global loss function from the collected parameters by (21). This process iterates until it converges.

In the following, we formally describe the federated DRL framework. We consider a wireless system with a BS and N_s UEs with the local data sets $\mathcal{D} = \{D_1, D_2, \ldots, D_j, \ldots, D_s\}$. In the learning problem, the task is to find the objective parameters w with the loss function $f(w)$. Some well-known loss functions are $f_i(w) = (1/2)\min_k \|x_i - w_k\|^2$, where $w{:=}[w_1, w_2, \ldots, w_k]$ for K-means and $f_i(w) = (1/2)\|y_i - w^T x_i\|^2$, $y_i \in \mathbb{R}$ for linear regression, as well as $f_i(w) = (\lambda/2)\|w\|^2 + (1/2)\max\{0, 1 - y_i w^T x_i\}^2$ (λ is const.) for the support vector machine. For each local data set D_j at UE j, the loss function is

$$F_j(w) := \frac{1}{D_j} \sum_{j=1}^{N_s} f_j(w) \tag{19}$$

and the local problem is

$$w_j^t = \arg \min_{w_j \in \mathbb{R}^d} F_j\left(w_j | w_j^{t-1}\right). \tag{20}$$

The global loss function is defined as

$$F(w) := \frac{\sum_{j \in D} f_j(w)}{D} = \frac{\sum_{j=1}^{N_s} D_j F_j(w)}{D} \tag{21}$$

where $D := \sum_{j=1}^{N_s} D_j$ and the learning problem is to find

$$w^* = F(w). \tag{22}$$

It is impossible to expect a general solution of (22) due to the inherent complexity of the local problem [49]. Thus, a distributed algorithm is needed to solve the problem (22).

1) Gradient Descent Algorithm: We present a general gradient descent algorithm to solve the learning problem (22), which is widely used in some work (e.g., [50]). For each UE j, w_j^t are the local parameters and t is the iteration index, where $t = 1, 2, 3, \ldots, T$. The process of the gradient descent algorithm is shown in Algorithm 2.

First, the initialization values of local parameters are assigned by pretrained parameters from Algorithm 1 at $t = 0$. For $t > 0$, each UE j computes its parameters w_j^{t+1} according to the update rule as follows:

$$w_j^{t+1} = w_j^t - \eta \nabla F_j\left(w^t\right) \tag{23}$$

Algorithm 2 Gradient Descent Algorithm for FADE

Initialization:

$w_j^0 = w_i$;
Pretrained parameters w_i^t from Algorithm 1;
Number of iterations T;
Step size η.

Iteration:

for $t = 1, 2, 3..., T$ **do**
  for each UE j **do**
    Compute its local update.
    $w_j^{t+1} = w_j^t - \eta \nabla F_j(w^t.)$
  end for
  Update the global model.
  $w^{t+1} = \frac{\sum_{j=1}^{N_s} D_j w^{t+1}}{D}$.
end for

where $\eta \geq 0$ is the gradient step size. After T iterations, the global parameters w^{t+1} are updated at BS, defined as

$$w^{t+1} = \frac{\sum_{j=1}^{N_s} D_j w^{t+1}}{D}. \tag{24}$$

The updated global parameters w^{t+1} are used to the next-round training of DRL in the local BS. In the following section, we will present the convergence analysis of problem (22).

C. Convergence Analysis

Denote w_* as the optimal solution. Similar to [51], we have the following assumptions.

Assumption 1: For all i:

1) $f_i(w)$ is convex;
2) $f_i(w)$ is L-smooth, i.e., $f_i(w') \leq f_i(w) + \nabla f_i(w) \cdot (w' - w) + (L/2)\|w - w'\|^2$, for $\forall$ w and w'.

Assumption 1 guarantees the feasibility of the linear regression and the update rule of federated learning. Thus, we have the following lemma.

Lemma 1: $f(w)$ is convex and L-smooth.

Proof: Please see Appendix A and the triangle inequality. ■

Theorem 1: Considering that $f(w)$ is L-smooth and μ-strongly, let $\eta_t = 1/L$ and $w_* = \text{argmin}_w f(w)$; thus, we have

$$\|w_t - w_*\| \leq \left(1 - \frac{\mu}{L}\right)^t \|w_1 - w_*\| \tag{25}$$

where $O(\varpi) = (L/\mu)\log(\|w_1 - w_*\|/\varpi)$ is denoted as the *gradient dispersion*, which is used to illustrate how the parameters w_i are distributed in each user.

Proof: Please see Appendix B. ■

We have the convergence in expectations

$$\left[f(w_t) - f(w_*)\right] \leq \varpi^t \left[\Delta^t(f(w_*))\right]. \tag{26}$$

Thus, $f(w)$ is proven to be bounded where $\Delta^t(f(w_*)) = f(w_1) - f(w_*)$.

We use the aforementioned theoretical analysis and results to design the FADE algorithm, as shown in Algorithm 3. Suppose that the BS initiates the DRL learning process, w_i^t is

Algorithm 3 FADE: Federated DRL-Based Edge Caching

Initialization:

$w_j^t = w_i^t$;
Pretrained parameters w_i^t from Algorithm 1;
Number of iterations T;
Step size η.

Iteration:

for $t = 1, 2, 3..., T$ **do**
2: **for** each UE j **do**
Compute its local update.
4: Set $w_j^{t+1} = w_j^t - \eta \nabla F_j(w^t)$
Estimate the convergence according to (26).
6: Return w_j^{t+1}.
end for
8: Send w_j^{t+1} to local BS n.
for each BS n **do**
10: Receive w_j^{t+1} from each UE j
Update global model according to:
$w^{t+1} = \frac{\sum_{j=1}^{N_s} D_j w^{t+1}}{D}$.
12: Input the global model:
$w_i^{t+1} = w^{t+1}$.
end for
14: BS n disperses w_i^{t+1} to UEs.
end for

the pretrained parameters from Algorithm 1, which is assigned by the local BS n. This procedure is responsible for data-localized training on each UE j, and the training process begins using a local update according to (23), when UE j receives the parameters w_i^t, and estimates the updated parameters w_j^{t+1} by (26) until it converges (lines 2–6). Then, the updated parameters are sent back to the local BS n for aggregation (line 8).

