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 convergence of the proposed FADE, and it achieves the expectation convergence. Trace-driven simulation results show that the proposed FADE framework reduces 92% of performance loss and average 60% system payment over the centralized deep reinforcement learning (DRL) algorithm, achieves only a 4% performance loss of the desirable omniscient oracle algorithm, and obtains 7%, 11% and 9% network performance improvements compared to some existing schemes, i.e., least recently used (LRU), least frequently used (LFU) and first-in first-out (FIFO), respectively.**

***Index Terms*—Internet of Things, Edge Caching, Cooperative Caching, Hit Rate, Deep Reinforcement Learning, Federated Learning.**

Part of this work was presented at the IEEE Wireless Communications and Networking Conference (WCNC), April 15-18, 2019, Marrakesh, Morocco. This work is supported in part by the National Key R & D Program of China through grant No. 2019YFB2101901, 2018YFC0809803, 2018YFF0214700 and 2018YFF0214706, China NSFC through grants 61702364, 61902044 and 61672117, China NSFC GD Joint Fund U1701263, and Chongqing Research Program of Basic Research and Frontier Technology through Grant No. cstc2019jcyj-msxmX0589, Chinese National Engineering Laboratory for Big Data System Computing Technology, Canadian NSERC, the European Union's Horizon 2020 Research and Innovation Program through the MonB5G Project under Grant No. 871780, the Academy of Finland 6Genesis project under Grant No. 318927, and the Academy of Finland CSN project under Grant No. 311654. *(Corresponding author: Xiuhua Li.)*

X. Wang and C. Wang are with College of Intelligence and Computing, Tianjin University, Tianjin, 300072 China (e-mail: {xiaofeiwang, chenyangwang}@tju.edu.cn).

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

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

T. Taleb is with the Department of Communications and Networking, School of Electrical Engineering, Aalto University, 02150 Espoo, Finland and Information Technology and Electrical Engineering, Oulu University, Pentti Kaiteran katu 1, 90570 Oulu, Finland and the Department of Computer and Information Security, Sejong University, 209 Neungdong-ro, Gunja-dong, Gwangjin-gu, Seoul, 05006 South Korea (e-mail: Tarik.Taleb@aalto.fi).

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., smartphones, tablets and smartwatches) are embedded in people's daily lives. For instance, as reported in [1], the number of IoT devices will surpass 10 billion by 2020, and higher real-time quality service requirements (e.g., heart rate monitor, step count and outdoor video live) from these devices are required. Facing the rocket-rising network traffic load and Quality of Service/Experience (QoS/QoE) of user demands, enormous challenges have emerged for mobile networks and the IoT [2]–[5].

In particular, due to the integration of powerful sensoring and computing functions, IoT devices are equipped with intelligent identification, behavior tracking and daily management, heart rate monitoring, etc. [6]–[8]. 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, 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 [9]–[11].

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., base stations (BSs) and IoT devices) from the mobile network operator (MNO) or the cloud. This paper 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 resiliency of QoS/QoE and provide the best performance for IoT devices with content service requirements (e.g., system update package) from the internet service provider (SP), related applications exist. For instance, Apple Inc. recently launched the Apple Edge Cache service [12], enabling the delivery of Apple content services directly to the equipment

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

within SP partner networks. Caching the requested contents in edge nodes (e.g., BSs) are similar to that in IoT devices, which improves the efficiency of content access compared to excessive downloading via backhaul links [13]–[15].

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 [16]–[19] investigated the architectures of collaborative edge caching in mobile networks. The authors in [20]–[23] optimized the issue of 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 [24] and context awareness [25].

Recently, learning-based approaches have also been widely utilized to design and optimize edge caching [26]–[29]. For instance, deep reinforcement learning (DRL) was considered for comprehensive resource allocation, as in [30]. This approach maximized the long-term reward of energy consumption and required 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 datacenter by sharing the original data, leading to a large amount of network resource consumption during the data transmission process [31], [32].

To cope with the dynamic environment and ensure data localization training for IoT devices, we expect to optimize the edge caching problem in a long-term and decentralized fashion. Motivated by the aforementioned, we propose a **F**eder**A**ted **D**eep reinforcement learning-based cooperative **E**dge 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. The proposed FADE is performed in a decentralized model. 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 will be averaged to improve the shared model.

The main contributions of this paper are summarized as follows:

- 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 (DQN) to address the problem of data sampling in the uncontinuous huge spaces.
- We propose a FADE framework, which can enable fast training and decouple the learning process from the data stored in the cloud in a distributed-centralized way, which keeps the data training in the local UEs. In addition, we prove that the FADE is L-smooth and μ-strong and derive its expectation of convergence.
- Trace-driven simulation results show that the proposed FADE framework reduces 92% loss performance and average 60% system payment over the centralized DRL algorithm and outperforms the existing LRU, LFU and FIFO by 7%, 11% and 9% improvements, respectively.

The remainder of this paper is organized as follows. Sec. II summarizes the previous work. We establish the system model and formulate the optimization problem in Sec. III. The framework design of the proposed FADE is presented in Sec. IV. Trace-driven simulation results evaluate the effectiveness of the proposed framework in Sec. V. Finally, Sec. VI concludes this paper.

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, [33] focused on maximizing traffic offloading and reducing the system costs by designing a hierarchical edge caching strategy. [34] 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. [35] proposed a content replacement strategy with multi-attribute 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, [36] 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 adapt to dynamic environments and hard to deploy due to the global information required in practice.

