A Learning-based Incentive Mechanism for Federated Learning

Yufeng Zhan, Peng Li, *Member, IEEE*, Zhihao Qu, *Member, IEEE*, Deze Zeng, *Member, IEEE*, and Song Guo, *Fellow, IEEE*

***Abstract*—Internet of Things (IoT) generates large amounts of data at the network edge. Machine learning models are often built on these data, to enable the detection, classification, and prediction of future events. Due to network bandwidth, storage, and especially privacy concerns, it is often impossible to send all the IoT data to the data center for centralized model training. To address these issues, federated learning has been proposed to let nodes use local data to train models, which are then aggregated to synthesize a global model. Most of the existing work has focused on designing learning algorithms with provable convergence time, but other issues such as incentive mechanism are unexplored. Although incentive mechanisms have been extensively studied in network and computation resource allocation, yet they can not be applied to federated learning directly due to the unique challenges of information un-sharing and difficulties of contribution evaluation. In this paper, we study the incentive mechanism for federated learning to motivate edge nodes to contribute model training. Specifically, a deep reinforcement learning-based (DRL) incentive mechanism has been designed to determine the optimal pricing strategy for the parameter server and the optimal training strategies for edge nodes. Finally, numerical experiments have been implemented to evaluate the efficiency of the proposed DRL-based incentive mechanism.**

***Index Terms*—Federated learning, incentive mechanism, deep reinforcement learning (DRL)**

I. Introduction

Deep learning has demonstrated great potentials to revolutionize the Internet-of-Things (IoT) by improving the efficiency of deployment and management of IoT, enhancing IoT security and privacy protection, and enabling various smart applications [1]–[3]. The success of deep learning for IoT stems from the availability of big training data and massive computation power. However, in many applications, training data are generated by distributed devices or equipments owned by individuals or different organizations, who hesitate to share their data that expose privacy. Moreover, it becomes difficult to aggregate these data to a single computing site for centralized training due to the increasing data size.

Federated learning has been proposed to enable distributed computing nodes to collaboratively train models without exposing their own data. Its basic idea is to let these computing nodes train local models using their own data, respectively, and then upload the local models, instead of data, to a logically centralized parameter server that synthesizes a global model. Since an inception by Google [4], federated learning has attracted great attentions from both academia and industry.

Although federated learning has shown great advantages in enabling collaborative learning while protecting data privacy, it still faces an open challenge of incentivizing people to join the federated learning by contributing their computation power and data. An intuitive idea is to reward participants according to their contributions, following the existing incentive mechanism designs for many other scenarios [5]–[11]. Unfortunately, there are two main difficulties that make traditional incentive mechanisms unfit in federated learning. First, computing nodes do not share their decisions due to privacy concerns. Without the information of other nodes, it would be impossible for a participant to derive an optimal decision with close-form expression. Second, it is difficult to evaluate the contribution of participants to the accuracy of trained models. Evidences have shown that the relationship between model accuracy and the amount of training data is nonlinear [12], [13]. The model accuracy depends on the model complexity and data quality, and can hardly be predicted in advance. Without accurate evaluation of contributions, incentive mechanisms cannot correctively reward participants, leading to financial loss or low participation rate.

Another challenge, in parallel with incentive mechanism design, imposed by many intelligent IoT applications is to update models using fresh data, so that they can provide services with high accuracy to adopt to the new environment [14]. For example, the weather-related predication services [15], [16] always prefer the latest temperature and humidity data. Therefore, the model for weather prediction needs to be retrained periodically using recently collected data from sensors. This requirement has been advocated by a recently proposed concept called the Age of Information (AoI) [17], which is a new metric to quantify the freshness of collected data in IoT. Although there exist some preliminaries studies [18], [19], they cannot be combined with incentive mechanism design for federated learning.

In this paper, we propose a novel incentive mechanism design that integrates model updating using fresh data for federated learning in IoT applications. A parameter server, usually residing at the cloud, publishes a federated learning task with rewards. A number of edge computing nodes, each of which is in charge of some IoT devices, participate the

Y. Zhan, and S. Guo are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. E-mail: zhanyf1989@gmail.com, song.guo@polyu.edu.hk.