After receiving the parameters from all the local UEs, local BS n aggregates all the distributed parameters (lines 10–12) and then distributes the updated global parameters to all the participating UEs (line 14).

The core idea of FADE is to federate UEs to collaboratively train the parameters and accelerate the training process. FADE consists mainly of two parts: 1) distributed training procedure at UEs and 2) aggregation computation at BS.

V. Trace-Driven Simulation Results

A. Simulation Settings

In this section, we evaluate the proposed FADE algorithm in terms of network performance. For simulation purposes, we consider four BSs, each of which has the maximum coverage of a circle with a radius of 250 m. In addition, the channel gain is modeled as $g_{u,n} = 30.6 + 36.7 \log_{10} l_{u,n}$ dB. Each BS has 20 channels, the channel bandwidth is 20 MHz [47], and the transmit power of the BS is 40 W. The double DQN consists of a fully connected feedforward neural network with one mid-layer consisting of 200 neurons, which is used to construct the Q network and $\hat{Q}$ network. The values of some key parameters are given in Table II.

TABLE II
Parameter Values

Parameter	Value	Description
F	10,000-100,000	Content number
B	20 MHz	Channel bandwidth
σ^2	-95 dBm	Noise power
d^{b}	20 ms	Delay of BS cooperation
d^{p}	200 ms	Delay of BS-CP
$\mathcal{M}$	5000	Capacity of replay memory
D_f	(0,8] Mbit	Content size
$\mathcal{M}_i$	200	Size of minibatch
γ	0.9	Reward decay
ϵ	0.1	State transition probability
α	0.05	Learning rate
ϕ	250	The period of replacing target Q network

To implement the experiments more practically, we obtain the global and local content popularity from the real-world data sets. We use a large-scale offline MSN real-world data set derived from an application *Xender*. Xender is widely used in India to share content. The dates of the experimental data we used were from 01/08/2016 to 31/08/2016, including 450 and 786 trajectories of UEs. Over 153, 482 files were shared, and the number of requests was 271, 785, and 952 [34].

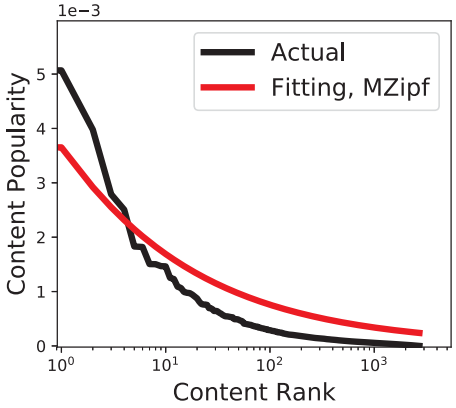
As shown in Fig. 6(a), we obtain the plateau factor $\tau = -0.88$ and skewness factor $\beta = 0.35$ by fitting the content popularity of the experimental traces with the MZipf distribution. The parameters are used for the centralized DRL simulations. Due to the homophily and locality of content popularity [18], considering the distribution of differential scenario properties (e.g., user numbers/user requests, mobility, content popularity, etc.), we include different local content popularity for four local BSs by various parameters in Fig. 6(b).

B. Evaluation Results

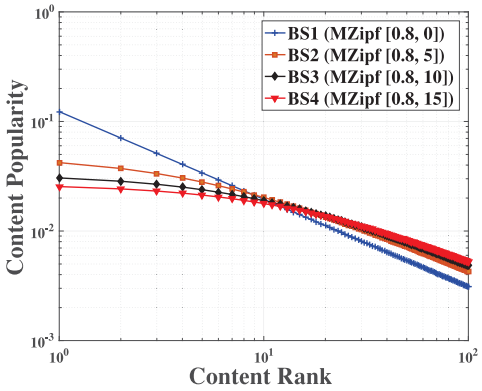
The loss function reflects the gap between the estimated values of the proposed model and the actuality. The lower loss performance means better efficiency and fast convergence of the proposed model. In this section, we first present the loss function performance between the traditional DRL and FADE. Then, the statistics of the system simulation traffic offloading are demonstrated. Furthermore, to show the efficiency of the proposed algorithm, we compare the network performance in terms of the average content access delay of UEs, the content request hit rate, and the system backhaul traffic offload.

To evaluate the proposed FADE framework, some state-of-the-art caching schemes are used for comparison such as follows.

1) *Least Recently Used (LRU):* The LRU content will be replaced first.
2) *Least Frequently Used (LFU):* The LFU content will be replaced first.
3) *First-In–First-Out (FIFO):* The oldest content will be replaced first.
4) *Oracle* [52]*:* This algorithm requires the complete knowledge of UEs, which is conducted as the baseline of best network performance.



(a)



(b)

Fig. 6. (a) Global and (b) local content popularity.

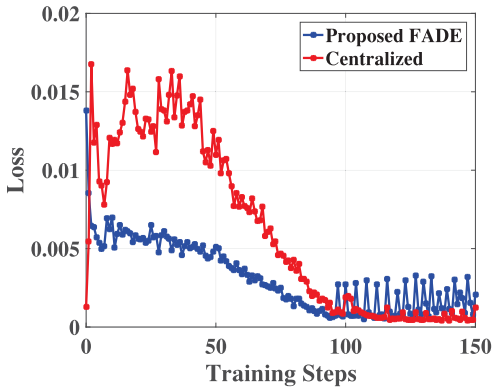


Fig. 7. Demonstration of the loss function between FADE and traditional centralized DRL.

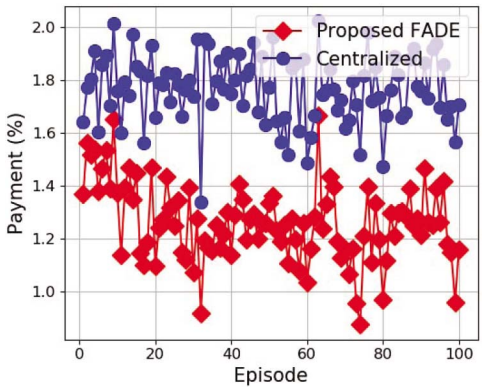


Fig. 8. Demonstration of system payments between FADE and traditional centralized DRL.

