IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON MOBILE SERVICE COMPUTING WITH INTERNET OF THINGS

Received April 15, 2019, accepted May 3, 2019, date of current version June 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919736

Federated Learning-Based Computation Offloading Optimization in Edge Computing-Supported Internet of Things

JIANJI REN, HAICHAO WANG, TINGTING HOU, SHUAI ZHENG, AND CHAOSHENG TANG

College of Computer Science and Technology (Software College), Henan Polytechnic University, Jiaozuo 454010, China

Corresponding author: Chaosheng Tang (tcs@hpu.edu.cn)

ABSTRACT Recently, smart cities, smart homes, and smart medical systems have challenged the functionality and connectivity of the large-scale Internet of Things (IoT) devices. Thus, with the idea of offloading intensive computing tasks from them to edge nodes (ENs), edge computing emerged to supplement these limited devices. Benefit from this advantage, IoT devices can save more energy and still maintain the quality of the services they should provide. However, computational offload decisions involve federation and complex resource management and should be determined in the real-time face to dynamic workloads and radio environments. Therefore, in this work, we use multiple deep reinforcement learning (DRL) agents deployed on multiple edge nodes to indicate the decisions of the IoT devices. On the other hand, with the aim of making DRL-based decisions feasible and further reducing the transmission costs between the IoT devices and edge nodes, federated learning (FL) is used to train DRL agents in a distributed fashion. The experimental results demonstrate the effectiveness of the decision scheme and federated learning in the dynamic IoT system.

INDEX TERMS Federated learning, computation offloading, IoT, edge computing.

I. INTRODUCTION

IoT devices are widely used in industrial control, network physical system, public safety equipment, environmental monitoring and other fields. Large-scale IoT devices will be deployed everywhere in the future to meet the growing demand for services such as smart cities, smart homes and smart medical systems. These devices typically require low latency and power consumption to perform tasks such as monitoring, sensor data upload, and real-time decision making.

Typically, these IoT devices are relatively weak and heterogeneous, and of course they are unlikely to directly support the intensive computational costs caused by the above tasks. However, as an emerging technology, edge computing is envisioned as a promising architecture for offloading tasks from IoT devices. On the other hand, edge nodes in edge computing systems are used as coordinators between them and are responsible for their communication and even load balancing.

Therefore, in this work, we use FL to guide the training of DRL agents for joint allocation of communication and computing resources. Experimental results confirm its effectiveness compared to unrealistic centralized training methods. The main contributions of this paper include: 1) The combination of DRL training and FL in the IoT system supported by edge computing is studied. 2) Joint allocation of communication and computing resources. 3) Experiments verify the effectiveness of the FL-based DRL.

The structure of this paper is as follows. We give related work in Section II. The Section III describes the system architecture and dynamic system model. The Section IV gives the problem formulation and the training frame. We provide simulation experiments and analysis in Section V. Finally, Section VI concludes our work.

II. RELATED WORK

To reduce the latency of IoT devices, Qiu *et al.* proposed EARS, which uses packet priority and expiration dates to describe the urgency of the packet [14]. Through the analysis of the arrival process of different data packets, the back pressure queue model with emergency package is designed, an event-aware back-pressure scheduling scheme (EABS) for EIoT is proposed [17]. Qiu *et al.* proposed a new type of spider network transmission mechanism for emergency data in a vehicle-mounted self-organizing network [10].

The associate editor coordinating the review of this manuscript and approving it for publication was Tie Qiu.

In addition, we should pay special attention to the security and privacy of IoT devices. A spam authentication scheme based on Gaussian Mixture Model (SIGMM) was proposed in [12], which is applied to machine learning recognition of mobile networks.

Other than this, edge computing can also be used to pre-cache popular content. In HetNets, an effective cooperative multi-layer caching framework is proposed, and the maximum capacity of network infrastructure to offload network traffic locally and support user content requests is discussed [16]. In addition, advanced device to device (D2D) communications are available to reduce congestion when using cellular networks. Wang *et al.* designed a big data platform named device-to-device, which effectively promoted the use of wireless network between users to achieve device-to-device communication, accurately provided content to users and effectively provided intelligent discharge for operators [8]. Li *et al.* considered the social behavior and preference of mobile users, heterogeneous cache size and the analysis of the derived system topology [4]. Wang *et al.* proposed a framework for auxiliary traffic offloading of social network services (SNS), which offload traffic of social network services to users through opportunity sharing (TNS) in mobile social networks (MSN), so as to achieve user sharing [6].

With edge computing, IoT devices can offload their intensive tasks to edge nodes, leveraging the trade-offs between communication and computing, giving them the potential for energy savings and performance enhancement. Therefore, it is very necessary to propose a useful and efficient scheduling method. In the past, related work proposed to use convex optimization [3] and game theory [5] to solve the resource allocation problem in the offloading. However, these traditional methods require accurate channel state information or global information of devices and edge nodes, and it is impractical to obtain such information from a complex environment. In addition, computational offloading involves comprehensive resources for wireless communications, computing, and networking, and these methods are not easily changed when resource constraints or objective functions change to address related but different optimization issues.

