Federated Learning-based Computation Offloading Optimization in Edge Computing-supported Internet of Things

Yiwen Han[†], Ding Li[†], Haotian Qi[†], Jianji Ren[§], Xiaofei Wang[†,⋆]
[⋆]Corresponding author: xiaofeiwang@tju.edu.cn
[†]College of Intelligence and Computing, Tianjin University, Tianjin, China
[§]Institute of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, Henan, China

ABSTRACT

Recent visualizations of smart cities, factories, healthcare system and etc. raise challenges on the capability and connectivity of massive Internet of Things (IoT) devices. Hence, edge computing is emerged to complement these capability-constrained devices with an idea offloading intensive computation tasks from them to edge nodes. By taking advantage of this feature, IoT devices are able to conserve more energy and still maintain the quality of services they shall provide. Nevertheless, computation offloading decisions concern joint and complex resource management and should be determined in real time facing dynamic workloads and radio environment. Therefore, in this work, we use multiple Deep Reinforcement Learning (DRL) agents deployed on IoT devices to instruct the decision making of themselves. On the other hand, Federated Learning is utilized to train DRL agents in a distributed fashion, aiming to make the DRL-based decision making practical and further decrease the transmission cost between IoT devices and Edge Nodes. Experimental results corroborate the effectiveness of both the DRL and Federated Learning in the dynamic IoT system.

CCS CONCEPTS

• **Networks** → **Mobile networks**; • **Human-centered computing** → **Mobile computing**; • **Computing methodologies** → *Cooperation and coordination.*

KEYWORDS

Federated learning, computation offloading, IoT, edge computing

ACM Reference Format:
Yiwen Han[†], Ding Li[†], Haotian Qi[†], Jianji Ren[§], Xiaofei Wang[†,⋆]. 2019. Federated Learning-based Computation Offloading Optimization in Edge Computing-supported Internet of Things. In *ACM Turing Celebration Conference - China (ACM TURC 2019) (ACM TURC 2019), May 17–19, 2019, Chengdu, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3321408.3321586

ACM TURC 2019, May 17–19, 2019, Chengdu, China

ACM ISBN 978-1-4503-7158-2/19/05…$15.00
https://doi.org/10.1145/3321408.3321586

1 INTRODUCTION

Catering for the increasing service requirement in future smart cities, factories, healthcare system and etc., massive IoT devices will be deployed in a pervasive fashion to carry out tasks like monitoring, sensing data collection and preprocessing and immediate decision-making. Generally, these IoT devices are relatively weak and heterogeneous, and naturally, it is infeasible to directly support intensive computation cost raised by the above tasks. Nevertheless, edge computing, as an emerging technique, is envisioned as a promising architecture to admit the offloading tasks from IoT devices [7]. On the other hand, the edge node in edge computing systems is adopted as a coordinator among them and responsible for their communications and even load-balance.

In [4], without using convex optimization [2] and game theory [3] to solve resource allocation problems in offloading, authors adopted Deep Reinforcement Learning (DRL) to deal with the comprehensive resource allocation in computation offloading for maximizing the long-term benefits of the energy consumption and execution delay, which requires no prior knowledge of network statics and only need partial instead of global information. Particularly, this kind of optimization can tackle the following issues: 1) *Uncertain Inputs*: some of the key information, which are necessary for model-dependent optimization, are difficult to acquire due to privacy issues and variational radio channels; 2) *Dynamic Conditions*: real-time workload dynamics of the whole edge system should be taking into consideration; 3) *Temporal Isolation*: not only optimizing a snapshot of the system but also the long-term utility of the system should be thought over. However, an unnoticed assumption has been made in [4]. Specifically, the IoT device is considered as a very powerful machine and has the computation ability to train its own DRL agent independently. However, in the near future, IoT devices may not be that powerful and they can at most support lightweight neutral network computation.