5) *Centralized DRL:* The dynamic content replacement is operated in the centralized BS or cloud by deploying the DRL algorithm [46].

First, we demonstrate the loss function between FADE and centralized DRL training, shown as Fig. 7. The results are averaged by 100 trials. It is obvious that FADE reduces the 92% loss in the first 100 training steps and then obtains almost the same values compared to the centralized DRL. This finding indicates that our algorithm achieves better performance in terms of stability and has quick convergence with much lower loss performance. This result may be caused by the small training data in the individual UE.

Moreover, we introduce the offload utility as the system payment to compare the efficiency between FADE and the centralized algorithm. The system payment here refers to the ratio of offloaded content to the downlink data rate at each episode, indicating the network overhead of information exchange. From Fig. 8, FADE outperforms the traditional centralized algorithm with an average 60% improvement for the system payment. This situation occurs mainly because a large amount of content needs to be transferred to the cloud for training in the centralized algorithm, while FADE shares the training parameters. Only when cache replacement occurs is the content transmitted. Thus, the network overhead is significantly reduced.

The performance demonstrations of the average content access delay, hit rate, and backhaul traffic offload are shown in Fig. 9. The parameters of the content number and cache size of UEs are set as $F = 10\,000$ and $C = 100$ MB. Oracle always shows the best performance over the other strategies. Compared to it, the proposed FADE algorithm has only an average 5% performance loss gap, which shows the superiority of the proposed algorithm.

From Fig. 9(a), the proposed FADE algorithm shows a high value of average delay at first. However, after decreasing rapidly after two episodes, the value is maintained in a relatively stable state. The proposed FADE algorithm outperforms the other algorithms; it achieves the lowest average delay of 0.29 s, improving the performance of 29%, 27%, and 26% compared to LRU, FIFO, and LFU, respectively.

In particular, affected by the advantages in the performance of average delay, the proposed FADE algorithm also achieves better performance with respect to the hit rate. Shown as Fig. 9(b), almost 50% content requests are satisfied by the proposed FADE algorithm and it outperforms the LRU, LFU, and FIFO algorithms with up to 8%, 10%, and 15% improvements.

From Fig. 9(c), it is observed that the proposed algorithm can offload more backhaul traffic by 54%–75%, and outperforms the LRU, LFU, and FIFO algorithms by up to 5%, 20%, and 15%, respectively.

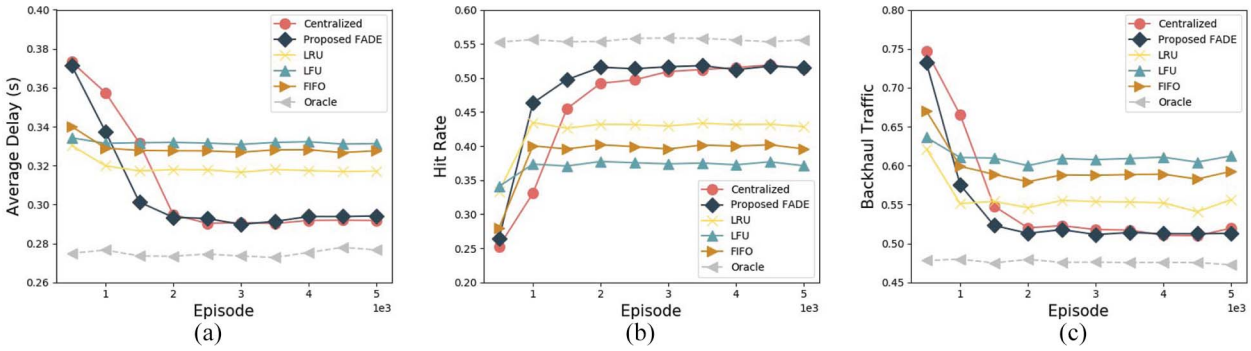


Fig. 9. Performance evaluation in terms of the (a) average delay, (b) hit rate, and (c) traffic with respect to time.

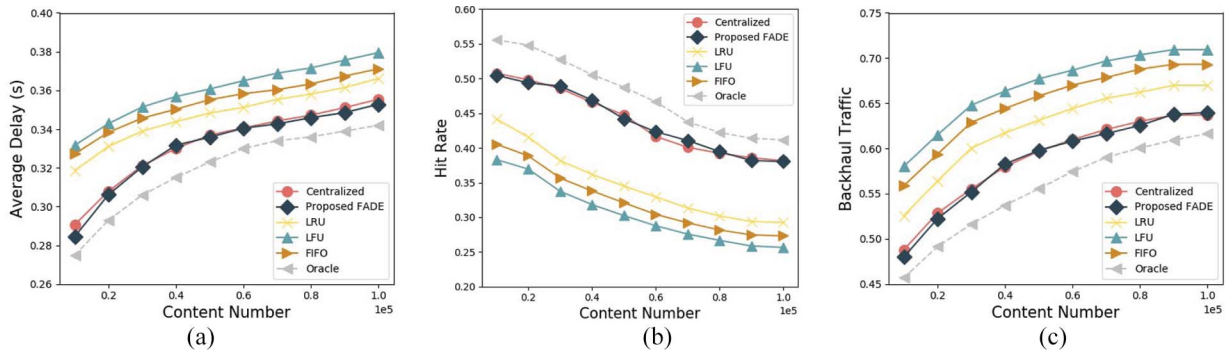


Fig. 10. Performance evaluation in terms of (a) average delay, (b) hit rate, and (c) traffic with respect to content numbers.