P. Li is with the School of Computer Science and Engineering, The University of Aizu, Japan. E-mail: pengli@u-aizu.ac.jp. P. Li is the corresponding author.

Z. Qu is with Hohai University and the Department of Computing, The Hong Kong Polytechnic University. E-mail: quzhihao@hhu.edu.cn.

D. Zeng is with the School of Computer Science, China University of Geoscience, China. E-mail: deze@cug.edu.cn.

federated learning by training local models using collected data from devices. The parameter server aims to minimize the total reward, while each edge node has its own interests of maximizing the revenue that is defined by the received reward from the parameter server minus its cost of data collection and model training. We formulate the problem as a Stackelberg game and derive the Nash Equilibrium that describes the steady state of the whole federated learning system, given the full knowledge of participants' contributions.

To address the challenges of unshared decisions and ambiguous contribution evaluation, we design an algorithm based on deep reinforcement learning (DRL), which can learn system states from historical training records. DRL approach has been widely used in mobile networks [20]–[24]. Furthermore, it can adjust the strategies of the parameter server and edge nodes, according to environmental changes that may impose different requirements on training data. The main contributions of this paper are summarized as follows.

- We study and formulate a Stackelberg game for federated learning in IoT by integrating an incentive mechanism and model updating requirements.
- We derive the Nash Equilibrium for the cases that participants share their decisions (e.g., the amount of data used for model training) and the parameter server can accurately evaluate their contributions to the training accuracy.
- We design an algorithm using DRL, so that the parameter server and edge nodes can dynamically adjust their strategies to optimize their interests, even without the knowledge of participants' decisions and accurate contribution evaluation.
- Extensive simulations are conducted to evaluate the performance of our proposals. The results show that the proposed learning-based approach matches the theoretical analysis.

The rest of this paper is organized as follows. Some important literature related to our work is reviewed in Section II. Section III presents the problem description. Section IV gives the incentive mechanism for scenarios of complete information sharing. Section V provides the DRL-based incentive mechanism of scenarios without knowing any prior information. Finally, Section VI evaluates the system performance by numerical experiments and Section VII concludes the paper.

II. Related Works

A. Federated Learning

As a natural extension of distributed learning, federated learning moves the training computation from centralized data centers to devices or computing sites at network edge, so that data can be kept at local storage to avoid the risks of privacy leakage. This concept of federated learning is initialized by Google [4] and later has attracted great research attentions [25]–[28]. Since training computation is distributed among devices connected with Internet, communication becomes the main bottleneck. McMahan et al. [25] have presented a practical model for the federated learning based on model averaging and conducted an extensive empirical evaluation. Tran et al. [26] have studied the federated learning over wireless networks by formulating an optimization problem that capture tradeoff between communication and computation cost. Zhan et al. [29] have proposed an experience-driven computation resource allocation scheme to improve the energy efficiency of federated learning by lowing CPU-cycle frequency of mobile devices who are faster in the training group.

B. Incentive Mechanism Design

Incentive mechanisms based on game or auction theories have been extensively studied in network and computation resource allocation. Xu et al. [30] have studied truthful incentive mechanism for scenarios where mobile crowdsensing tasks are time dependent. Li et al. [31] have summarized incentive mechanism for both open and sealed markets for device-to-device communications. An incentive mechanism for the temperature setting in shared spaces in smart buildings has been studied by [32]. Liu et al. [33] have proposed an auction-based incentive mechanism to motivate spare vehicle nodes to participate data caching in vehicle networks. In [34], Zhan et al. have proposed the incentive mechanism based on bargaining approach for crowdsensing. However, none of the above work can be applied to solve our problem due to the special challenges of unshared decisions and difficulties of contribution evaluation in federated learning.

III. Problem Description

In this section, we briefly introduce the basics of federated learning followed by the system model and problem formulation.