Thus, we propose a distributed training scheme based on Federated Learning (FL) [5] to alleviate the training burden on each device. Unlike the traditional distributed training in the data center with an excellent networking environment, this training is restrained by the wireless communication and networking and shall be performed in an efficient manner of communication. In this vein, observation data sensed by each IoT device are not required to be transmitted frequently between it and the edge node. Observation data on a specific IoT device are used for local training, and only updated parameters of the DRL agent are uploaded to the edge node for further model aggregation.

Therefore, in this work, we use FL to conduct the training process of DRL agents for jointly allocating communication and computation resources. Experimental results corroborate its effectiveness comparing to the centralized training method which is impractical though.

2 SYSTEM MODEL

2.1 System Architecture

In this work, a system model in IoT environment with energy harvesting, as shown in Fig. 1, is taken for analysis. In this scenario, IoT devices, denoted as $\mathcal{D} = \{1, \cdots, D\}$, are in the service range of a set $\mathcal{N} = \{1, \cdots, N\}$ of Edge Nodes (ENs) providing both communication and computation offloading. Each device can choose one EN out of $\mathcal{N}$ to establish communication and offload intensive computation tasks with allocated frequency bandwidth W Hz. For quantitative analysis, the time horizon is discretized into time epochs indexed by i with equivalent duration as δ (in seconds).

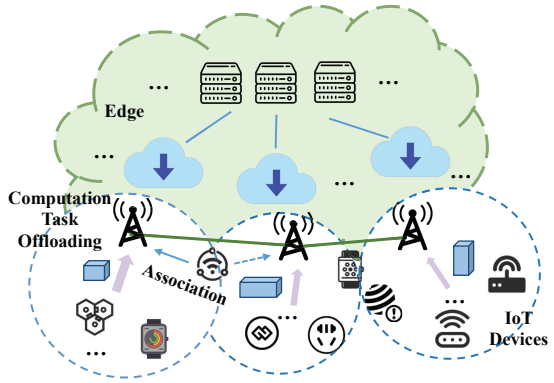


Figure 1: Edge computing-supported IoT system

On the side of IoT devices, one IoT device is taken as a typical one for illustrating the model. It can harvest energy units from edge nodes and store them in an energy queue with a maximum length $q^{\mathrm{e}}_{\max}$ for wireless transmission and computation. Meanwhile, it always admits computation tasks for performing specific services, and these generated tasks constitute an Independent and Identically Distributed (I.I.D.) sequence of Bernoulli random variables with a common parameter $\lambda^{\mathrm{t}} \in [0, 1]$. In addition, there is a local task queue with a maximum length $q^{\mathrm{t}}_{\max}$ 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). If a task is generated during an epoch i, the task arrival indicator for the device at epoch i is represented as $a^{\mathrm{t}}_i = 1$ and otherwise $a^{\mathrm{t}}_i = 0$. The computation task is modeled as (μ, ν), of which μ (in bits) and ν represent the transmission data size required for offloading a task and the needed number of CPU cycles for processing the task, respectively.

With respect to the computation task, a computation task taken from task queue can be determined for execution locally on the IoT device or offloading to an EN for processing. Back to the IoT device, it should make a joint action (c_i, e_i) at the beginning of each epoch i to make a decision on 1) whether the task should be processed locally ($c_i = 0$) or offloaded to an EN ($c_i \in \mathcal{N}$), noted that $c_i \in \{0\} \cup \mathcal{N}$; 2) how many energy units ($e_i \in \mathbb{N}_+$) stored in the energy queue should be allocated. In this circumstance, when a computation task is allocated to be processed locally with permitted energy units e_i (if there is any), viz., $c_i = 0$, the allocated CPU frequency f_i for this task can be modeled with a maximum limitation as

$$f_i = \sqrt{e_i/(\tau \cdot \nu)} \le f^{\mathrm{c}}_{\max}\,, \tag{1}$$

where τ 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

$$d^{\mathrm{m}}_i = \nu/f_i\,. \tag{2}$$

If the IoT device decides to associate with an EN, the radio channel quality between them should be considered, in that it directly affects the wireless communication particularly the transmission rate. We denote g^n_i as the channel gain during the epoch i between the IoT device and an EN $n \in \mathcal{N}$, which is assumed static and independently taken from a finite state space $\mathcal{G}_n$. When a radio link is established for them, the achievable data rate can be calculated as