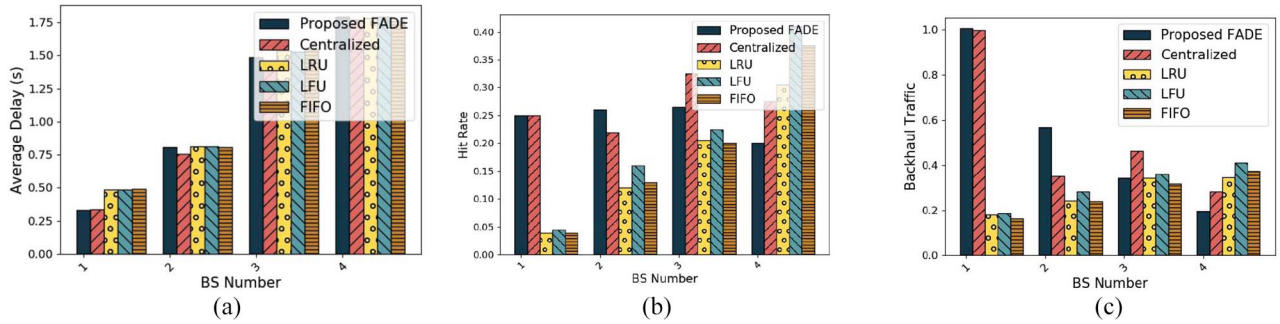


Fig. 11. Performance evaluation in terms of (a) average delay, (b) hit rate, and (c) traffic with respect to BS numbers.

Specifically, the proposed FADE algorithm outperforms the traditional centralized algorithm in the first two episodes and then achieves almost the same performance because the system reward is used to reduce the content access delay of UEs, which makes FADE minimize the average delay.

Fig. 10 demonstrates the network performance under different content numbers. The content number ranges from 10 000 to 100 000 and the cache capacity is set as $C = 100$ MB. According to Fig. 10(a)–(c), the proposed algorithm suffers an average of 3% performance loss compared to the Oracle algorithm and outperforms the other algorithms. For instance, the FADE algorithm improves the hit rate performance with up to 8%, 15%, and 11% compared to the LRU, LFU, and FIFO algorithms, respectively, in Fig. 10(b). However, the trends of the curves in Figs. 9 and 10 are exactly the opposite because more popular contents are cached under the fixed cache capacity of BS along with the increasing content number, leading to the rising trend in Fig. 10(a) and (c). The decreasing trend in Fig. 10(b) is caused by the newly requested contents not being replaced when more popular contents are cached in BSs.

We also evaluate the network performance under different BS numbers (in this case, the cache size of BS is fixed as 100 MB), as shown in Fig. 11. The network performance of FADE fluctuates depending on the BS number. For instance, FADE achieves the best delay performance when the BS number is 1 in Fig. 11(a); however, it shows a decreasing trend with increasing BS number because the number of information exchanges between UE and BS increased, leading to the excessive cost of communications. A similar situation occurs in the aspect of hit rate performance in Fig. 11(b); the FADE outperforms the centralized algorithm by 30% when the BS number is equal to 2. However, the performance of other algorithms is better than that of FADE when the number of BSs is equal to 3 and 4, mainly because other algorithms do not need to learn the user's request behavior, and more BS means more popular

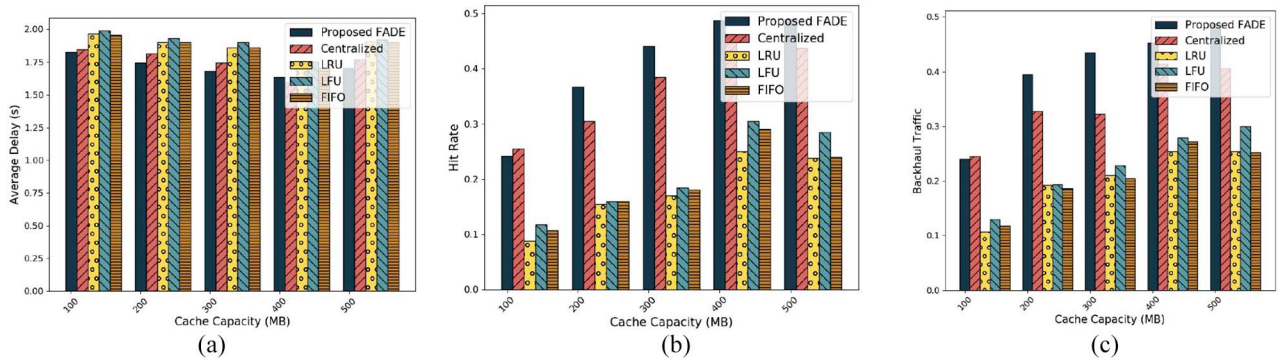


Fig. 12. Performance evaluation in terms of (a) average delay, (b) hit rate, and (c) traffic with respect to cache capacity.

content is cached. In this way, it is easier to obtain better performance for the linear cache replacement algorithms (e.g., LRU, LFU, and FIFO). Fig. 11(c) shows that the proposed algorithm outperforms the centralized, LRU, LFU, and FIFO algorithms with up to 21%, 35%, 30%, and 37% improvements when the BS number is 2. The performance of FADE decreases when the BS number is 3 and 4, mainly because the traffic pressure is apportioned by more BSs.

We compare the network performance under different cache capacities of the BS (in this case, the BS number is 4) in Fig. 12. It can be observed that FADE achieves better performance in terms of hit rate and backhaul traffic offload. However, the delay performance of FADE is poor due to the excessive information exchange. From Fig. 12(b), on average the proposed FADE algorithm improves the hit rate performance by 5%, 27%, 25%, and 26% compared to the centralized LRU, LFU, and FIFO algorithms. FADE also achieves better performance in terms of backhaul traffic offload when the cache size is larger than 100 MB, as shown in Fig. 12(c). The proposed FADE algorithm outperforms the centralized, LRU, LFU, and FIFO algorithms by 9%, 22%, 20%, and 24% because when the cache capacity is large enough to store more contents, replacement processes rarely occur.

The aforementioned simulation results verify that FADE achieves almost the same level of network performance as the traditional centralized approach, which shows its efficiency. In the centralized training process, since it is assumed that the training data can be uploaded to the cloud or edge servers without loss, the transmission delay is ignored [31], [48]. However, it is impossible in practical scenarios, which further proves the efficiency of FADE.

We evaluate FADE on different learning-related parameters in terms of the exploration probability ϵ, reward decay γ, learning rate α, and batch size $\tilde{\mathcal{M}}_i$. In this case, the cache size of each BS is set as 100 MB, and the number of BS is 4.

Fig. 13 shows the performance comparisons for FADE in terms of the hit rate with different exploration probabilities $\epsilon = 0.1$, $\epsilon = 0.5$, and $\epsilon = 0.9$. The exploration probability has strong effects on the convergence and performance of the FADE algorithm. Simply increasing the exploration probability may not improve the performance. Thus, a large number of trials need to be performed to obtain an appropriate exploration probability in the considered edge caching scenarios.