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

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. [37], [38] showed that RL has great potential for the utilization in the scheme design of content caching in BSs. Specifically, [37] proposed a cache replacement strategy based on Q-Learning to reduce traffic load in future cellular networks, and reinforcement learning (RL) was also employed for cache placement [38] by using a multi-armed bandit (MAB). Chen et al. [39] proposed a popularity-based caching strategy for IoT networks by deploying deep neural networks to predict the near future popularity of IoT data. However, most centralized learning algorithms are prone to posing high cache diversity and storage utilization, which leads 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 demonstrated. Finally, we formulate the optimization problem of edge caching for IoT systems. Some key parameters are listed in Table I.

TABLE I
KEY PARAMETERS AND NOTATIONS.

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 SPs, 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
$\mathbf{s}_{i,n}^c$	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
$\boldsymbol{\Phi}(\boldsymbol{\chi}_i) = \{\mathbf{a}_i^{\text{local}}, \mathbf{a}_i^{\text{co-BS}}, a_i^{\text{SP}}\}$	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 cooperative edge caching-supported IoT networks is illustrated in Fig. 2. Particularly, in the considered IoT network, a high number of geographically distributed UEs (e.g., smartphones, tablets, and 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 by edge servers or directly by downloading the contents from SPs (in the cloud) to the BSs via the MNO core.

Considering a hierarchical IoT network, $\mathcal{N} = \{1, 2, ..., N_b\}$ fully connected BSs with a finite cache size of C and $\mathcal{U} = \{1, 2, ..., N_u\}$ UEs are distributed in the service area. Denote $\mathcal{F} = \{1, 2, ..., F\}$ as a library of contents that are supported by Internet SPs and that 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 from 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] [42] as

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

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 SPs. The wireless transmission delay d^{c} can be regarded as the period of a UE obtaining the content from the local BS. Considering $\mathcal{M} = \{1, 2, ..., M\}$, wireless channels are deployed, and $a_u \in \mathcal{M}$ is the channel that is assigned to UE u by BS. Similar to [43], 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 $\mathcal{D} = \{D_1,\ D_2, ..., D_f, ...D_{\mathcal{F}}\}$ to be the size of local datasets as well, which will be introduced in Section IV-B.

[3]Note that it is also widely used in mobile IoT scenarios [40] [41].

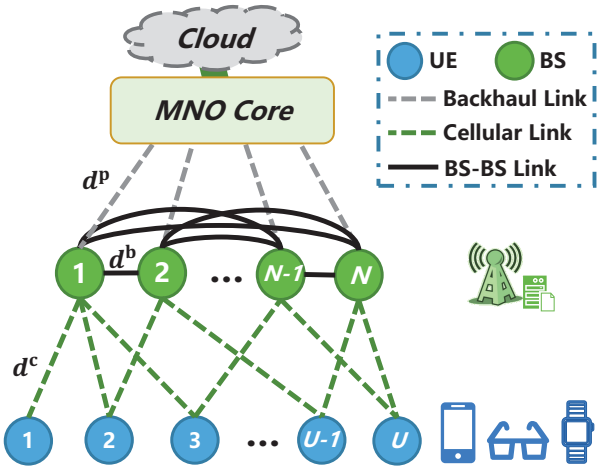


Fig. 2. Topology of the caching-supported IoT system.

Considering $P_{u,f} = r_{u,f}/R_u, f \in \mathcal{F}$ 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 a Markov decision process (MDP) [44]. The state of the cache and request, system action and feedback reward are demonstrated.

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

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

*2) **System Action**:* To adapt the continuous changes in the dynamic environment, BSs can choose which contents should be replaced and decide where the requests are processed (via local BS, BS cooperation or SPs). 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

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

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

*a) **Local Processing Action**:* We denote $\mathbf{a}_i^{\mathrm{local}} \overset{\mathrm{def}}{=} [a_{i,0}^{\mathrm{local}}, a_{i,1}^{\mathrm{local}}, \ldots, a_{i,F}^{\mathrm{local}}]$ as the local processing action when the cache state controlled by the local BS is available, where $a_{i,f}^{\mathrm{local}} \in \{0,1\}, f \in \mathcal{F}$, and $a_{i,f}^{\mathrm{local}} = 1$ indicates that content f needs to be replaced by the current requested content, while $a_{i,f}^{\mathrm{local}} = 0$ 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 $\mathbf{a}_i^{\mathrm{co-BS}} \overset{\mathrm{def}}{=} [a_{i,1}^{\mathrm{co-BS}}, \ldots, a_{i,N}^{\mathrm{co-BS}}]$ as the cooperation processing action, where $a_{i,n}^{\mathrm{co-BS}} \in \{0,1\}$, and $a_{i,n}^{\mathrm{co-BS}} = 1$ 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 neighbors. The local BS decides whether to forward the request to SPs, denoted as $a_i^{\mathrm{SP}} \in \{0,1\}$, where $a_i^{\mathrm{SP}} = 1$ represents the request that will be handled by SPs. In this case, the UE should obtain the requested content f directly from the remote SPs.

*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 the (3) of the communication model in Section II.B, 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

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

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; $p_{nf}e^{-\left(\xi_1 d_n^c + \xi_2 d^b\right)}$ 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 will be $p_{nf}e^{-\left(\xi_1 d_n^c + \xi_3 d^P\right)}$.

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

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

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

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.

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

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

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

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

$$\begin{aligned} & \max \quad V(\chi, \mathbf{\Phi}) \\ s.t. \quad & \sum_{f \in \mathcal{F}} s^c_{i,n,f} \leq \mathcal{C}, \ n \in \mathcal{N}, \\ & s^c_{i,n,f} \in \{0,1\}, \ n \in \mathcal{N} \text{ and } f \in \mathcal{F}, \end{aligned} \tag{10}$$

where i=$\{1, 2, 3, ...\}$ 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 $\mathcal{C}$.