$$r_i = W \cdot \log_2\left(1 + \cdot g^n_i \cdot p^{\mathrm{tr}}_i / I\right)\,, \tag{3}$$

where I is the power of interference plus noise and p^{tr}_i is the transmit power with maximum limitation $p^{\mathrm{tr}}_{\max}$, which satisfies

$$p^{\mathrm{tr}}_i = e_i/d^{\mathrm{tr}}_i \le p^{\mathrm{tr}}_{\max}. \tag{4}$$

Thus, the total time for transmitting the input data μ is

$$d^{\mathrm{tr}}_i = \mu/r_i\,. \tag{5}$$

According to the proof in [4], given the association $c_i \in \mathcal{N}$ and the allocated energy units $e_i > 0$ at a epoch i, the transmitting rate should remain a constant for achieving the minimum transmission time, which is preferred in practical. Therefore, the minimum transmission time can be derived by solving simultaneous equations of (3), (4) and (5) as

$$\log_2(1 + \frac{g^{c_i}_i \cdot e_i}{I \cdot d^{\mathrm{tr}}_i}) = \frac{\mu}{W \cdot d^{\mathrm{tr}}_i}\,. \tag{6}$$

Noticing that the processing delay on the EN is assumed to be much less compared to the transmission time when the IoT decides to offload.

2.2 Dynamic System Model

In the real-time variational scenario, the energy queue and the task queue should be particularly focused, since they represent the computation resource and the workload, respectively. We use q^{e}_i to represent the energy queue length insider the IoT device at the beginning of an epoch i, its dynamics can be described as

$$q^{\mathrm{e}}_{i+1} = \min\left\{q^{\mathrm{e}}_i - e_i + a^{\mathrm{e}}_i,\ q^{\mathrm{e}}_{\max}\right\}\,, \tag{7}$$

where $a^{\mathrm{e}}_i \in \mathbb{N}_+$ is the total number of energy units received till the end of epoch i. With the available energy units provided by the energy queue, the achievable task execution delay, which includes both communication and computation, is the main focus. Besides

the processing delay of a task and the transmission delay, the handover delay is also considered, and the task execution delay can be expressed as

$$d_i = \begin{cases} d_i^{\mathrm{m}}, & \text{if } e_i > 0 \text{ and } c_i = 0 \\ h_i + d_i^{\mathrm{tr}} + d^{\mathrm{s}}, & \text{if } e_i > 0 \text{ and } c_i \in \mathcal{N} \\ 0, & \text{if } e_i = 0 \end{cases}, \tag{8}$$

where d^s is the delay of server-side execution, which is relatively a small constant as aforementioned, and h_i is the handover delay resulting from altering EN association.

In more detail, failures of task processing and offloading are also taken into consideration. Specifically, the execution of a task will be deemed as a failure and thus wait for later processing in the task queue till successfully executed in two circumstances, viz., 1) the IoT device can not process a computation task even until the end of an epoch; 2) the IoT device chooses to offload a task to a specific EN, and it fails due to the longtime transmission introduced by either not enough allocated energy units or bad radio channel quality. For convenient expression, dynamics of the task queue length can be calculated as

$$q_{i+1}^{\mathrm{t}} = \min\left\{q_i^{\mathrm{t}} - \mathbf{1}_{\{0<d_i\leq\delta\}} + a_i^{\mathrm{t}},\ q_{\max}^{\mathrm{t}}\right\}. \tag{9}$$

Certainly, if the task queue is full of awaiting tasks, freshly generated tasks must be dropped, which shall be avoided in the ideal case. Then the number of computation task drop in an epoch i can be described as

$$\eta_i = \max\left\{q_i^{\mathrm{t}} - \mathbf{1}_{\{0<d_i\leq\delta\}} + a_i^{\mathrm{t}} - q_{\max}^{\mathrm{t}},\ 0\right\}. \tag{10}$$