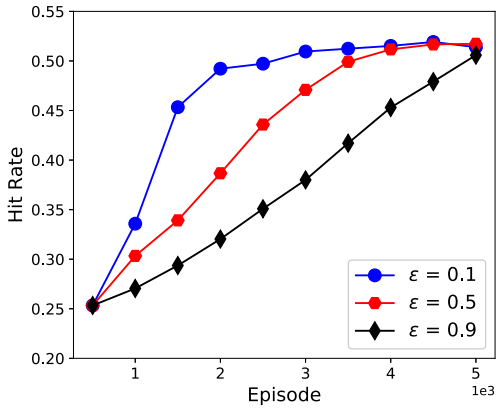


Fig. 13. Performance of the hit rate under different exploration probabilities.

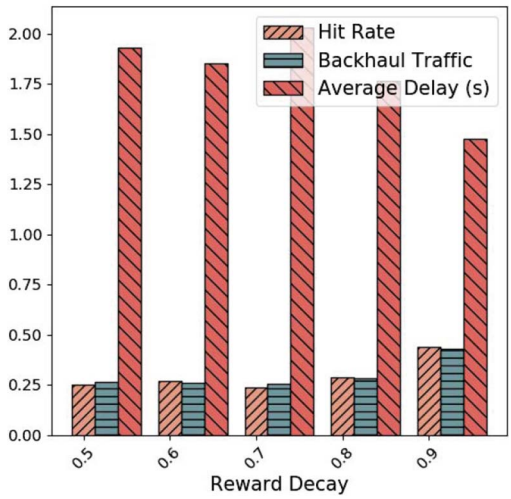


Fig. 14. Performance of the hit rate, backhaul traffic, and average delay under different reward decays.

Hence, in our setting, $\epsilon = 0.1$ is selected to achieve better performance.

Moreover, we demonstrate the network performance of the hit rate, backhaul traffic, and average delay in the following figures. Fig. 14 compares the network performance under different reward decay γ. It is observed that FADE achieves the best network performance of the overall metrics when $\gamma = 0.9$. Note that 0.9 is an empirical value that is widely used in other studies. The results also prove the desirability

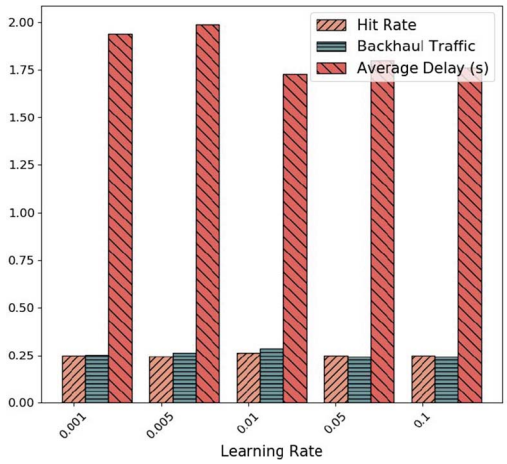


Fig. 15. Performance of the hit rate, backhaul traffic, and average delay under different learning rates.

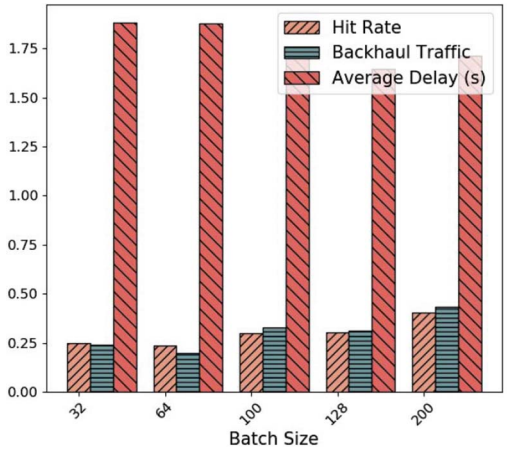


Fig. 16. Performance of the hit rate, backhaul traffic, and average delay under different batch sizes.

of this value, and $\gamma = 0.9$ is selected as the parameter in our simulations.

We compare the network performance with different learning rates α in Fig. 15. The performance of the hit rate and backhaul traffic offload change little with increasing learning rates. Nevertheless, the average delay shows a stable performance change when the learning rate α ranges from 0.01 to 0.1. Thus, we set the learning rate α = 0.05 as an empirical value to maintain the stability and effectiveness of our algorithm.

Fig. 16 shows the network performance under different batch sizes $\tilde{\mathcal{M}}_i$. It can be seen that the batch size has little effect on the network performance due to the stochastic selection mechanism from the transition memory.

VI. Conclusion

In this article, we have proposed FADE, a federated DRL-based cooperative edge caching framework for IoT systems, to cope with the challenge of offloading duplicated traffic and improving the specific QoS of delays and the hit rate. Different from other caching strategies, the proposed FADE framework has federated all local UEs to collaboratively train the parameters and feed them back to the BS to accelerate the overall convergence speed. Finally, trace-driven simulation results have shown that the proposed FADE framework outperforms the baseline schemes of LRU, LFU, FIFO, and Oracle in terms of the average delay, the hit rate, and the traffic offload of backhaul, and achieves the approximate performance of the centralized DRL scheme with a low loss.

Appendix A
Proof of Lemma 1

Straightforward from Assumption 1, according to the definition of convex, $f(w)$ is the finite-sum structure of $f_i(w)$ and triangle inequality.

Appendix B
Proof of Theorem 1

First, we prove the μ-strongly convexity of $f_(w)$.