Wang *et al.* recommended combining Deep Reinforcement Learning and Federated Learning frameworks with mobile edge systems to optimize mobile edge computing, caching and communication [2]. Therefore, deep reinforcement learning is used in [7] to deal with the allocation of integrated resources in computational offloading to maximize the long-term benefits of energy consumption and execution delays, without the need to know the channel state information in advance, only local information is needed. Integrated resource allocation is achieved without global information. Particularly, this optimization can solve the following problems: 1) Uncertain Inputs: due to privacy issues and dynamic changes in the radio channel, it is difficult to obtain some key information necessary for model-related optimization. 2) Dynamic Conditions: the entire edge system should be considered real-time dynamic workload. 3) Temporal Isolation: not only to optimize the snapshot of the system, but also to consider the long-term utility of the system. However, an unnoticed assumption has been made in [7]. Specifically, IoT device is considered as a very powerful device with the ability to independently train their own deep reinforcement learning agents. However, in the near future, IoT devices may not be as powerful, they may only support computing of lightweight neural networks at most.

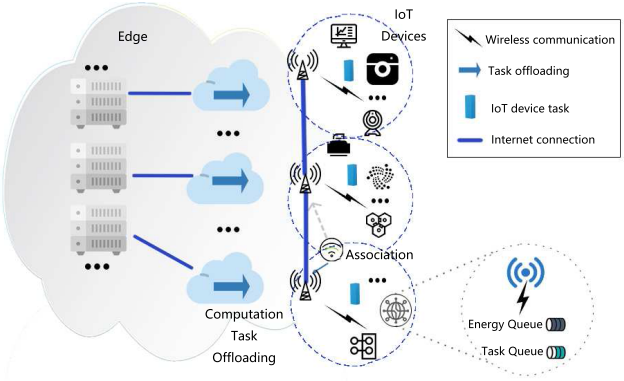


FIGURE 1. IoT system with edge computing nodes.

III. SYSTEM ARCHITECTURE

A. STATIC SYSTEM MODEL

In this paper, a system model in IoT environment with edge nodes is taken for analysis, as shown in Fig.1. The IoT devices in this model have the capability of energy harvesting, they can harvest energy units from edge nodes and store them in their energy queue. Based on this system model, edge nodes (ENs) providing both communication and computation offloading for IoT devices, $\mathcal{I} = \{1, \ldots, I\}$ is the set of IoT devices, and $\mathcal{E} = \{1, \ldots, E\}$ is the set of edge nodes (ENs). One edge node out of $\mathcal{E}$ can be chosen by each device to establish communication and offload intensive computation tasks with allocated frequency bandwidth W Hz. For quantitative analysis, time horizon is discretized into time epochs indexed by i with equivalent duration as ζ (in seconds).

One IoT device is taken as a typical one for illustrate this model, the device in this model has the following capabilities: 1) energy units from edge nodes can be harvested and stored in an energy queue with a maximum length l^e_{max} for wireless transmission and computation; 2) computation tasks for performing specific services is always be admitted, and these generated tasks constitute an Independent and Identically Distributed (I.I.D.) sequence of Bernoulli random variables with a common parameter $\gamma^t \in [0, 1]$. If a task is generated during an epoch i, the task arrival indicator for the device at epoch i is represented as $\beta^t_i = 1$ and otherwise $\beta^t_i = 0$; 3) there is a local task queue with a maximum length l^t_{max} inside the IoT device, and it can maintain unprocessed and not successfully processed tasks for later processing in the manner of First In First Out (FIFO); 4) the device can establish a connection with the edge node for uploading updates

and offloading computation tasks, downloading model parameters.

As for the computation task, a computation task taken from the task queue can be determined for execution locally on the IoT device or offloading to an EN for processing. A joint action (j_i, u_i) at the beginning of each epoch i should be made for make a decision on: 1) whether the task should be processed locally ($j_i = 0$) or offloaded to an EN ($j_i \in \mathcal{E}$), noted that ($j_i \in \{0\} \cup \mathcal{E}$); 2) how many energy units ($u_i \in \mathbb{N}_+$) stored in the energy queue should be allocated.

$$r_i = \sqrt{u_i/(\omega \cdot \mu)} \leq r^c_{max}, \tag{1}$$

If the computation task is determined for execution locally on the IoT device. The computation task should be modeled as (d, μ), of which d (in bits) and μ represent the transmission data size required for offloading a task and the needed number of CPU cycles for processing the task. In this circumstance, when a computation task is allocated to be processed locally with permitted energy units u_i (if there is any), viz., $j_i = 0$, the allocated CPU frequency r_i for this task can be modeled with a maximum limitation as above. Here ω is the commonly adopted effective switched capacitance that depends on the architecture of chips [1]. Then, the corresponding time consumption for the local task execution is

$$t^m_i = \mu / r_i \tag{2}$$

If the IoT device decides to associate with an EN, and a radio link is established for them. The radio channel quality between them should be considered, since it directly affects the wireless communication particularly the transmission rate. The achievable data rate can be calculated as follow, where A is the power of interference plus noise.