A. Edge-based Federated Learning

Edge-based federated learning is a promising distributed privacy-preserving machine learning technique that enables edge nodes to collaboratively train a shared global model without the need of uploading private local data to a central server. Assume that there are N edge nodes with local datasets $\{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_N\}$. We define $x_n \triangleq |\mathcal{X}_n|$, where $|\cdot|$ denotes the size of the dataset. As shown in Fig. 1, each edge node downloads a shared global model $\boldsymbol{\vartheta}$ from the parameter server and trains the model using its local data. Then, edge nodes upload the new weights or gradients (i.e., local model update) to the parameter server that updates the global model. Therefore, the total size of data samples from N edge nodes is $\sum_{n=1}^{N} x_n = X$. The loss function of the edge node n with dataset $\mathcal{X}_n$ is

$$F_n(\boldsymbol{\vartheta}) \triangleq \frac{1}{x_n} \sum_{j \in \mathcal{X}_n} f_j(\boldsymbol{\vartheta}),$$

where $f_j(\boldsymbol{\vartheta})$ is the loss function on the data sample j. The goal is to optimize the global loss function $F(\boldsymbol{\vartheta})$ by minimizing the weighted average of every edge node n's local loss function $F_n(\boldsymbol{\vartheta})$ on its local training samples [35], i.e.,

$$F(\boldsymbol{\vartheta}) \triangleq \frac{\Sigma_{j \in \bigcup_n \mathcal{X}_n} f_j(\boldsymbol{\vartheta})}{|\bigcup_n \mathcal{X}_n|} = \frac{\sum_{n=1}^{N} x_n F_n(\boldsymbol{\vartheta})}{X},$$

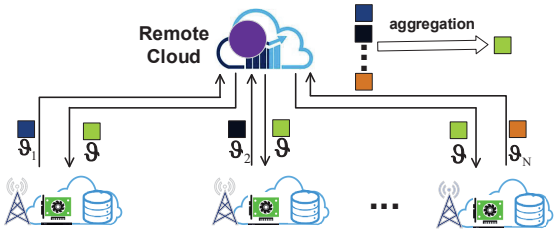


Fig. 1: Training Process of federated learning.

and

$$\boldsymbol{\vartheta}^* = \arg\min F(\boldsymbol{\vartheta}). \tag{1}$$

Due to the inherent complexity of many machine learning models, it is hard to find a close-form solution to Eq. (1). Therefore, it is often solved through gradient-descent techniques [25].

B. System Model

We consider one parameter server resides in the cloud, who wants to motivate a set $\mathcal{N} = \{1, 2, \cdots, N\}$ of edge nodes to participate in the model training. These edge nodes connect to the cloud via the Internet backbone. In order to participate in the model training, each edge node collects data from the IoT devices and then trains a model shared with other edge nodes. In this paper, we model systems in a quasi-static state, which means no edge node joins or leaves.

The parameter server generates only one training task in each period by announcing a total payment $\tau > 0$, while each edge node decides its level of participation based on the parameter server's payment. Without loss of generality, each edge node $n \in \mathcal{N}$ maintains a dataset $\mathcal{X}_n$ and $\mathcal{X}_n \bigcap \mathcal{X}_m = \emptyset, \forall m \in \mathcal{N}, m \neq n$. If $\mathcal{X}_n = \emptyset$, edge node n does not participate in the model training. The training cost of edge node n includes two parts, computational cost and communication cost, which are proportional to the amount of data used for training. We let c_n^{cmp} and c_n^{com} denote unit computational cost and communication cost, respectively. Assume that all the training data in each edge node has the same quality and is independently and identically distributed, then the reward received by edge node n is proportional to x_n. Therefore, the utility of edge node n is defined by

$$u_n(x_n, \boldsymbol{x}_{-n}) = \frac{x_n}{\sum_{m=1}^{N} x_m}\tau - c_n^{com}x_n - c_n^{cmp}x_n, \tag{2}$$

where $\boldsymbol{x}_{-n} = (x_1, x_2, \cdots, x_{n-1}, x_{n+1}, \cdots, x_N)$ is the training strategies of others except edge node n. The training data contributed by each edge node can help the parameter server train a better model. In this case, the computational cost of edge node n is $c_n^{cmp}x_n$, and the communication cost is $c_n^{com}x_n$. We use $u(\tau)$ to denote the utility of the parameter server, which indicates the gain of model accuracy minus the the total rewards paid to edge nodes. We conduct experiments to measure the model accuracy under different amount of training data and show the results in From Fig. 2. We observe