IV. Framework Design of FADE

In this section, we first formulate the deep reinforcement learning (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 on BS

In our federated learning-based cooperative edge caching architecture, the local BS first carries out the corresponding action senses $\mathbf{\Phi}(\boldsymbol{\chi}_i)$ based on the current state $\boldsymbol{\chi}_i$. Then, the current feedback reward $\mathcal{R}(\boldsymbol{\chi}_i, \mathbf{\Phi}(\boldsymbol{\chi}_i))$ will be obtained. Finally, the former system state $\boldsymbol{\chi}_i$ is transmitted into the next new one $\boldsymbol{\chi}_{i+1}$. The pretrained parameters will be 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 [44]:

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

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

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

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

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

Incorporating (13), we rewrite (12) as

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

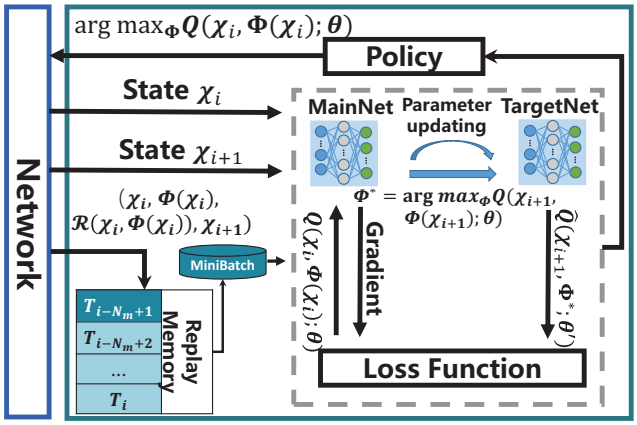


Fig. 3. Caching replacement process of double DQN.

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

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

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

We use double DQN [45] to deploy the pretraining process in local BSs. The caching replacement process of double DQN is shown in Fig. 3. By updating the parameter $\boldsymbol{w}_i$ of the multiple layer perceptron (MLP), we can obtain the approximate optimal Q-value according to the following equation:

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

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 $\mathcal{M} = \{T_{i-N_m+1}, ..., T_i\}$, where $T_i = (\boldsymbol{\chi}_i, \mathbf{\Phi}(\boldsymbol{\chi}_i), \mathcal{R}(\boldsymbol{\chi}_i, \mathbf{\Phi}(\boldsymbol{\chi}_i)), \boldsymbol{\chi}_{i+1})$. $\mathcal{M}$ is updated by the most recent experienced transitions, and agent preserves the $Q(\boldsymbol{\chi}, \mathbf{\Phi}; \boldsymbol{w}_i)$ and $\hat{Q}(\boldsymbol{\chi}, \mathbf{\Phi}; \widehat{\boldsymbol{w}}_{\boldsymbol{i}})$. The Q network (MainNet) is used to select a system action, and the $\hat{Q}$ network (TargetNet) is utilized for evaluating it. It is worth noting that the weight parameters $\widehat{\boldsymbol{w}}_{\boldsymbol{i}}$ in the $\hat{Q}$ network periodically update along with the $\boldsymbol{w}_i$ in the Q network.

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

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

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

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

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 $< \boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i) >$ and the corresponding $Q(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i); \boldsymbol{w}_i)$.

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 $\arg\max_{\boldsymbol{\Phi}(\boldsymbol{\chi}_i)} Q(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i); \boldsymbol{w}_i)$.
12: Execute action $\boldsymbol{\Phi}(\boldsymbol{\chi}_i)$.
13: Obtain immediate reward $\mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i))$.
14: Get the new state $\boldsymbol{\chi}_{i+1}$.
15: Construct $T_i = (\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i), \mathcal{R}(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i)), \boldsymbol{\chi}_{i+1})$.
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 the parameter $\boldsymbol{\theta}_i$ by minimizing the gradient as in (17).
19: Update the parameter $\boldsymbol{w}_i$ of Q network with $\nabla_{w_i} L(w_i)$.
20: Update the parameter $\boldsymbol{\theta}_i$ of Q network with the gradient.
21: Update the parameter $\widehat{\boldsymbol{w}}_{\boldsymbol{i}}$ 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:

- **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 $Q(\boldsymbol{\chi}_i, \boldsymbol{\Phi}(\boldsymbol{\chi}_i); \boldsymbol{w}_i)$.
- **Procedure 2 (Online Caching Process):** On the premise of 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 back propagation. 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 back propagation and gradient descent to train parameters. 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 research [47] [48], complexity analysis proves that our proposed algorithm is sufficiently lightweight for IoT devices and easy to deploy.

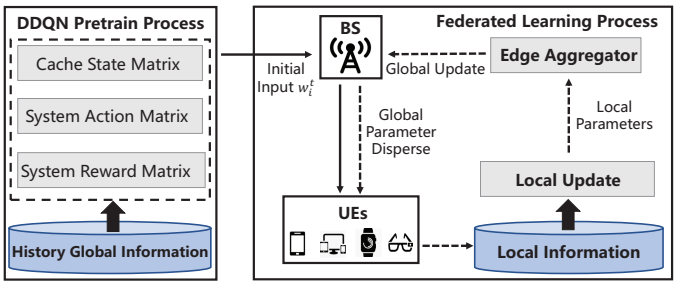


Fig. 4. Overall workflow of the proposed FADE.

B. FADE: Federated DRL-based Edge Caching Algorithm

As mentioned above, the 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:

- It will take a long time even to train each agent well;
- It may endanger sensitive data, especially in industrial informative scenarios;
- 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 carry out the training process distribution, there are still two shortcomings:

- Training each DRL agent from scratch would take a long time or even impossible to converge;
- 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 deep reinforcement learning 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., smartphones, glasses and laptops, etc.), decoupling machine learning from the data stored in the cloud.