$$v_i = W \cdot log_2(1 + s^e_i \cdot f^{tr}_i / A) \tag{3}$$

The channel gain between the IoT device and an EN $e \in \mathcal{E}$ is denoted as s^e_i during the epoch i, which is assumed static and independently taken from a finite state space S_e. The f^{tr}_i is the transmit power with maximum limitation f^{tr}_{max}, which satisfies

$$f^{tr}_i = u_i / t^{tr}_i \leq f^{tr}_{max} \tag{4}$$

At last, the total time for transmitting the input data d is

$$t^{tr}_i = d / v^i \tag{5}$$

Given the association $j^i \in \mathcal{E}$ and the allocated energy units $u_i > 0$ at an epoch i, the transmitting rate should remain a constant for achieving the minimum transmission time, which is preferred in practical according to the proof in [4]. Finally, the minimum transmission time can be derived by solving simultaneous equations of (3), (4) and (5) as follow. It should be noted that the processing delay on the EN is assumed to be much less compare to the transmission time when the IoT decides to offload.

$$\log_2(1 + s^{j_i}_i \cdot u_i \cdot (A \cdot t^{tr}_i)^{-1}) = d \cdot (W \cdot t^{tr}_i)^{-1} \tag{6}$$

B. DYNAMIC SYSTEM MODEL

In the scene that changes in real-time, the energy queue and task queue that represent the computation resource and the workload should be particularly focused. We use l^u_i to represent the energy queue length insider the IoT device at the beginning of an epoch i, its dynamics can be described follow. Where $\beta^u_i \in \mathbb{N}_+$ is the total number of energy units received till the end of epoch i.

$$l^u_{i+1} = min\{l^u_i - u_i + \beta^u_i, l^u_{max}\} \tag{7}$$

With the available energy units provided by the energy queue, the achievable task execution delay, which includes both communication and computation, is the main concerns. Besides the processing delay of a task and the transmission delay, the handover delay is also considered, the task execution delay can be expressed as follow. The delay of EN-side execution is t^s, which is relatively a small constant as aforementioned, the handover delay resulting from altering EN association is o_i.

$$t_i = \begin{cases} t^m_i & \text{if } u_i > 0 \text{ and } j_i = 0 \\ o_i + t^{tr}_i + t^s & \text{if } u_i > 0 \text{ and } j_i \in \mathcal{E} \\ 0 & \text{if } u_i = 0 \end{cases} \tag{8}$$

Specifically, the failure contents task processing and offloading is also taken into consideration. In more detail, the task execution will be deemed as a failure and thus remain in the task queue till being successfully executed in two circumstances, viz., 1) a computation task can not be processed by the IoT device even until the end of an epoch; 2) the IoT device decide to offload a task to a specific EN, but it fails owing to the long time transmission introduced by either not enough allocated energy units or bad radio channel quality.

$$l^t_{i+1} = min\{l^t_i - \mathbf{1}_{\{0<t_i<\zeta\}} + \beta^t_i, l^t_{max}\} \tag{9}$$

The dynamics task queue length can be calculated as above for convenient expression. Certainly, the new generated tasks must be dropped, if the task queue is full of awaiting tasks, which shall be avoided in the ideal case. Then the number of computation task drop in an epoch i can be described as

$$\xi_i = max\{l^t_i - \mathbf{1}_{\{0<t_i<\zeta\}} + \beta^t_i - l^t_{max}, 0\} \tag{10}$$

However, not every task can be successfully handled in one epoch ζ, the unwished queuing delay for computation tasks will be incurred. The length l^t_i of the task queue inside the IoT device is treated as the queuing delay ε_i during the epoch i, which is

$$\varepsilon_i = l^t_i - \mathbf{1}_{\{t_i>0\}} \tag{11}$$

$$\sigma_i = \mathbf{1}_{\{t_i>\zeta\}} \tag{12}$$

But if the execution of a computation task fails, corresponding penalty σ_i will be given as above. More over, the fee for occupying the EN shall be paid by the IoT device if it decides to offload its computation tasks to the EN. Such payment is product by the time consumed for the EN receiving and processing the task input data. $\pi \in \mathbb{R}_+$ is defined as the