Moreover, the unwished queuing delay for computation tasks will be incurred, since not every task can be successfully handled in one epoch δ. The queuing delay during the epoch i is treated as the length (q_i^{t}) of the task queue inside the IoT device, which is

$$\rho_i = q_i^{\mathrm{t}} - 1_{\{d_i>0\}}. \tag{11}$$

In the meantime, if the execution of a computation task fails, the corresponding penalty will be given as

$$\varphi_i = \mathbf{1}_{\{d_i>\delta\}}. \tag{12}$$

In addition, the IoT device shall pay the fee for occupying the EN when it decides to offload its computation tasks to the EN. Such payment is weighted by the time consumed for the EN receiving and processing the task input data. With defining $\pi \in \mathbb{R}_+$ as the price per unit of time, the payment expression can be written as

$$\phi_i = \pi \cdot (\min\{d_i, \delta\} - h_i) \cdot \mathbf{1}_{\{c_i\in\mathcal{N}\}}. \tag{13}$$

3 POLICY TRAINING COORDINATED BY FEDERATED LEARNING

3.1 Problem Formulation

Based on the system model descripted in Sec. 2.1 and Sec. 2.2, we expound the optimization problem in this section. Collecting all essential elements, the network state of the IoT device can be organized as

$$\begin{aligned}\mathcal{X}_i &= (q_i^{\mathrm{t}}, q_i^{\mathrm{e}}, s_i, \boldsymbol{g}_i) \in \boldsymbol{\mathcal{X}} \\ &\stackrel{\mathrm{def}}{=} \left\{0, 1, \cdots, q_{\max}^{\mathrm{t}}\right\} \times \left\{0, 1, \cdots, q_{\max}^{\mathrm{e}}\right\} \times \mathcal{N} \times \{\times_{n\in\mathcal{N}}\mathcal{G}_n\}\end{aligned}, \tag{14}$$

where $\boldsymbol{g}_i = \left(g_i^n : n \in \mathcal{N}\right)$. At the beginning of epoch i, the IoT device makes a decision on where to process the computation task and how many energy resources should be allocated, i.e.,

$$(c_i, e_i) \in \mathcal{J} \stackrel{\mathrm{def}}{=} \{\{0\} \cup \mathcal{N}\} \times \left\{0, 1, \cdots, q_{\max}^{\mathrm{e}}\right\}. \tag{15}$$

A sequence of the above actions should be determined by an optimal control policy $\boldsymbol{\Phi}$, maximizing the expected long-term utility as

$$U(\mathcal{X}, \boldsymbol{\Phi}) = \mathbb{E}_{\boldsymbol{\Phi}}\left[\lim_{I\to\infty} \frac{1}{I} \cdot \sum_{i=1}^{I} u\left(\mathcal{X}_i, \Phi\left(\mathcal{X}_i\right)\right) | \mathcal{X}_1 = \mathcal{X}\right], \tag{16}$$

where $\mathcal{X}_1$ is the initial network state, and $u(\cdot)$ is the immediate utility at epoch i defined as the customized operation combination of task execution delay d_i, the number of task drop η_i, the task queuing delay ρ_i, the penalty of execution failure φ_i and the payment ϕ_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 φ_i can be amplified to enhance its ratio in the whole utility.

3.2 Merits of Federated Learning in Edge Computing

In Sec. 2, single IoT device is taken as an example for interpretation. It is common that DRL techniques can deal with this kind of problem well, thus we use Double Deep Q Learning (DDQN) [6, 8] for each IoT device to maximize the long-term utility of its control policy. In this section, we will show the merits of using FL to coordinate the training process among multiple IoT devices.