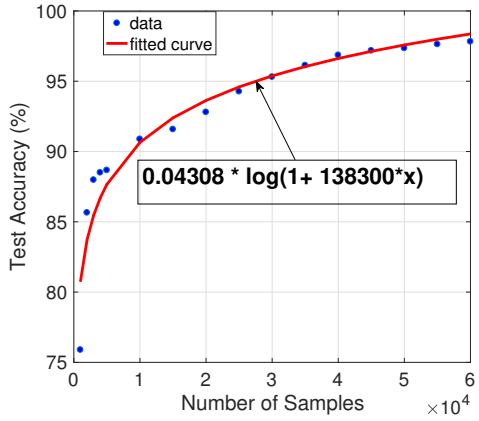


Fig. 2: Test accuracy with varying the number of training samples on MNIST dataset.

that the test accuracy of training model can be regarded as a concave function with respect to the amount of training data. Therefore, the utility of the parameter server is defined as

$$u(\tau) = \lambda g(X) - \tau, \tag{3}$$

where $\lambda > 0$ is a system parameter and $g(X)$ is a concave function with respect to the amount of training data.

C. Problem Formulation

We formulate the incentive mechanism for federated learning as a Stackelberg game [36] in each training period. There are two stages in this mechanism. In the first stage, the parameter server announces a total reward of τ, followed by the second stage that each user determines its training strategy to maximize its own utility. Therefore, the parameter server is the leader and the edge nodes are the followers in this Stackelberg game. Game theory is a powerful framework to analyze the interactions among multiple players who act in their own interests, such that no player has the incentive to deviate unilaterally. Moreover, by using the intelligence of each player, game theory is a useful tool for designing decentralized mechanisms with low complexity, such that the players can self-organize into a mutually satisfactory solution. The strategy of the parameter server is the reward τ and that of edge node n is the amount of contributed training data, which is denoted by x_n. Note that the second stage of the game can be considered as a non-cooperative game. For any reward τ given by parameter server and other nodes' decisions $\boldsymbol{x}_{-n}$, edge node n would like to determine a optimal decision x_n to maximize its utility in terms of obtained reward and cost, i.e., $\max_{x_n} u_n(x_n, \boldsymbol{x}_{-n})$.

Definition 1. ***Nash Equilibrium****. A set of strategies $\boldsymbol{x}^* = (x_1^*, x_2^*, \cdots, x_N^*)$ is a Nash equilibrium [36] of the second stage game if for any edge node n,*

$$u_n(x_n^*, \boldsymbol{x}_{-n}^*) \geq u_n(x_n, \boldsymbol{x}_{-n}^*),$$

for any $x_n \geq 0$.

The existence of the Nash equilibrium is important, since its strategy profile is stable (no player has an incentive to make

a unilateral change) whereas a non-Nash equilibrium strategy profile is unstable.

IV. Incentive Mechanism with Full Information

In this section, we first prove that for any given τ, the second-stage game has a unique Nash equilibrium. For the parameter server in the first stage, it needs to select a value of τ to maximize its utility in Eqn. (3), i.e., $\max_{\tau} u(\tau)$. In Section IV-B, we prove that the game has a unique Stackelberg equilibrium.