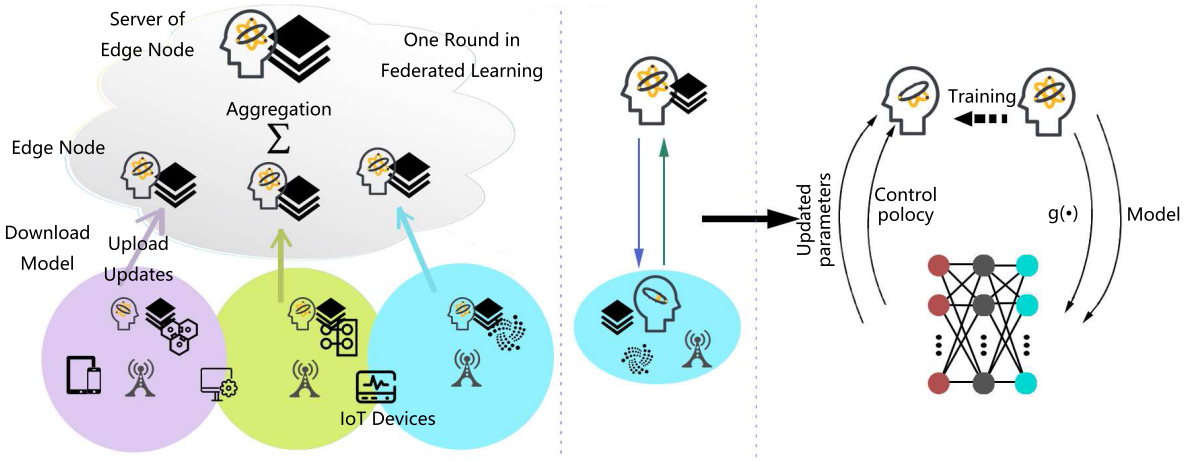


FIGURE 2. Training of computation tasks offloading based on Federated Learning DRL.

price paid per unit of time, the payment expression φ_i can be written as

$$\varphi_i = \pi \cdot (min\{t_i, \zeta\} - o_i) \cdot \mathbf{1}_{\{j_i \in \mathcal{E}\}} \tag{13}$$

IV. POLICY TRAINING BASED ON FEDERAL LEARNING

A. PROBLEM FORMULATION

After the system description in Sec.III.A and Sec.III.B, the optimization problem is be expound in this section. Collecting all essential elements for organizes the network state Y_i of the IoT device.

$$Y_i = (l_i^t, l_i^u, j_i, s_i) \in \boldsymbol{Y} \stackrel{def}{=} \left\{0, 1, \cdots, l_{max}^t\right\} \times \left\{0, 1, \cdots, l_{max}^u\right\} \times \mathcal{E} \times \{Y_{e \in \mathcal{E}} S_e\} \tag{14}$$

The $s_i = (s_i^e : e \in \mathcal{E})$ is the channel gain between the IoT device and an EN. At the beginning of epoch i. A decision made by the IoT device on where to process the computation task and how many energy resources should be allocated, i.e.,

$$(j_i, u_i) \in \mathcal{T} \stackrel{def}{=} \{\{0\} \cup \mathcal{E}\} \times \left\{0, 1, \cdots, l_{max}^u\right\} \tag{15}$$

A sequence of the above actions should be determined by an optimal control policy Φ to maximize the expected long-term utility as

$$G(Y, \Phi) = \mathbb{E}_{\Phi}\left[\lim_{M \to \infty} \frac{1}{M} \cdot \sum_{i=1}^{M} g(Y_i, \Phi(Y_i)) \mid Y_1 = Y\right] \tag{16}$$

where Y_i is the initial network state, and $g(\cdot)$ is the immediate utility at epoch i defined as the customized combination operation of task execution delay t_i, the task queuing delay ε_i, the payment φ_i, the number of task drop ξ_i and the penalty of execution failure σ_i. To be noted, the summarized utility can be designed for different objection. For instance, if the system regards the no-failure characteristic as the most important one, the penalty of execution failure σ_i can be amplified to enhance its ratio in the whole utility.

B. REASONS FOR CHOOSING FEDERAL LEARNING

In Sec.III, we take the single IoT device as an example for interpretation. In this section, we will show the merits of using FL to coordinate the training process among multiple IoT devices. This kind of problem can deal with well by DRL techniques, thus we use Double Deep Q Learning (DDQN) [11], [15] for each IoT device to maximize long-term utility of its control policy. DRL techniques is efficient in finding the optimal policy in the dynamic edge system, but it also demands abundant computation resources. Therefore, the deployment of DRL agent should be carefully thought over.

On one hand, it will bring about three disadvantages if the DRL agent is trained on the EN: 1) it may jeopardize the privacy, since the uploaded training data might be privacy sensitive, especially in the scenario of industrial informatics; 2) though the training data can be transformed for privacy protection, the proxy data received by the EN is less relevant and loose the pertinence for a specific IoT device; 3) massive training data will be always transmitted from IoT devices to the EN, and consequently burden the wireless channel.

On the other hand, two deficiencies are still remain if the DRL agent is trained on the IoT device individually: 1) extra energy wasting will be caused by the standalone training of separate DRL agents; 2) it will consume long time or even impossible to train each DRL agent well from scratch.