The workflow of the proposed FADE is shown in Fig. 4, 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

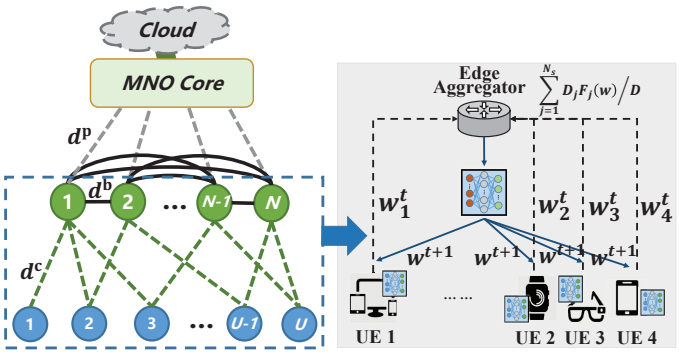


Fig. 5. Mechanism of federated learning.

all the updated local parameters, the improved global model will be continuously dispersed to local agents. The detailed mechanism of the federated learning process is demonstrated in Fig. 5, after the dispersion of the initial parameter w^t, Each UE computes the local update w_j^{t+1} according to (23). The edge aggregator (i.e., 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 datasets $\mathcal{D}$ ={D_1, $D_2, ..., D_j, ...D_s$}. In the learning problem, the task is to find the objective parameter w with the loss function $f(w)$. Some well-known loss functions are $f_i(w) = \frac{1}{2}\min_k \|x_i - w_k\|^2$, where $w := [w_1, w_2, \ldots, w_k]$ for K-means and $f_i(w) = \frac{1}{2}\left\|y_i - w^{\mathrm{T}} x_i\right\|^2$, $y_i \in \mathbb{R}$ for linear regression as well as $f_i(w) = \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\max\{0, 1 - y_i w^{\mathrm{T}} x_i\}^2$ (λ is const.) for the support vector machine. For each local dataset 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 is its local parameter and t is the iteration index, where $t = 1, 2, 3, ..., T$. The process of the gradient descent algorithm is shown in Algorithm 2.

Algorithm 2 Gradient descent algorithm for FADE.

Initialization:
  $w_j^0 = w_i$;
  w_i^t is the pretrained parameter from Algorithm 1;
  T is the number of iterations;
  η is the step size.
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 parameter.
    $w^{t+1} = \frac{\sum_{j=1}^{N_s} D_j w^{t+1}}{D}$.
  end for

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 parameter w_j^{t+1} according to the update rule as follows:

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

where $\eta \geq 0$ is the gradient step size. After T iterations, the global parameter w^{t+1} is 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 parameter w^{t+1} is used to take part in 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,

- $f_i(w)$ is convex;
- $f_i(w)$ is L-smooth, i.e., $f_i(w') \leq f_i(w) + \nabla f_i(w) \cdot (w' - w) + \frac{L}{2}\|w - w'\|^2$, 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 $w_* = argmin_w f(w)$; thus, we have

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

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

Proof: Please see Appendix B. ■

We have the convergence in expectations:

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

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

Algorithm 3 FADE: Federated DRL-based edge caching.

Initialization:
$w_j^t = w_i^t$;
w_i^t is the pretrained parameter from Algorithm 1;
T is the number of iterations;
η is the step size.
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 parameter according to:
$w^{t+1} = \frac{\sum_{j=1}^{N_s} D_j w^{t+1}}{D}$.
12: Input the global parameter:
$w_i^{t+1} = w^{t+1}$.
end for
14: BS n disperses w_i^{t+1} to UEs.
end for

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

After receiving the parameters from all the local UEs, local BS n aggregates all the distributed parameters (Lines 10-12) and then disperses the updated global parameter 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: distributed training procedure at UEs and 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 meters. 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, and the channel bandwidth is 20 MHz [46]. In addition, the transmit power of the BS is $40W$, which supports at most 300 UEs. The double DQN consists of a fully connected feed forward neural network with a 1 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-SP
$\mathcal{M}$	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

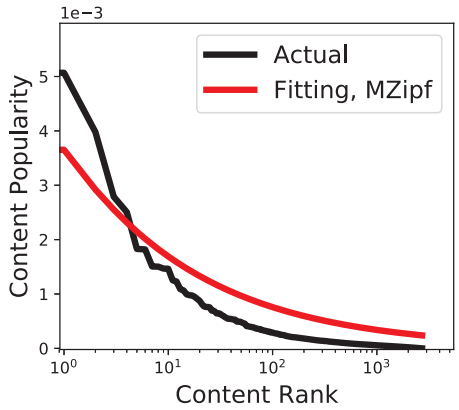
Moreover, we use a large-scale offline MSN real-world dataset derived from an application $Xender$. Xender is widely used in India to share content. The date of experimental data we used is from $01/08/2016$ to $31/08/2016$, including $450,786$ trajectories of UEs, over $153,482$ files are shared, and the number of requests is $271,785,952$ [33]. To implement the experiments more practically, we obtain the global and local content popularity from the real-world datasets.

As shown in Fig. **??**, 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 [16], considering the distribution of differential scenario properties (e.g., user numbers/user requests, mobility, content popularity, etc.), we include different local content popularity for 4 local BSs by various parameters in Fig. **??**.