DRL techniques are efficient in finding the optimal policy in the dynamic edge system, but they also demand abundant computation resources. Therefore, the deployment of DRL agent should be carefully thought over. On one hand, if the DRL agent is trained on the EN, it will bring about three disadvantages: 1) massive training data will be always transmitted from IoT devices to the EN, and consequently burden the wireless channel; 2) it may jeopardize the privacy, since the uploaded training data might be privacy-sensitive, especially in the scenario of industrial informatics; 3) in spite of the training data can be transformed for privacy protection, the proxy data received by the EN is less relevant and lose the pertinence for a specific IoT device.

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

Hence, taking efficiently training DRL agent in a distributed manner into consideration is a natural choice, as depicted in Fig. 2. Howbeit training the DRL agent in every IoT device or the EN can realize the best performance, it is practical to adopt distributed DRL training. In addition, due to the networking constraints coupled with the challenge of handling non-I.I.D. data and privacy issues in the edge computing system, most of efficient distributed deep learning techniques [5] are not feasible. For these reasons, FL is introduced in this work for distributively training DRL agents.

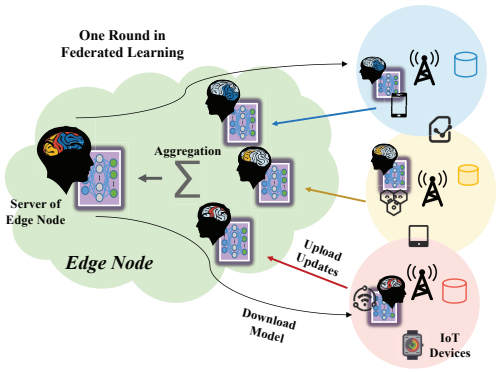


Figure 2: Federated Learning-based DRL Training

3.3 Federated Learning-based DRL Training about Computation Offloading

In computation offloading, each IoT should take a control action according to the dynamic of networking state and its own workload. Hence, we propose using the EN to coordinate IoT devices. By taking advantage of this scheme, IoT devices can be able to maintain a complex DRL agent with relatively less computation burden.

Illustrated by Algorithm 1, FL iteratively selects a random set of IoT devices to *1)* download parameters of the DRL agent from the EN; *2)* perform the training process on the upgraded (downloaded) model with their own data; *3)* upload only updated model parameters of the DRL agent to the EN for model aggregation. By this means, 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 in the system. EN-side proxy data is less relevant to the local data in the IoT device. In the envisioned IoT system, massive IoT devices can acquire various and localized sensing data for updating the DRL agent. These data may include the channel gain of the radio channel, the remained energy resources, the workload of both IoT devices and ENs and etc. Hence, FL can leverage these localized data to make the whole IoT system more cognitive.

4 PERFORMANCE EVALUATION

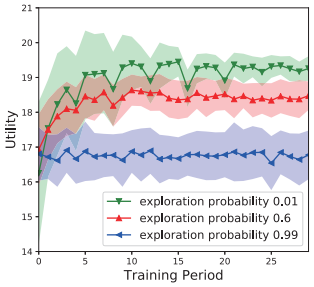


Figure 3: Exploration effect

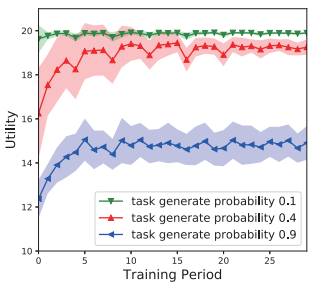


Figure 4: Performance against workload

To evaluate the capabilities of our proposed method, we investigate an edge system with $N = 6$ ENs and $D = 15$ IoT devices. The

Algorithm 1 Federated Learning-based Policy Training

Initialization:

With respect to the global DRL agent in the EN:

Initialize the DRL agent with random weights $\boldsymbol{\theta}_0$

Initialize the gross training times A_0 of all devices

With respect to each IoT device $d \in \mathcal{D}$:

Initialize the experience replay memory $\mathcal{M}_0^d$

Initialize the local DRL model $\boldsymbol{\theta}_0^d$

Download $\boldsymbol{\theta}_0$ from the EN and let $\boldsymbol{\theta}_0^d = \boldsymbol{\theta}_0$