Hence, as depicted in Fig.2, taking efficiently training DRL agent in a distributed manner into consideration is a natural choice. Although it can realize the best performance that the DRL agent trained in every IoT device or the EN, it is also practical to adopt distributed DRL training. In addition, due to the privacy issues in edge computing system and the networking constrains coupled with the challenge of handling non-I.I.D data, most of efficient distributed deep learning techniques [9] are not feasible. For these reasons, FL is introduced in this work for distributively training DRL agents.

C. TRAINING OF TASKS OFFLOADING WITH FEDERAL LEARNING

In the computation offloading scenario, a control action should be taken by each IoT according to the dynamic of networking state and its own workload. Hence, we proposed to use the EN for coordinating massive IoT devices covered by it. IoT devices can be able to maintain a complex DRL agent with relatively less computation burden by taking advantage of this scheme.

Rely on the description of Algorithm 1, FL iteratively selects a random set of IoT devices to 1) download parameters of the DRL agent from the EN; 2) use their own data to perform the training process on the upgraded (downloaded) model; 3) upload only updated model parameters of the DRL agent to the EN for model aggregation.

FL enables resource-limited IoT devices to learn a shared DRL agent without centralizing the training data, which can be extended to several more particular benefits by this means in our system. EN-side proxy data is less relevant to the local data in the IoT device. In the envisioned IoT system, various and localized sensing data can acquired for updating the DRL agent by massive IoT devices. These data may include the workload of both IoT devices, the remained energy resources, the channel gain of the radio channel and ENs and etc.. In generally, FL can leverage these localized data to make the whole IoT system more cognitive.

Algorithm 1 Framework of Ensemble Learning for Our System

Initialization:
With respect to the global DRL agent in the EN:
    Initialize the DRL agent with random weights θ_0;
    Initialize the gross training times A_0 of all devices;
With respect to each IoT device $I \in \mathcal{I}$:
    Initialize the experience replay memory $\mathcal{M}_0^I$;
    Initialize the local DRL model θ_0^I;
    Download θ_0 from the EN and let $\theta_0^I = \theta_0$;
Iteration:
For each round $t = 1$ **to** T **do**;
$S_t \leftarrow$ { random set of m available IoT devices};
  For each device $I \in S_t$ in parallel do;
    Fetch θ_t from the EN as let $\theta_t^I = \theta_t$;
    Sense and update $\mathcal{M}_t^I$;
    Train the DRL agent locally with θ_t^I on $\mathcal{M}^I$;
    Upload the trained θ_{t+1}^I to the EN;
    Notify the EN the times A_t^I of local training;
  End For
  With respect to the EN:
    Receive all model updates;
    Refresh the statistical $A_t = \sum_{I \in S} A_t^I$;
    Perform model aggregation as:
      $\theta_{t+1} \leftarrow \sum_{I \in S} (A_t^I / A_t) \cdot \theta_{t+1}^I$;
End For

V. EXPERIMENTAL EVALUATION

A. TRAINING PERFORMANCE UNDER FIXED TASKS PROBABILITY

We investigate an edge system with E = 6 ENs and I = 15 IoT devices to evaluate the capabilities of our proposed method. According to the quality of the wireless channel, the channel gain is quantified into 6 levels between the IoT device and the EN. As for the DRL agent, we use vanilla version of DDQN with parameter settings: exploration probability 0.001, replay memory capacity 5000, learning rate 0.005, discount factor 0.9, two full connected layers with 200 neurons activated by tanh function each layer, replacing the target Q network every 250 times training and mini batch size 200. Centralized DRL training is also realized and tested, as a baseline method for evaluating the effectiveness of our work, i.e.. All sensing data collected by IoT devices are uploaded to an EN for subsequent DRL training.

Three IoT devices are randomly solicited for investigation, the x-axis is training step, their training loss are given in Fig.3(a)-3(d). The performance of FL-enabled DRL training and centralized DRL training presented in Fig.3(e)-3(g) and Fig.3(h), respectively, the x-axis is training period. In Fig.3, the change in utility during the training process is represented by solid lines. Details of performance evaluation can be given as follows despite visible characteristics:

(1) At first it can be seen, the fluctuation range of FL-based DRL training with respect to utility variation is bigger than the centralized training. With the training losses decreasing, the achieved utilities of three randomly selected IoT devices hold the same level as the one realized by centralized DRL training. This result corroborates the performance of FL-based DRL training for computation offloading is approach to the results of centralized DRL training.

(2) But there are no limitations for the wireless channel in the centralized DRL training, it assumes that the training data can be successfully uploaded to the EN without any delay. However, this is impractical instead, and it in turn manifests the effectiveness of our work. Particularly, the performance of our work becomes comparable to the centralized DRL once the model aggregation of FL has been performed several times. Therefore, when networking is still the restriction, FL-based DRL training is more practical at least.