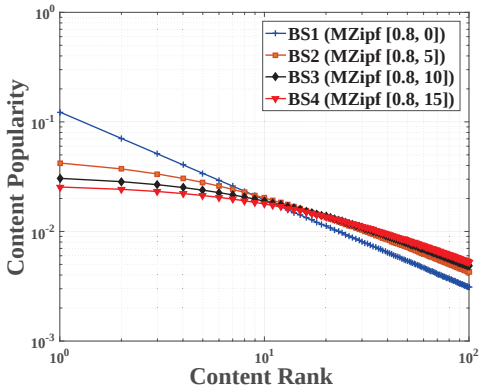
B. Evaluation Results

In this section, we first present the loss function performance between the traditional DRL and FADE. Then, the statistics of 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. Some state-of-the-art caching algorithms are shown as follows: including the LRU, LFU and FIFO, as well as Oracle [52], which is conducted as the baseline of best network performance.

First, we demonstrate the loss function between FADE and centralized DRL training, shown as Fig. 7. The results are derived from 100 times the average. 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



(a) Global content popularity.



(b) Local content popularity.

Fig. 6. Global and local content popularity.

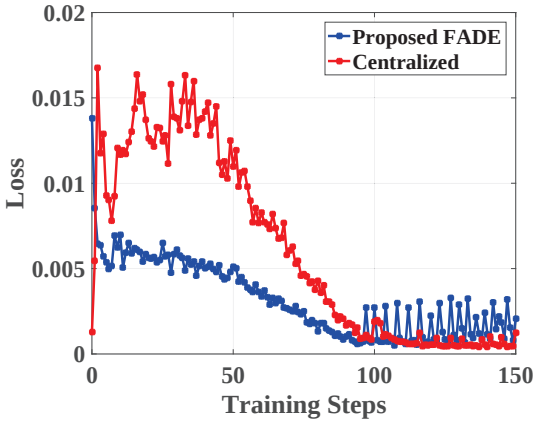


Fig. 7. Demonstration of the loss function between FADE and traditional centralized DRL.

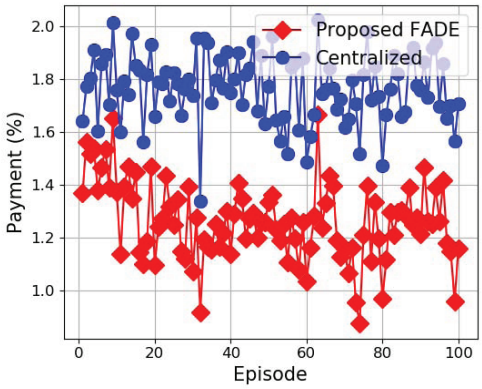


Fig. 8. Demonstration of system payments between FADE and traditional centralized DRL.

DRL. This finding indicates that our algorithm achieves better performance in terms of stability and has quick convergence. 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 the proposed FADE and the centralized algorithm. The system payment here refers to the ratio of offloaded content to downlink data rate at each episode, indicating the network overhead of information exchange. From Fig. 8, the proposed 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 the proposed 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.

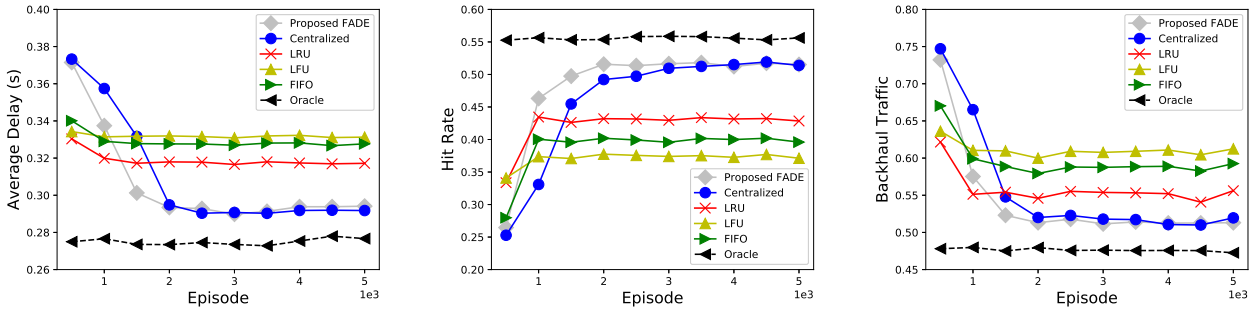
From Fig. 9(a), the proposed FADE algorithm shows the high value of average delay at first, However, after decreasing rapidly after 2 episodes, the value is maintained in a relatively stable state. The proposed FADE algorithm outperforms the other algorithms; it achieves the lowest average delay of $0.29s$, 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% to 75%, and outperform the LRU, LFU and FIFO algorithms by up to 5%, 20% and 15%, respectively.

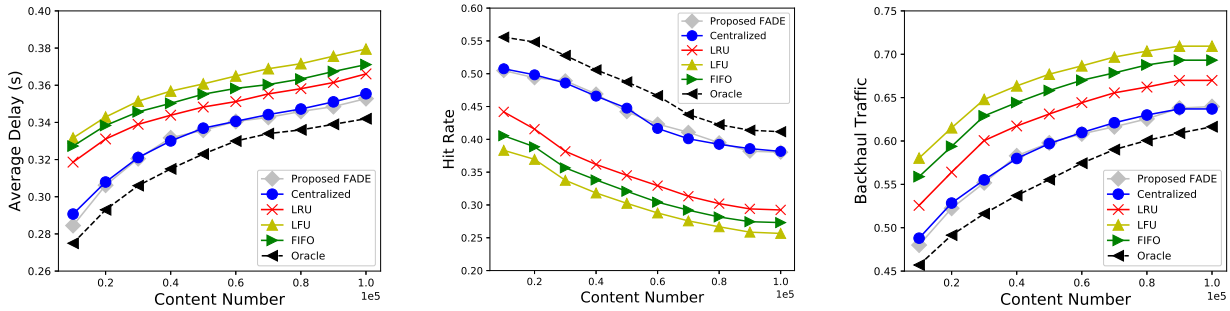
Specifically, the proposed FADE algorithm outperforms the traditional centralized algorithm in the first two episodes and then achieves almost the same performance because the system reward is used to reduce the content access delay of UEs, which makes the proposed FADE minimize the average delay.