Iteration:

for each round $t = 1$ **to** T **do**

$\mathcal{S}_t \leftarrow$ {random set of m available IoT devices}

for each device $d \in \mathcal{S}_t$ in parallel **do**

Fetch $\boldsymbol{\theta}_t$ from the EN as and let $\boldsymbol{\theta}_t^d = \boldsymbol{\theta}_t$

Sense and update $\mathcal{M}_t^d$

Train the DRL agent locally with $\boldsymbol{\theta}_t^d$ on $\mathcal{M}^d$

Upload the trained $\boldsymbol{\theta}_{t+1}^d$ to the EN

Notify the EN the times A_t^d of local training

end for

With respect to the EN:

Receive all model updates

Refresh the statistical $A_t = \sum_{d \in \mathcal{S}} A_t^d$

Perform model aggregation as:

$$\boldsymbol{\theta}_{t+1} \leftarrow \sum_{d \in \mathcal{S}} \left(A_t^d / A_t\right) \cdot \boldsymbol{\theta}_{t+1}^d$$

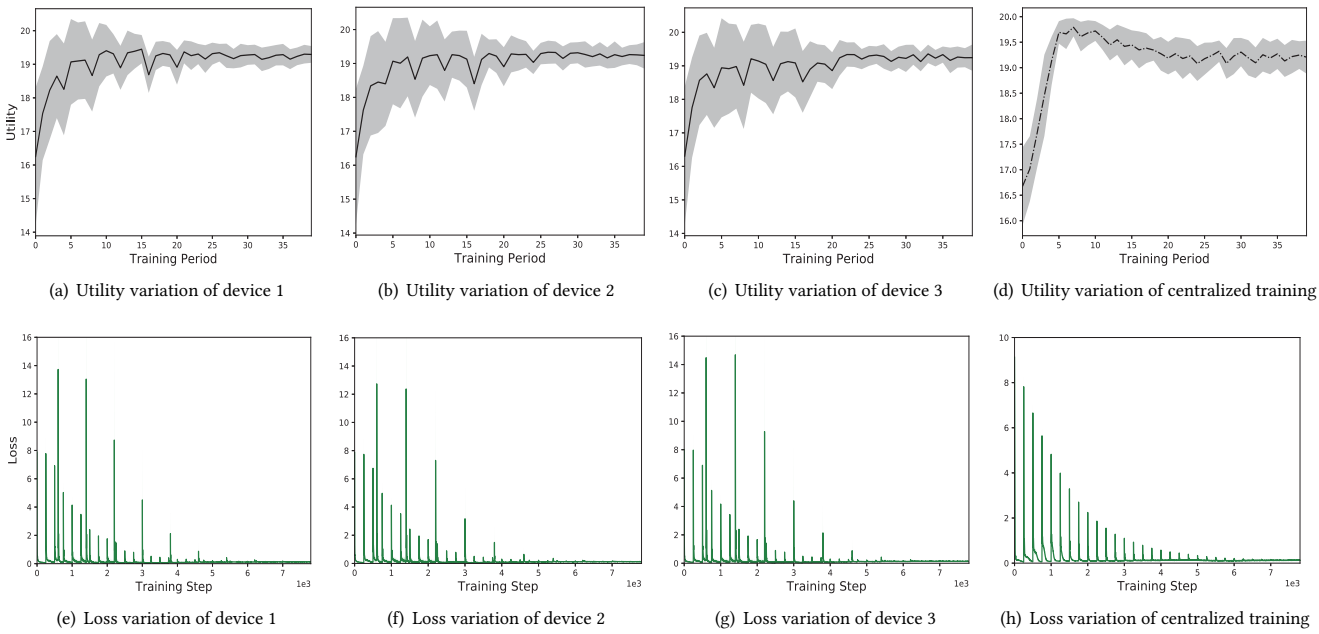
end for

channel gain between the IoT device and the EN are quantified into 6 levels, according to the quality of the wireless channel. About the DRL agent, we use vanilla version of DDQN with parameter settings: two full connected layers with 200 neurons activated by tanh function each layer, learning rate 0.005, discount factor 0.9, replay memory capacity 5000, mini-batch size 200 and replacing the target Q network every 250 times training. First, we evaluate the effect of exploration probability in our method, as presented in Fig. 3, and based on the results, accordingly we choose the final exploration possibility as 0.01 in the following experiments, the results of which are depicted in Fig. 5.