B. TRAINING PERFORMANCE UNDER DIFFERENT TASKS PROBABILITIES

We noticed that different IoT devices have different functions, some require frequent data collection and uploading frequently, while others do not. Therefore, different IoT devices may have a different probability of generation tasks in a period of time. This paper considers the probability of task generation in IoT devices. In the experiment, the task generation probability is divided into nine levels (0.1-0.9), where different numbers represent the rate at which tasks may be generated in a time period. We compared several factors,

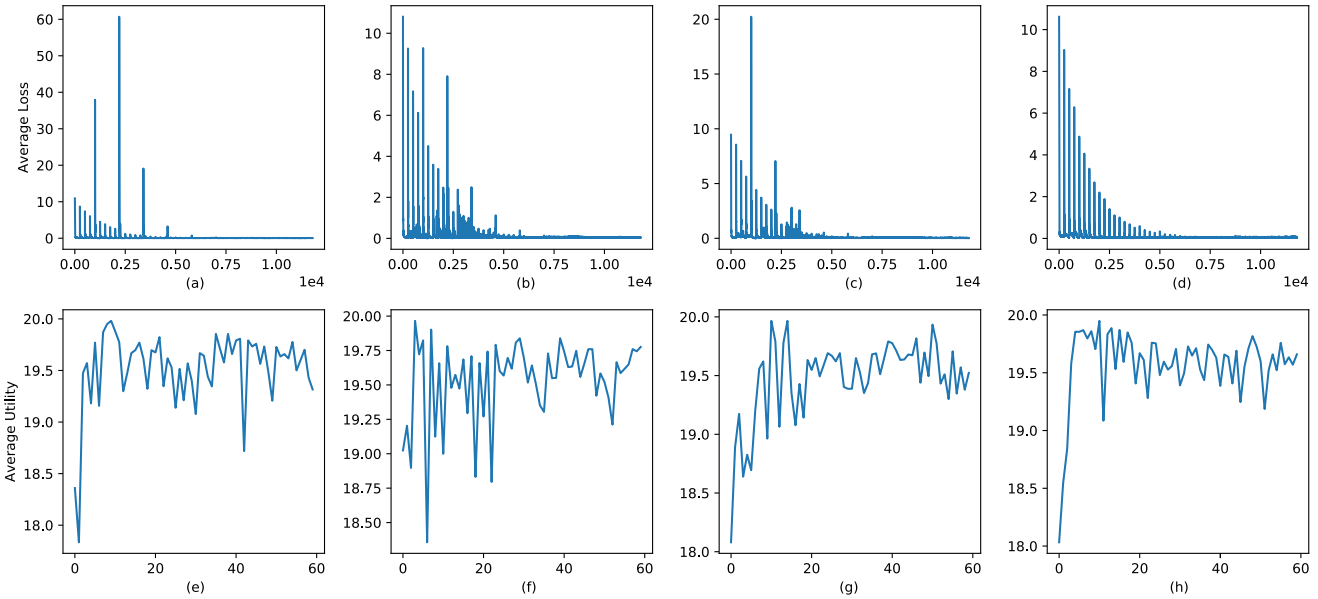


FIGURE 3. Training performance of IoT devices with Federated Learning-based DRL and centralized. (a) Loss of device A. (b) Loss of device B. (c) Loss of device C. (d) Loss of centralized. (e) Utility of device A. (f) Utility of device B. (g) Utility of device C. (h) Utility of centralized.