Fig. 10 demonstrates the network performance under different content numbers. The content number ranges from $10,000$



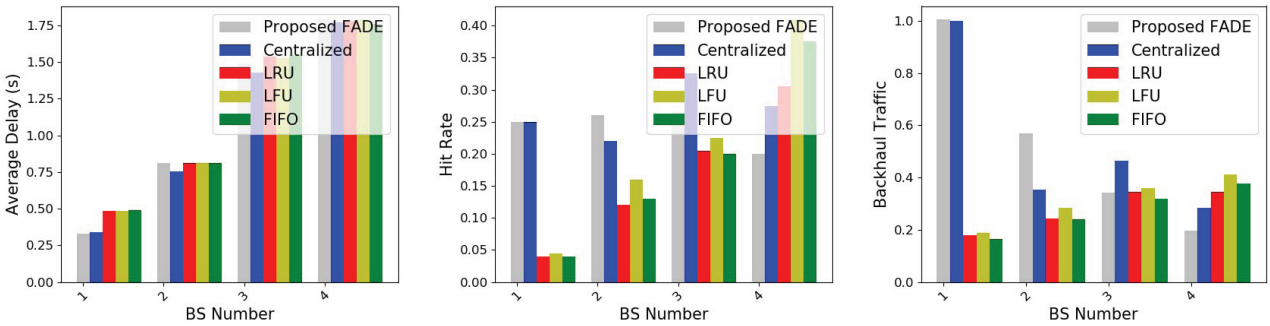
(a) Performance comparison of the average delay. (b) Performance comparison of the hit rate. (c) Performance comparison of traffic.

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



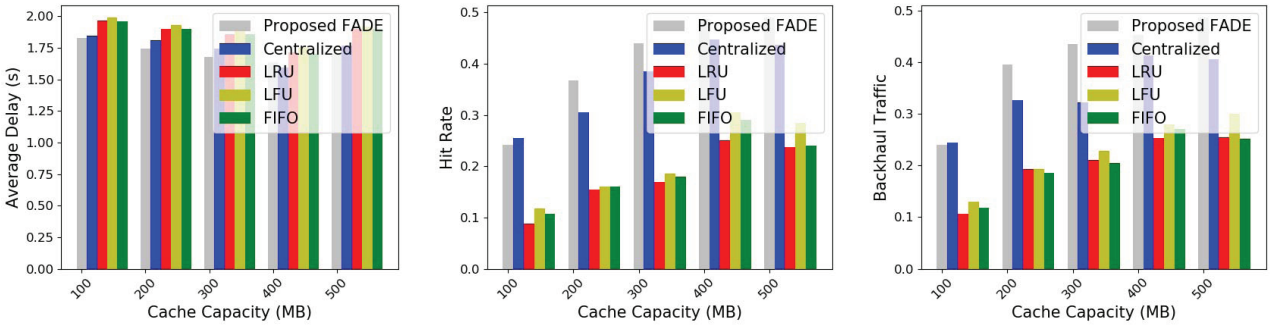
(a) Average delay with different content numbers. (b) Hit rate with different content numbers. (c) Backhaul traffic with different content numbers.

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



(a) Average delay with different BS numbers. (b) Hit rate with different BS numbers. (c) Backhaul traffic with BS numbers.

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



(a) Average delay with different cache capacities. (b) Hit rate with different cache capacities. (c) Backhaul traffic with different cache capacities.

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

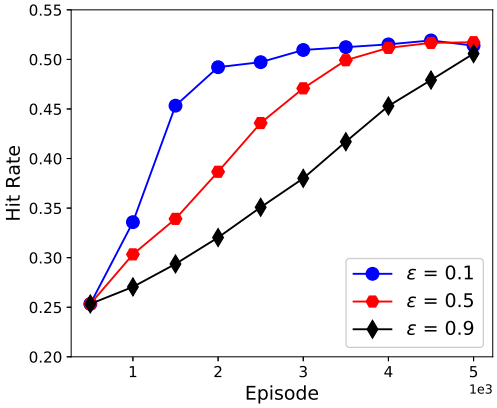


Fig. 13. Performance of the hit rate under different exploration probabilities.

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

to $100,000$, and the cache capacity is set as $C = 100$ MB. According to Fig. 10(a)-10(c), the proposed algorithm achieves the average 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 line trends in Fig. 10 and Fig. 9 are exactly the opposite because more popular contents will be cached under the fixed cache capacity of BS along with the increasing content number, leading to the rising trend in Fig. 10(a) and Fig. 10(c). The decreasing trend in Fig. 10(b) is caused by the new 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 the proposed 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 proposed 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 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 the proposed 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 the proposed FADE achieves better performance in terms of hit rate and backhaul traffic offload. The similar reason with that of Fig. 11(a), leading to the poor delay performance. From Fig. 12(b), the proposed FADE improves the hit rate performance by 5%, 27%, 25% and 26%, compared to the centralized LRU, LFU and FIFO algorithms on average. The proposed 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 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 content, replacement processes rarely occur.

The aforementioned simulation results verify that the proposed 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 a loss, the transmission delay is ignored [30] [47]. However, it is impossible in practical scenarios, which further proves the efficiency of the proposed FADE.

We evaluate the proposed 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, the number of BS is 4.