Given $\forall w, w' \in \mathbb{R}$, $\beta \in [0, 1]$, assume that $x \coloneqq w + w'$, $y \coloneqq \beta w + (1 + \beta)w'$ and $\exists \beta_1, \beta_2 \in (0, 1)$. Derived from the Taylor formula, we can obtain

$$f(w) = f(y) + \nabla f(y)(w - y) + \frac{1}{2}(w - y)\nabla^2 f(e_1)(w - y) \tag{27}$$

and

$$f(w') = f(y) + \nabla f(y)(w' - y) + \frac{1}{2}(w' - y)\nabla^2 f(e_2)(w' - y) \tag{28}$$

where $e_1 \coloneqq y + \beta_1(w - y)$ and $e_2 \coloneqq y + \beta_2(w - y)$; by incorporating the two formulas above, we have

$$\beta f(w) + (1 - \beta) f(w') = f(y) + \frac{1}{2}\beta(1 - \beta)(w - w')^2 \times \Big[(1 - \beta)\nabla^2 f(e_1) + \beta\nabla^2 f(e_2)\Big]. \tag{29}$$

Recalling the definition of strong convexity, there exists the constant μ^* obtaining

$$\frac{1}{2}(w - w')\Big[(1 - \beta)\nabla^2 f(e_1) + \beta\nabla^2 f(e_2)\Big] \geq \mu^* \|w - w'\|^2. \tag{30}$$

Thus, we rewrite (29) as

$$\begin{aligned}\beta f(w) + (1 - \beta) f(w') &= f(y) + \frac{1}{2}\beta(1 - \beta)(w - w')^2 \\ &\quad \times \Big[(1 - \beta)\nabla^2 f(e_1) + \beta\nabla^2 f(e_2)\Big] \\ &\geq f(y) + \mu^*\beta(1 - \beta)\|w - w'\|^2 \end{aligned} \tag{31}$$

where $\mu^* = \mu/2$ and bring $y \coloneqq \beta w + (1 + \beta)w'$ in (31), the μ-strongly convexity of $f(w)$ is proven.

According to the μ-strongly convexity of $f(w)$, we have

$$\nabla f(w)(w - w_*) \geq f(w) - f(w_*) + \frac{\mu}{2}\|w - w_*\|^2. \tag{32}$$

Thus, we can obtain the following:

$$\begin{aligned}\|w_{t+1} - w_*\|^2 &= \|w_t - \eta\nabla f(w_t) - w_*\|^2 \\ &= \|w_t - w_*\|^2 - 2\eta\nabla f(w_t)(w_t - w_*)\end{aligned}$$

$$
\begin{aligned}
&+ \eta^2 \|\nabla f(w_t)\|^2 \\
&\leq \|w_t - w_*\|^2 \\
&\quad - 2\eta\Big(f(w) - f(w_*) + \frac{\mu}{2}\|w_t - w_*\|^2\Big) \\
&\quad + \eta^2 \|\nabla f(w_t)\|^2. \qquad (33)
\end{aligned}
$$

By smoothing $f(w)$, we can obtain the gradient bound

$$
\begin{aligned}
f(w_*) &\leq f\left(w - \frac{1}{L}\nabla f(w)\right) \\
&\leq f(w) - \|\nabla f(w)\|^2 + \frac{1}{2L}\|\nabla f(w)\|^2 \\
&\leq f(w) - \frac{1}{2L}\|\nabla f(w)\|^2. \qquad (34)
\end{aligned}
$$

By incorporating (34), (33) can be transformed as

$$
\begin{aligned}
\|w_{t+1} - w_*\|^2 &= \|w_t - \eta\nabla f(w_t) - w_*\|^2 \\
&\leq \|w_t - w_*\|^2 - \eta\mu\|w_t - w_*\|^2 \\
&\quad + 2\eta(\eta L - 1)(f(w) - f(w_*)) \\
&\leq \left(1 - \frac{\mu}{L}\right)\|w_t - w_*\|^2 \leq \left(1 - \frac{\mu}{L}\right)\|\Delta^*(w)\|^2 \qquad (35)
\end{aligned}
$$

where η is set as the last step.

REFERENCES

[1] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Process. Mag.*, vol. 31, no. 6, pp. 45–55, Nov. 2014.

[2] D. Li *et al.*, "Deep reinforcement learning for cooperative edge caching in future mobile networks," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Marrakesh, Morocco, Apr. 2019, pp. 1–6.

[3] L. Pu, X. Chen, G. Mao, Q. Xie, and J. Xu, "*CHIMERA*: An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 84–99, Feb. 2019.

[4] L. Zeng, E. Li, Z. Zhou, and X. Chen, "Boomerang: On-demand cooperative deep neural network inference for edge intelligence on the industrial Internet of Things," *IEEE Netw.*, vol. 33, no. 5, pp. 96–103, Sep./Oct. 2019.

[5] Q. Wu, X. Chen, Z. Zhou, and L. Chen, "Mobile social data learning for user-centric location prediction with application in mobile edge service migration," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7737–7747, Oct. 2019.

[6] K. Samdanis, T. Taleb, and S. Schmid, "Traffic offload enhancements for eUTRAN," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 3, pp. 884–896, 3rd Quart., 2012.

[7] H. Zhou, H. Wang, X. Li, and V. C. M. Leung, "A survey on mobile data offloading technologies," *IEEE Access*, vol. 6, pp. 5101–5111, 2018.

[8] M. Dai, H. Deng, B. Chen, G. Su, X. Lin, and H. Wang, "Design of binary erasure code with triple simultaneous objectives for distributed edge caching in industrial IoT networks," *IEEE Trans. Ind. Informat.*, early access, Nov. 13, 2019, doi: 10.1109/TII.2019.2952643.

[9] S. Ajmal, M. B. Muzammil, A. Jamil, S. M. Abbas, U. Iqbal, and P. Touseef, "Survey on cache schemes in heterogeneous networks using 5G Internet of Things," in *Proc. ACM Int. Conf. Future Netw. Distrib. Syst. (ICFNDS)*, Paris, France, Jul. 2019, pp. 1–8.

[10] H. Wei, H. Luo, Y. Sun, and M. S. Obaidat, "Cache-aware computation offloading in IoT systems," *IEEE Syst. J.*, vol. 14, no. 1, pp. 61–72, May 2019.

[11] L. Lei, X. Xiong, L. Hou, and K. Zheng, "Collaborative edge caching through service function chaining: Architecture and challenges," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 94–102, Jun. 2018.

[12] T. A. Do, S. W. Jeon, and W. Y. Shin, "How to cache in mobile hybrid IoT networks?" *IEEE Access*, vol. 7, pp. 27814–27828, 2019.

[13] Y. Han, R. Wang, and J. Wu, "Random caching optimization in large-scale cache-enabled Internet of Things networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 385–397, Jan.–Mar. 2019.