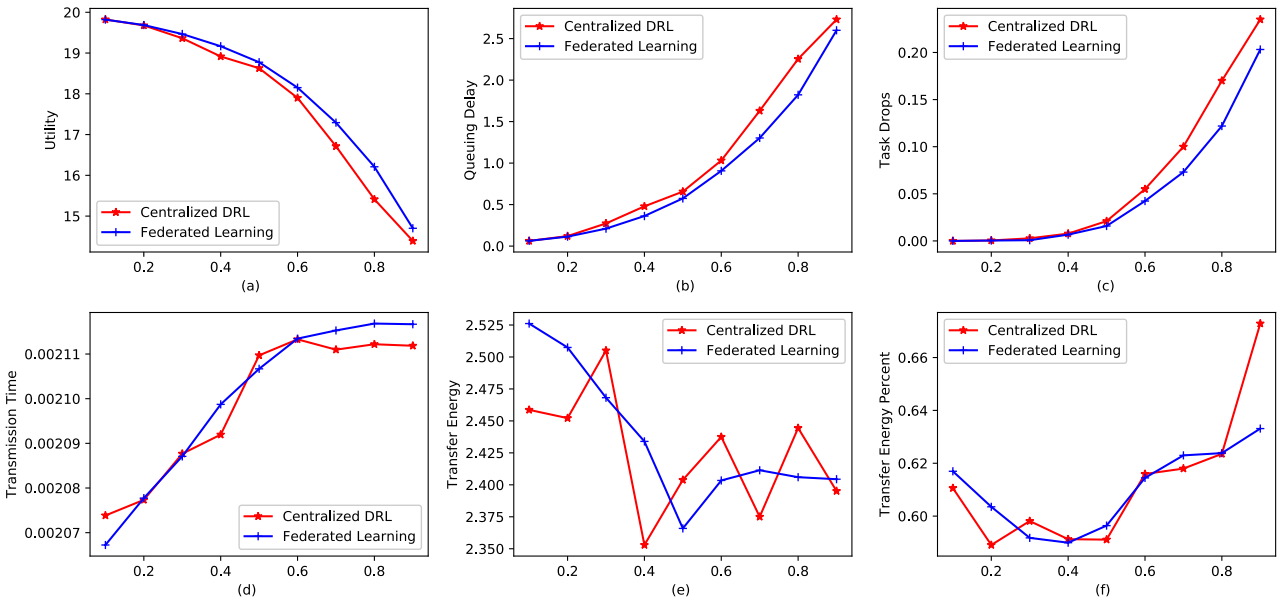


FIGURE 4. Training performance under different task probabilities with Federated Learning-based DRL and centralized. (a) Average utility per task probability. (b) Average queuing delay per task probability. (c) Average task drops per task probability. (d) Average transfer time per task probability. (e) Average transfer energy per task probability. (f) Average transfer energy percent per task probability.

such as utility, tasks queuing delay, tasks drop, transmission time, transmission energy and percentage of transmitted energy.

In Fig.4(a), when the probability of task generation increases, the average utility of FL and DRL decreases significantly, but the average utility of FL is greater than that of DRL. This proves that FL and DRL can reach similar levels in different probabilities of task generation. The detailed training changes in utility and loss in the case where the probability of task generation is fixed (0.3) are shown in Fig.3. The utility calculation results are related to the number of task drops, the task queue delay, the penalty of the task failure, the offload payment and the calculation duration (the maximum is the duration of one cycle). Utility is the weighted summation of these factors and the weights can be adjusted for the purpose of the experiment.

When the task generation probability is very small (0.1-0.2), the difference of queuing delay between the two learning methods is very small, as shown in Fig.4(b). At this time, the number of tasks in the queue is small, and the delay processing speed of the task queue is very quickly. However, as the number of tasks increases, more and more offloading policies DRL-based always have higher delay than those FL-based. The impact factor for dropping a task is the length of the task queue. In our experiment, we treated the task queue length as a queue delay. Comparative analysis shows

that the increase of task queue delay leads to the increase of dropped tasks.

The task is dropped because the task queue is full and the device cannot add a new task to it. The number of waiting tasks in the task queue is closely related to the probability of task generation and energy in the energy queue. When the task generation probability is small (0.1-0.3), the dropped tasks in both modes are at a low level Fig.4(c). With the increase of the probability of task generation, the dropped task of the center-based DRL is higher than that of the FL-based dropped tasks. When the probability increases, the FL-based training can decide whether to perform local calculation or offload the task to an edge node according to various environmental conditions. If conditions permit, newly arrived tasks can be executed locally, which reduces the number of dropped tasks. This will maintain the integrity of the data to some extent.

According to the task generation probability and the transmission time of edge task Fig.4(e), when the task generation probability is small (0.1-0.3), the transmission time of DRL and FL is very similar, and the transmission time mainly depends on the size of task data and the channel strength during transmission. When the task generation probability increases (0.4-0.5), the task transmission time of FL-based learning is close to that of center-based DRL training. When the task generation probability increases to (0.7-0.9), the task transmission of FL-based is even higher than DRL. Since the task transmission time is affected by channel gain (usually a fixed value within a certain period and a limited space), transmission power, noise, task queue and energy queue, the FL-based agent will make decisions based on the current state information. As the number of tasks increases, FL uploads new tasks on the EN, while DRL drops more tasks. Therefore, the difference in the number of transmission tasks results in the difference in transmission time.

It is assumed that the sum of the energy transmitted and the energy used for local processing in IoT devices is constant. The ratio of the sum of the transmitted energy and the applied energy (local processing energy, transmitted energy) is the proportion of the transmitted energy in IoT devices. With the increase of workload, the transmission energy usage based on FL and central DRL reached a similar level Fig.4(f), and these two levels even reached the same level at 0.4, 0.6 and 0.8. However, when the task generation probability is 0.9, the transfer energy proportion based on the central DRL is higher than that of FL-based. Through the above analysis, the number of dropped tasks, the queuing delay and the transmission energy of DRL-based are higher than that of FL-based, which proves that the training based on federal learning is better than that of deep reinforcement learning.

With the increase of the number of tasks, the task exceeds the processing capacity of the IoT device and the number of tasks waiting for the IoT device will increase until the waiting task fills the device task queue. Therefore, when the queue is full, the new task will be dropped. When the probability is fixed, the number of tasks generated within a time period can be regarded as a fixed value. There are more drop tasks than FL in the centralized DRL, which leads to fewer tasks in DRL than FL. From the perspective of product services, complete data can bring better service experience. Perform the appropriate offload, the lower drop tasks and the short transmission time prove the advantages of FL.