A. Edge node Participation

To study the Nash equilibrium of the second stage game in federated learning game, we derive the first order derivative of $u_n(x_n, \boldsymbol{x}_{-n})$ with respect to x_n as

$$\frac{\partial u_n(x_n, \boldsymbol{x}_{-n})}{\partial x_n} = \frac{-\tau x_n}{(\sum_{m=1}^{N} x_m)^2} + \frac{\tau}{\sum_{m=1}^{N} x_m} - c_n^{com} - c_n^{cmp}. \quad (4)$$

Based on the first-order derivative of u_n, we can derive the second-order derivative of $u_n(x_n, \boldsymbol{x}_{-n})$ with respect to x_n as

$$\frac{\partial^2 u_n(x_n, \boldsymbol{x}_{-n})}{\partial {x_n}^2} = -\frac{2\tau \Sigma_{m\neq n} x_m}{(\sum_{m=1}^{N} x_m)^3} < 0. \quad (5)$$

Lemma 1. *If the following conditions are satisfied, there exists a Nash equilibrium in the game [36].*

- *The player set is finite.*
- *The strategy sets are closed, bounded, and convex.*
- *The utility functions are continuous and quasi-concave in the strategy space.*

Based on Eqn. (2), we know that $x_n \leq \frac{\tau}{c_n^{com}+c_n^{cmp}}$ because u_n must be positive. Therefore, from Lemma 1, we can derive that there exists a Nash equilibrium in the second stage game. Setting $\frac{\partial u_n(x_n, \boldsymbol{x}_{-n})}{\partial x_n} = 0$, we have

$$\frac{-\tau x_n}{(\sum_{m=1}^{N} x_m)^2} + \frac{\tau}{\sum_{m=1}^{N} x_m} - c_n^{com} - c_n^{cmp} = 0. \quad (6)$$

Solving x_n in Eqn. (6), we obtain

$$x_n = \sqrt{\frac{\tau \sum_{m\neq n} x_m}{c_n^{com} + c_n^{cmp}}} - \sum_{m\neq n} x_m. \quad (7)$$

If the right hand side of Eqn. (7) is positive, it is also the best response strategy of edge node n, due to the concavity of u_n. Otherwise, edge node n does not participate in the edge-based federated learning. Furthermore, if the right hand side of Eqn. (7) is more than d_n, then edge node n participates with its best response by setting $x_n = d_n$. Therefore, we have

$$x_n = \begin{cases} 0, if \quad \tau \leq (c_n^{com} + c_n^{cmp}) \sum_{m\neq n} x_m; \\ \sqrt{\frac{\tau \sum_{m\neq n} x_m}{c_n^{com}+c_n^{cmp}}} - \sum_{m\neq n} x_m, x_n \in (0, d_n); \\ d_n, otherwise. \end{cases} \quad (8)$$

Theorem 1. *For any $m \in \mathcal{M} \subseteq \mathcal{N}$ participating the game (i.e., $x_m \in (0, d_n)$), its optimal strategy is*

$$x_m^* = \frac{(M-1)\tau}{\Sigma_{n\in\mathcal{M}(c_n^{com}+c_n^{cmp})}}(1 - \frac{(M-1)(c_m^{com}+c_m^{cmp})}{\Sigma_{n\in\mathcal{M}}(c_n^{com}+c_n^{cmp})}). \quad (9)$$

Proof. According to Eqn. (8), for any $m \in \mathcal{M}$, we have

$$\sum_{n=1}^{M} x_n^* = \sqrt{\frac{\tau \sum_{n\in\mathcal{M}\setminus\{m\}} x_n^*}{c_m^{com} + c_m^{cmp}}}. \quad (10)$$

By setting $\xi = \sum_{n=1}^{M} x_n^*$, we can derive that

$$\begin{cases} x_1^* = \xi - \frac{\xi^2(c_1^{com}+c_1^{cmp})}{\tau} \\ x_2^* = \xi - \frac{\xi^2(c_2^{com}+c_2^{cmp})}{\tau} \\ \vdots \\ x_M^* = \xi - \frac{\xi^2(c_M^{com}+c_M^{cmp})}{\tau} \end{cases}, \quad (11)$$