As a baseline method for evaluating the effectiveness of our work, centralized DRL training is also realized and tested, i.e., all sensing data collected by IoT devices are uploaded to an EN for subsequent DRL training. One hundred times of experiments are taken for collecting statics. In Fig. 3-5, solid lines and the shallow area around them represent mean values and the standard deviation, respectively.

Randomly soliciting three IoT devices for investigation, Fig. 5(a)-5(c) and Fig. 5(d) present the performance of FL-enabled DRL training and centralized DRL training, respectively. Accordingly, their training loss statics are correspondingly given in Fig. 5(e)-5(h).

Despite visible characteristics, details of performance evaluation can be given as follows. ***1)*** It can be seen, at first, the standard deviation of centralized training with respect to utility variation is smaller than the FL-based DRL training. With the decrease of training losses, 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



(a) Utility variation of device 1 (b) Utility variation of device 2 (c) Utility variation of device 3 (d) Utility variation of centralized training

(e) Loss variation of device 1 (f) Loss variation of device 2 (g) Loss variation of device 3 (h) Loss variation of centralized training

Figure 5: Computation offloading performance with Federated Learning-based and centralized DRL training training

DRL training for computation offloading is approach to the results of centralized DRL training. ***2)*** Noticing that the centralized DRL training assumes that there are no limitations for the wireless channel and tremendous 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, once the model aggregation of FL has been performed several times, the performance of our work becomes comparable to the centralized DRL. Therefore, FL-based DRL training is more practical at least when the networking is still the restriction. ***3)*** Fig. 4 shows the performance of one random IoT device with varying the workload on itself, which corroborates that our work can adapt to the workload variation and be converged in dynamic situation.

Certainly, FL trades two things for its advantages. On one hand, it is not feasible for required fast DRL training, since it needs at least several valid model aggregation on the EN. On the other hand, it will lose model accuracy compared to the centralized DRL training, though relatively negligible. Therefore, how to schedule the time for downloading model, uploading model updates and performing model aggregation should be investigated in the future work for better weighting these trade-offs.

5 CONCLUSIONS

Combination of DRL training and FL in this paper is studied in the edge computing-supported IoT system. After performing experiments on a use case, viz., computation task offloading, the effectiveness of FL-based DRL is affirmed. In the future, we will take a deep study in if there are model compression techniques with respect to DRL, and how to schedule the FL-based DRL training in a fine-grained fashion.

ACKNOWLEDGMENTS

The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions. This work is supported by the National Key R&D Program (2018YFC0809803) of China and China NSFC (Youth) through grant 61702364.

REFERENCES

[1] Thomas D. Burd and Robert W. Brodersen. 1996. Processor design for portable systems. *VLSI Signal Processing* 13, 2-3 (Aug. 1996), 203–221.

[2] Min Chen and Yixue Hao. 2018. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE Journal on Selected Areas in Communications* 36, 3 (Mar. 2018), 587–597.

[3] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing. *IEEE/ACM Trans. Netw.* 24, 5 (Oct. 2016), 2795–2808.

[4] Xianfu Chen, Honggang Zhang, Celimuge Wu, Shiwen Mao, Yusheng Ji, and Mehdi Bennis. 2018. Optimized Computation Offloading Performance in Virtual Edge Computing Systems via Deep Reinforcement Learning. *IEEE Internet of Things Journal* (2018).

[5] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, FL, USA (Proceedings of Machine Learning Research)*, Vol. 54. PMLR, 1273–1282.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533.

[7] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (Oct. 2016), 637–646.

[8] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence,2016, Phoenix, Arizona, USA*. AAAI Press, 2094–2100.