Of course, there are two problems with FL because of its advantages. On the one hand, it is not feasible to perform fast DRL training because it requires at least several effective model aggregations on the EN. On the other hand, it will lose the accuracy of the model compared to DRL training, although it is relatively negligible. Therefore, in future work, we will study how to rationally arrange model updates and model aggregation time in order to better weigh these advantages and disadvantages.

VI. CONCLUSIONS

This paper studies the combination of DRL and FL in the IoT environment that supports edge computing. The effectiveness of FL-based learning is proved by the experiment of the use case, viz., computation task offloading. In the future, we will delve into whether DRL has model compression techniques and how to arrange FL-based learning training at a finer level.

REFERENCES

[1] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 13, nos. 2–3, pp. 203–221, 1996. doi: 10.1007/BF01130406.

[2] X. Wang, Y. Han, and C. Wang, "In-Edge AI: Intelligentizing mobile edge computing, caching and communication by federated learning," 2018, *arXiv:1809.07857*. [Online]. Available: https://arxiv.org/abs/1809.07857

[3] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018. doi: 10.1109/JSAC.2018.2815360.

[4] 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.

[5] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016. doi: 10.1109/TNET.2015.2487344.

[6] X. Wang, M. Chen, V. C. M. Leung, Z. Han, and K. Hwang, "Integrating social networks with mobile device-to-device services," *IEEE Trans. Services Comput.*, to be published.

[7] 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.*, to be published. doi: 10.1109/JIOT.2018.2876279.

[8] X. Wang, Y. Zhang, V. C. M. Leung, N. Guizani, and T. Jiang, "D2D big data: Content deliveries over wireless device-to-device sharing in large-scale mobile networks," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 32–38, Feb. 2018.

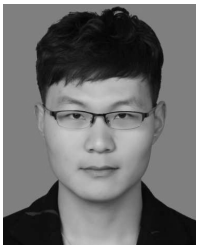
[9] H. B. McMahan and E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Fort Lauderdale, FL, USA, vol. 54, 2017, pp. 1–10. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[10] T. Qiu, X. Wang, C. Chen, M. Atiquzzaman, and L. Liu, "TMED: A spider-Web-like transmission mechanism for emergency data in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8682–8694, Sep. 2018.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. doi: 10.1038/nature14236.

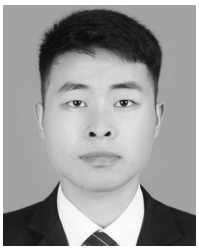
[12] T. Qiu, H. Wang, K. Li, H. Ning, A. K. Sangaiah, and B. Chen, "SIGMM: A novel machine learning algorithm for spammer identification in industrial mobile cloud computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2349–2359, Apr. 2019.
[13] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016. doi: 10.1109/JIOT.2016.2579198.
[14] T. Qiu, K. Zheng, M. Han, C. L. P. Chen, and M. Xu, "A data-emergency-aware scheduling scheme for Internet of Things in smart cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2042–2051, May 2018.
[15] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, Phoenix, AZ, USA, 2016, pp. 2094–2100. [Online]. Available: http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12389
[16] 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. 6926–6939, Oct. 2017.
[17] T. Qiu, R. Qiao, and D. Wu, "EABS: An event-aware backpressure scheduling scheme for emergency Internet of Things," *IEEE Trans. Mobile Comput.*, vol. 17, no. 1, pp. 72–84, Jan. 2018.

JIANJI REN received the B.S. degree from the Department of Mathematics, Jinan University, in 2005, and the M.S. and Ph.D. degrees from the School of Computer Science and Engineering, Dong-A University, in 2007 and 2010, respectively. He is currently an Associate Professor with the College of Computer Science and Technology, Henan Polytechnic University. His current research interests include mobile content-centric networks and collaborative caching in edge computing.

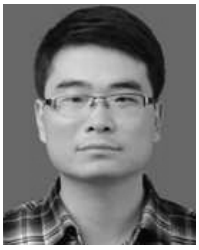


HAICHAO WANG received the B.S. degree in natural geography and resource environment from Henan Polytechnic University, Jiaozuo, Henan, China, in 2018, where he is currently pursuing the master's degree in software engineering from the College of Computer Science and Technology (Software College). His research interests include edge computing, edge caching, big data analysis, deep learning, and the Internet of Things technology.

TINGTING HOU is currently pursuing the B.S. degree from the College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, Henan, China. Her current research interests include edge computing, edge caching, deep learning, big data analysis, and the Internet of Things technology.



SHUAI ZHENG is currently pursuing the B.S. degree with the College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, Henan, China. His current research interests include edge computing, edge caching, deep learning, big data analysis, and data mining.



CHAOSHENG TANG received the Ph.D. degree in management science and engineering from Yanshan University, Qinhuangdao, Hebei, China, in 2015. His current research interests include machine learning, complexity theory, multimedia applications, and online social networks.