Therefore,

$$\xi = M\xi - \frac{\xi^2 \sum_{n=1}^{M}(c_n^{com} + c_n^{cmp})}{\tau}. \quad (12)$$

Based on Eqn. (12), we have

$$\xi = \frac{(M-1)\tau}{\sum_{n=1}^{M}(c_n^{com} + c_n^{cmp})}. \quad (13)$$

By plugging Eqn. (13) into Eqn. (11), we can derive

$$x_m^* = \frac{(M-1)\tau}{\Sigma_{n\in\mathcal{M}(c_n^{com}+c_n^{cmp})}}(1 - \frac{(M-1)(c_m^{com}+c_m^{cmp})}{\Sigma_{n\in\mathcal{M}}(c_n^{com}+c_n^{cmp})}).$$

Since $x_m \in (0, d_n)$, we can derive that $\frac{(M-1)\tau}{\Sigma_{n\in\mathcal{M}(c_n^{com}+c_n^{cmp})}}(1 - \frac{(M-1)(c_m^{com}+c_m^{cmp})}{\Sigma_{n\in\mathcal{M}}(c_n^{com}+c_n^{cmp})}) \in (0, d_n)$. □

B. parameter Server Payment Determination

According to the above analysis, the parameter server, which is the leader in the Stackelberg game, knows that there exists a unique Nash equilibrium among edge nodes under any given value of τ. Therefore, the parameter server can maximize its utility by choosing the optimal τ. Substituting Eqn. (9) into Eqn. (3), we have

$$u(\tau) = \lambda g(\sum_{n=1}^{M} x_n^*) - \tau. \quad (14)$$

Theorem 2. *There exists a unique Stackelberg equilibrium $(\tau^*, \boldsymbol{x}^*)$, where τ^* is the unique maximizer of the parameter server utility in Eqn. (3) over $\tau \in [0, \infty)$.*

Proof. The first order derivative of $u(\tau)$ is

$$\begin{aligned} \frac{\partial u(\tau)}{\partial \tau} &= \lambda g'(X)\frac{\partial X}{\partial \tau} - 1 \\ &= \lambda g'(X)(\frac{\partial x_1^*}{\partial \tau} + \frac{\partial x_2^*}{\partial \tau} + \cdots + \frac{\partial x_N^*}{\partial \tau}) - 1. \end{aligned}$$

Hence, the second order derivative of $u(\tau)$ is

$$\begin{aligned}\frac{\partial^2 u(\tau)}{\partial \tau^2} &= \lambda g''(X)(\frac{\partial x_1^*}{\partial \tau}+\frac{\partial x_2^*}{\partial \tau}+\cdots+\frac{\partial x_M^*}{\partial \tau})^2 \\ + \quad &\lambda g'(X)(\frac{\partial^2 x_1^*}{\partial \tau^2}+\frac{\partial^2 x_2^*}{\partial \tau^2}+\cdots+\frac{\partial^2 x_M^*}{\partial \tau^2}) \\ = \quad &\lambda g''(X)(\sum_{n=1}^{M}\frac{(M-1)}{\Sigma_{m\in\mathcal{M}(c_m^{com}+c_m^{cmp})}}(1- \\ &\frac{(M-1)(c_n^{com}+c_n^{cmp})}{\Sigma_{m\in\mathcal{M}}(c_m^{com}+c_m^{cmp})}))^2\end{aligned}$$

Since $g(X)$ is a concave function of X, then we can derive that $\frac{\partial^2 u(\tau)}{\partial \tau^2} < 0$. Therefore the utility of the parameter server $u(\tau)$ defined in Eqn. (3) is a strictly concave function of τ for $\tau \in [0, \infty)$. Since the value of $u(\tau)$ is 0 for $\tau = 0$ and goes to $-\infty$ when τ goes to ∞, it has a unique maximizer τ^*. Therefore, there exists a unique Stackelberg equilibrium. □

V. DRL-based Incentive Mechanism with Incomplete Information

In this section, we study the DRL-based incentive mechanism design without any prior information. We first introduce the basic learning mechanism of applying DRL into the decentralized incentive mechanism design problem. We then describe how we transform it into a learning task and design a DRL approach to determine the optimal strategies for the parameter server and edge nodes, respectively.