[14] Apple Inc. (2020). *Apple Edge Cache*. [Online]. Available: https://cache.edge.apple/

[15] X. Li, X. Wang, K. Li, Z. Han, and V. C. M. Leung, "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6926–6936, Oct. 2017.

[16] T. Wang, G. Zhang, A. Liu, M. Z. Bhuiyan, and Q. A. Jin, "A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4831–4843, Jun. 2019.

[17] X. Wang, M. Chen, Z. Han, D. O. Wu, and T. T. Kwon, "TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2014, pp. 2346–2354.

[18] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[19] M. Sheng, C. Xu, J. Liu, J. Song, and J. Li, "Enhancement for content delivery with proximity communications in caching enabled wireless networks: Architecture and challenges," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 70–76, Aug. 2016.

[20] E. Zeydan *et al.*, "Big data caching for networking: Moving from cloud to edge," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 36–42, Sep. 2016.

[21] N. Golrezaei, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.

[22] S. Woo, E. Jeong, S. Park, J. Lee, S. Ihm, and K. Park, "Comparison of caching strategies in modern cellular backhaul networks," in *Proc. ACM MobiSys*, Jun. 2013, pp. 319–332.

[23] X. Li, X. Wang, S. Xiao, and V. C. M. Leung, "Delay performance analysis of cooperative cell caching in future mobile networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2015, pp. 5652–5657.

[24] S. H. Chae, J. Y. Ryu, T. Q. S. Quek, and W. Choi, "Cooperative transmission via caching helpers," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[25] T. X. Vu, S. Chatzinotas, and B. E. Ottersten, "Edge-caching wireless networks: Energy-efficient design and optimization," 2017. [Online]. Available: arXiv.abs/1705.05590.

[26] X. Zhao, P. Yuan, H. Li, and S. Tang, "Collaborative edge caching in context-aware device-to-device networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 10, pp. 9583–9596, Oct. 2018.

[27] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov./Dec. 2018.

[28] Y. M. Saputra, D. T. Hoang, D. N. Nguyen, E. Dutkiewicz, D. Niyato, and D. I. Kim, "Distributed deep learning at the edge: A novel proactive and cooperative caching framework for mobile edge networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 1220–1223, Apr. 2019.

[29] X. Fan, Y. Huang, X. Ma, J. Liu, and V. C. M. Leung, "Exploiting the edge power: An edge deep learning framework," *CCF Trans. Netw.*, vol. 2, no. 1, pp. 4–11, Jun. 2018.

[30] H. Pang, J. Liu, X. Fan, and L. Sun, "Toward smart and cooperative edge caching for 5G networks: A deep learning based approach," in *Proc. Int. Workshop Qual. Service (IWQoS)*, Banff, AB, Canada, Jun. 2018, pp. 1–6.

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

[32] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.

[33] M. Zhang, J. Chen, S. He, L. Yang, X. Gong, and J. Zhang, "Privacy-preserving database assisted spectrum access for industrial Internet of Things: A distributed learning approach," *IEEE Trans. Ind. Electron.*, vol. 67, no. 8, pp. 7094–7103, Aug. 2020.

[34] X. Li, X. Wang, P.-J. Wan, Z. Han, and V. C. M. Leung, "Hierarchical edge caching in device-to-device aided mobile networks: Modeling, optimization, and design," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1768–1785, Aug. 2018.

[35] S. Zhang, P. He, K. Suto, P. Yang, L. Zhao, and X. S. Shen, "Cooperative edge caching in user-centric clustered mobile networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 8, pp. 1791–1805, Aug. 2018.

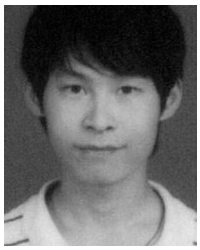
[36] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 3185–3190.

[37] S. Zhang and J. Liu, "Optimal probabilistic caching in heterogeneous IoT networks," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3404–3414, Apr. 2020.
[38] J. Gu, W. Wang, A. Huang, H. Shan, and Z. Zhang, "Distributed cache replacement for caching-enable base stations in cellular networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 2648–2653.
[39] P. Blasco and D. Gündüz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Sydney, NSW, Australia, Jun. 2014, pp. 1897–1903.
[40] B. Chen, L. Liu, M. Sun, and H. Ma, "IoTCache: Toward data-driven network caching for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10064–10076, Dec. 2019.
[41] S. Lim, W.-C. Lee, G. Cao, and C. R. Das, "A novel caching scheme for Internet based mobile ad hoc networks," in *Proc. ICCCN*, Dallas, TX, USA, Oct. 2003, pp. 38–43.
[42] M. X. Goemans, L. Li, V. S. Mirrokni, and M. Thottan, "Market sharing games applied to content distribution in ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 5, pp. 1020–1033, May 2006.
[43] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1447–1460, Dec. 2008.
[44] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
[45] M. Volodymyr *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[46] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double *Q*-learning," in *Proc. AAAI*, Feb. 2016, pp. 2094–2100.
[47] "Further advancements for E-UTRA physical layer aspects (release 9), v1.2.0" 3GPP, Sophia Antipolis, France, Rep. TR 36.814, Jun. 2009.
[48] S. Shen, Y. Han, X. Wang, and Y. Wang, "Computation offloading with multiple agents in edge-computing-supported IoT," *ACM Trans. Sens. Netw.*, vol. 16, no. 1, pp. 1–27, Dec. 2019.
[49] J. Konecnỳ, H. B. McMahan, F. X. Yu, P. Richtàrik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proc. Neural Inf. Process. Syst. (NIPS)*, Barcelona, Spain, Dec. 2016, pp. 1223–1231.
[50] H. B. Mcmahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2016. [Online]. Available: arXiv.1602.05629.
[51] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.
[52] S. Müller, O. Atan, M. van der Schaar, and A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024–1036, Feb. 2017.

Xiaofei Wang (Senior Member, IEEE) received the master's and Doctoral degrees from Seoul National University, Seoul, South Korea, in 2006 and 2013, respectively.