Fig. 13 shows the performance comparison for the proposed 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. 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 the proposed

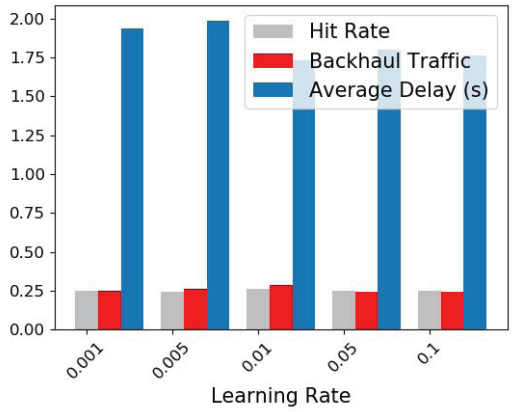


Fig. 15. Performance of the hit rate, backhaul traffic, and average delay under different learning rates.

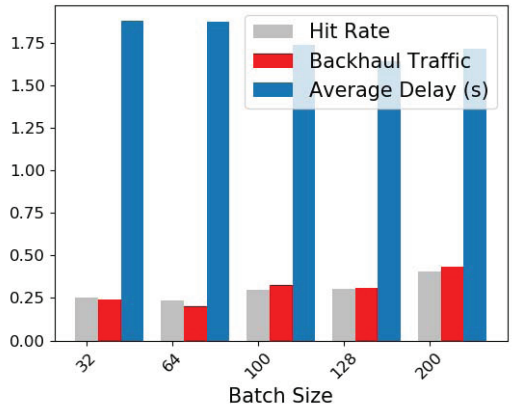


Fig. 16. Performance of the hit rate, backhaul traffic, and average delay under different batch sizes.

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 optimality of this value, and γ = 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 $alpha$ = 0.05 as an empirical value to maintain the stability and effectiveness of our algorithm.

Fig. 16 shows the network performance demonstration 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 paper, 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 on the premise of a low loss function.

Appendix

A. Proof of Lemma 1

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

B. Proof of Theorem 1

Proof: First, we prove the μ-strongly convexity of $f_(w)$: Given $\forall w, w' \in \mathbb{R}$, $\beta \in [0,1]$, assume that $x := w + w'$, $y := \beta w + (1+\beta)w'$ and $\exists \beta_1, \beta_2 \in (0,1)$. Derived from the Taylor formula, we can obtain

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

and

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

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

$$\begin{aligned} &\beta f(w) + (1-\beta) f(w') \\ &= f(y) + \tfrac{1}{2}\beta(1-\beta)(w-w')^2[(1-\beta)\nabla^2 f(e_1) + \beta\nabla^2 f(e_2)]. \end{aligned} \quad (29)$$

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

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

Thus, we rewrite (29) as

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

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

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

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

Thus, we can obtain the following:

$$\begin{aligned} &\|w_{t+1} - w_*\|^2 = \|w_t - \eta\nabla f(w_t) - w_*\|^2 \\ &= \|w_t - w_*\|^2 - 2\eta\nabla f(w_t)(w_t - w_*) + \eta^2\|\nabla f(w_t)\|^2 \\ &\leq \|w_t - w_*\|^2 - 2\eta(f(w) - f(w_*) \\ &+ \frac{\mu}{2}\|w_t - w_*\|^2) + \eta^2\|\nabla f(w_t)\|^2. \end{aligned} \quad (33)$$

Note that by smoothing $f(w)$, we can obtain the gradient bound:

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

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

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

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 WCNC*, pp.1-6, Apr. 15-18, Marrakesh, Morocco, 2019.

[3] L. Zeng, E. Li, Z. Zhou, et al, "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, 2019.

[4] K. Samdanis, T. Taleb, and S. Schmid, "Traffic Offload Enhancements for eUTRAN," *IEEE Commun. Surveys & Tutorials.*, vol. 14, no. 3, pp. 884-896, Third Quarter 2012.

[5] 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, Jan. 2018.

[6] M. Dai, H. Deng, B. Chen, G. Su, X. Lin, H. Wang, "Design of Binary Erasure Code with Triple Simultaneous Objectives for Distributed Edge Caching in Industrial IoT Networks," *IEEE Trans. Ind. Inform.*, pp. 1-1, Nov 13, 2019.

[7] S. Ajmal, MB. Muzammil, A. Jamil, SM. Abbas, U. Iqbal, P. Touseef, "Survey on Cache Schemes in Heterogeneous Networks using 5G Internet of Things," in *Proc. ACM ICFNDS*, pp. 1-8, Jul 1, Paris France, 2019.

[8] H. Wei, H. Luo, Y. Sun, MS. Obaidat, "Cache-Aware Computation Offloading in IoT Systems," *IEEE Systems J.*,vol. 14, no. 1, pp. 61-72, May 14, 2019.

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

[10] T. A. Do, S. W. Jeon, W. Y. Shin, "How to cache in mobile hybrid IoT networks?", *IEEE Access*, vol. 7, pp. 27814-28, Mar 4, 2019.

[11] Y. Han, R. Wang, J. Wu, "Random Caching Optimization in Large-scale Cache-Enabled Internet of Things Networks," *IEEE Trans. Netw. Sci. Eng.*, pp. 385-397, Jan 18, 2019.

[12] Apple Inc., "Apple Edge Cache," *URL:https://cache.edge.apple/*, 2020.

[13] X. Li, X. Wang, K. Li, Z. Han, et al., "Collaborative multi-tier caching in heterogeneous networks: modeling, analysis, and design," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 10, pp. 6926-6936, Oct. 2017.

[14] T. Wang, G. Zhang, A. Liu, MZ. Bhuiyan, 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, Sep. 13 2018.

[15] X. Wang, M. Chen, Z. Han, et al., "TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks," in *Proc. IEEE INFOCOM*, pp. 2346-2354, Apr. 27-May. 2, Toronto, Canada, 2014.

[16] X. Wang, M. Chen, T. Taleb, et al. "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.

[17] 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.

[18] 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.

[19] Li, Xiuhua, et al. "Collaborative multi-tier caching in heterogeneous networks: Modeling, analysis, and design," *IEEE Trans. Wirel. Commun.*, vol. 16, no. 10, pp. 6926-6939, Aug. 4, 2017.

[20] N. Golrezaei, et al., "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402-8413, Dec. 2013.

[21] 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*, pp. 319-332, Jun. 25-28, Taipei, Taiwan, China, 2013.

[22] X. Li, X. Wang, Xiao S, and V. C. M. Leung, "Delay performance analysis of cooperative cell caching in future mobile networks," in *Proc. IEEE ICC*, pp. 5652-5657, May 10-14, Sydney, Australia, 2015.

[23] S. H. Chae, J. Y. Ryu, T. Q. S. Quek, and W. Choi, "Cooperative transmission via caching helpers," in *Proc. IEEE GLOBECOM*, pp. 1-6, Dec. 6-10, San Diego, CA, USA, 2015.

[24] X. Thang, S. Chatzinotas, and B. Ottersten, "Edge Caching Wireless Networks: Energy-Efficient Design and Optimization," *IEEE Trans. Wirel. Commun.*, vol. 99, pp.1-1, 2017.

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

[26] H. Zhu et al. "Deep Reinforcement Learning for Mobile Edge Caching: Review, New Features, and Open Issues." *IEEE Netw.* vol. 32, no.6, pp. 50-57, Nov. 2018.

[27] S. Y. Mulya, et al. "Distributed Deep Learning at the Edge: A Novel Proactive and Cooperative Caching Framework for Mobile Edge Networks," *IEEE Wirel. Commun. Lett.*, vol. 8, no. 4, pp. 1220-1223, Apr. 2018.

[28] X. Fan, et al. "Exploiting the edge power: an edge deep learning framework," *CCF Trans. Netw.* vol. 2, no. 1, pp. 4-11, Jun. 2018.

[29] H. Pang et al. "Toward smart and cooperative edge caching for 5g networks: A deep learning based approach," in *Proc. IWQoS*, Jun. 4-6, Banff, AB, Canada, 2018.

[30] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, 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, Oct 16 2018.

[31] Y. Yang, L. Wu, G. Yin, L. Li, H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no.5, pp. 1250-8, Apr 2017.

[32] M. Zhang, J. Chen, S. He, L. Yang, X. Gong, J. Zhang, "Privacy-Preserving Database Assisted Spectrum Access for Industrial Internet of Things: A Distributed Learning Approach", *IEEE Trans. Ind. Electron.*, pp.1-1, Sep 5 2019.

[33] 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, Jun. 2018.

[34] 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. Mob. Comput.*, vol. 17, no. 8, pp. 1791-1805, 2017.

[35] S. Vural, P. Navaratnam, N. Wang, C. Wang, L. Dong, and R. Tafazolli, "In-network caching of Internet-of-Things data," *Proc. IEEE ICC*, pp. 3185-3190, Sydney, NSW, Australia, 2014.

[36] S. Zhang, J. Liu, and L. Jiajia, "Optimal Probabilistic Caching in Heterogeneous IoT Networks," *IEEE Internet Things J.*, Jan 27, 2020.

[37] J. Gu, W. Wang, A. Huang, et al., "Distributed cache replacement for caching-enable base stations in cellular networks," *in Proc. IEEE ICC*, pp. 2648-2653, Jun. 2014.

[38] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," *in Proc. IEEE ICC*, pp. 1897-1903, Jun. 2014.

[39] B. Chen, L. Liu, M. Sun, H. Ma, "IoTCache: Toward Data-Driven Network Caching for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10064-76, Aug 15, 2019.

[40] 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*, pp. 38-43, Oct. 22, Dallas, TX, USA, 2003.

[41] 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.

[42] H. Mohamed, 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, Mar. 2008.
[43] 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.
[44] M. Volodymyr, et al. "Human-level control through deep reinforcement learning," *Nature* vol. 518, no. 7540, pp. 529, Feb. 2015.
[45] H. V. Hasselt, A. Guez, D. Silver, "Deep Reinforcement Learning with Double Q-learning," in *Proc. AAAI*, pp. 2094-2100, Feb. 12-17, Phoenix, Arizona USA, 2016.
[46] 3GPP, "Further advancements for E-UTRA physical layer aspects (release 9)," TR 36.814 V1.2.0, Jun. 2009.
[47] S. Shen, Y. Han, X. Wang, Y. Wang, "Computation Offloading with Multiple Agents in Edge-Computing-Supported IoT," *ACM Trans. Sens. Netw.*, vol. 16, no. 1, pp. 1-27, Dec 19, 2019.
[48] B. Chen, L. Liu, M. Sun, H. Ma, "IoTCache: Toward Data-Driven Network Caching for Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10064-76, Aug 15, 2019.
[49] Konecny, Jakub, et al. "Federated Learning: Strategies for Improving Communication Efficiency," in *Proc. NIPS*, Dec. 5-10, Barcelona, Spain, 2016.
[50] H. B. Mcmahan, E. Moore et al. "Communication-Efficient Learning of Deep Networks from Decentralized Data" *arXiv preprint*, 1602.05629, 2016.
[51] Wang S, Tuor T, Salonidis T, et al. "Adaptive federated learning in resource constrained edge computing systems", in *Proc. IEEE INFOCOM*, pp. 8-9, Apr. 15-19, Honolulu, HI, USA, 2018.
[52] S. Müller, O. Atan, et al., "Context-aware proactive content caching with service differentiation in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 1024-1036, Feb. 2017.