He is currently a Professor with Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, Tianjin, China. He was a Postdoctoral Fellow with the University of British Columbia, Vancouver, BC, Canada, from 2014 to 2016. He has authored over 100 technical papers in the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE WIRELESS COMMUNICATIONS, IEEE COMMUNICATIONS, the the IEEE TRANSACTIONS ON MULTIMEDIA, IEEE INFOCOM, and IEEE SECON. Focusing on the research of social-aware cloud computing, cooperative cell caching, and mobile traffic offloading.

Prof. Wang was a recipient of the National Thousand Talents Plan (Youth) of China. He received the "Scholarship for Excellent Foreign Students in IT Field" by NIPA of South Korea from 2008 to 2011, the "Global Outstanding Chinese Ph.D. Student Award" by the Ministry of Education of China in 2012, and the Peiyang Scholar from Tianjin University. In 2017, he received the "Fred W. Ellersick Prize" from the IEEE Communication Society.



Chenyang Wang (Student Member, IEEE) received the B.S. and M.S. degrees in computer science and technology from Henan Normal University, Xinxiang, China, in 2013 and 2017, respectively. He is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, College of Intelligence and Computing, Tianjin University, Tianjin, China.

His current research interests include edge computing, big data analytics, reinforcement learning, and deep learning.

Mr. Wang received the "Best Student Paper Award" of the 24th International Conference on Parallel and Distributed Systems by IEEE Computer Society in 2018.

Xiuhua Li (Member, IEEE) received the B.S. degree from the Honors School, Harbin Institute of Technology, Harbin, China, in 2011, the M.S. degree from the School of Electronics and Information Engineering, Harbin Institute of Technology, in 2013, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, in 2018.

He joined Chongqing University through One-Hundred Talents Plan of Chongqing University in 2019. He is currently a tenure-track Assistant Professor with the School of Big Data and Software Engineering, and the Director of the Institute of Intelligent Software and Services Computing associated with the Key Laboratory of Dependable Service Computing in Cyber Physical Society, Education Ministry, Chongqing University, Chongqing, China, where he is also leading a research team with the State Key Laboratory of Power Transmission Equipment and System Security and New Technology. His current research interests are 5G/B5G mobile Internet, mobile edge computing and caching, big data analytics, and machine learning.

Victor C. M. Leung (Fellow, IEEE) received the B.A.Sc. (Hons.) and Ph.D. degrees in electrical engineering from the University of British Columbia (UBC), Vancouver, BC, Canada, in 1977 and 1982, respectively.

He is a Distinguished Professor of computer science and software engineering with Shenzhen University, Shenzhen, China. He is also an Emeritus Professor of electrical and computer engineering and the Director of the Laboratory for Wireless Networks and Mobile Systems, UBC. He has coauthored more than 1300 journal/conference papers and book chapters. His research is in the broad areas of wireless networks and mobile systems.

Dr. Leung received the APEBC Gold Medal in 1977, the NSERC Postgraduate Scholarships from 1977 to 1981, an IEEE Vancouver Section Centennial Award, the UBC Killam Research Prize in 2011, the Canadian Award for Telecommunications Research in 2017, and the IEEE TCGCC Distinguished Technical Achievement Recognition Award in 2018. He coauthored papers that won the IEEE ComSoc Fred W. Ellersick Prize in 2017, the IEEE SYSTEMS JOURNAL Best Paper Award in 2017, the IEEE CSIM Best Journal Paper Award in 2018, and the IEEE TCGCC Best Journal Paper Award in 2019. He is named in the current Clarivate Analytics list of "Highly Cited Researchers." He is serving on the editorial boards of the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, the IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE ACCESS, IEEE NETWORK, and several other journals. He has previously served on the editorial boards of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS-Wireless Communications Series and Series on Green Communications and Networking, the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON COMPUTERS, and IEEE WIRELESS COMMUNICATIONS LETTERS. He has guest-edited many journal special issues, and provided leadership to the organizing committees and technical program committees of numerous conferences and workshops. He is a Fellow of the Royal Society.

Tarik Taleb (Senior Member, IEEE) received the B.E. degree (Distinction) in information engineering and the M.Sc. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 2001, 2003, and 2005, respectively.

He is currently a Professor with the School of Electrical Engineering, Aalto University, Espoo, Finland. He is the Founder and the Director of the MOSA!C Lab, Espoo. He is a part-time Professor with the Center of Wireless Communications, University of Oulu, Oulu, Finland. He was an Assistant Professor with the Graduate School of Information Sciences, Tohoku University, in a laboratory fully funded by KDDI until 2009. He was a Senior Researcher and a 3GPP Standards Expert with NEC Europe Ltd., Heidelberg, Germany. He was then leading the NEC Europe Labs Team, involved with research and development projects on carrier cloud platforms, an important vision of 5G systems. From 2005 to 2006, he was a Research Fellow with the Intelligent Cosmos Research Institute, Sendai. He has also been directly engaged in the development and standardization of the Evolved Packet System as a Member of the 3GPP System Architecture Working Group. His current research interests include architectural enhancements to mobile core networks (particularly, 3GPP's), network softwarization and slicing, mobile cloud networking, network function virtualization, software defined networking, mobile multimedia streaming, intervehicular communications, and social media networking.

Prof. Taleb was a recipient of the 2017 IEEE ComSoc Communications Software Technical Achievement Award in 2017 for his outstanding contributions to network softwarization and the Best Paper Awards at prestigious IEEE-flagged conferences for some of his research work. He was a corecipient of the 2017 IEEE Communications Society Fred W. Ellersick Prize in 2017, the 2009 IEEE ComSoc Asia–Pacific Best Young Researcher Award in 2009, the 2008 TELECOM System Technology Award from the Telecommunications Advancement Foundation in 2008, the 2007 Funai Foundation Science Promotion Award in 2007, the 2006 IEEE Computer Society Japan Chapter Young Author Award in 2006, the Niwa Yasujirou Memorial Award in 2005, and the Young Researcher's Encouragement Award from the Japan Chapter of the IEEE Vehicular Technology Society in 2003. He is a member of the IEEE Communications Society Standardization Program Development Board. He is/was on the editorial board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, *IEEE Wireless Communications Magazine*, the IEEE INTERNET OF THINGS JOURNAL, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and a number of Wiley journals.