A. A Basic Learning Mechanism

Unlike existing approaches, DRL strives to learn a general action decision from past experiences based on the current state and the given reward. The workflow of DRL-based incentive mechanism is illustrated in Fig. 3, where the parameter server acts as a leader who interacts with the environment in the DRL setting. At each training period t, the parameter server agent observes a state $\boldsymbol{s}_t$ and determines an action τ_t. When this action is done, edge nodes interact with each other to determine their optimal participation level strategies. Since each edge node does not know any information about the decisions of other edge nodes, they need to learn the optimal strategy. We can use an offline mode to train the edge nodes. All the edge nodes interact with each other in a non-cooperative game simulation environment to learn the Nash equilibrium. After each edge node learns the Nash equilibrium, they update the model based on their local data and upload the updated model to the parameter server. Utile now, the t-th training period ends. Then the current state transits to the next state $\boldsymbol{s}_{t+1}$ and the parameter server agent receives a reward r_t. If the parameter server agent continues this process, it gets accumulated rewards after every action until done. The objective of DRL is to find an optimal policy π mapping a state to an action that maximizes the expected discounted accumulated reward.

B. Design Details

Our DRL-based incentive mechanism uses an actor-critic DRL model based on the state-of-the-art policy gradient method Proximal policy optimization (PPO) [37].

1) State space of the parameter server: We consider the practical scenario where the parameter server needs to train a machine learning model periodically. The parameter server can only observe the past strategies of edge nodes. Hence, the state space of the parameter server's DRL formulation consists of two components, including its past payment strategy history $\{\tau_{t-L}, \tau_{t-L+1}, \cdots, \tau_{k-1}\}$ and edge nodes' past participation level history $\{\boldsymbol{x}_{t-L}, \boldsymbol{x}_{t-L+1}, \cdots, \boldsymbol{x}_{t-1}\}$. Integrating all these components together, the state input of the parameter server can be represented as $\boldsymbol{s}_t = \{\tau_{t-L}, \boldsymbol{x}_{t-L}, \tau_{t-L+1}, \boldsymbol{x}_{t-L+1}, \cdots, \tau_{t-1}, \boldsymbol{x}_{t-1}\}$.

2) State space of the edge nodes: In each training period, edge nodes should determine their optimal training strategies. Since the edge nodes do not know any prior information of the others, it is very hard for the them to determine the optimal training strategies under the given payment. Hence, we use a offline training mode where edge nodes learn their optimal training strategies (i.e., Nash equilibrium) in a simulated environment. As has formulated in Section III-B, the training strategies of edge nodes are affected by all the nodes' private information and the parameter server's payment strategy. But for each edge node, it could observe the history of other edge nodes' training strategies and current parameter server's payment. Therefore, the state input of edge node n is $\boldsymbol{s}_{t,k}^n = \{\boldsymbol{x}_{-n}^{t,k-L}, \boldsymbol{x}_{-n}^{t,k-L+1}, \cdots, \boldsymbol{x}_{-n}^{t,k-1}, \tau^t\}$.

3) Policy of the parameter server: When receiving a state $\boldsymbol{s}_t$, the parameter server agent needs to take an action τ_t to incentivize the edge nodes. The action space can be represented as $\tau_t \in [0, +\infty)$. Since the action space of the parameter server agent is continuous, there are infinite $\{state, action\}$ pairs so that they can not be stored in a tabular form and solve the problem by Q learning [38], [39]. To address this issue, we use a neural network to represent the policy $\pi_{\boldsymbol{\theta}}$. Then we can represent the parameter server's policy as $\pi(\tau_t|\boldsymbol{s}_t, \boldsymbol{\theta}) \rightarrow [0, \infty)$.

4) Policies of edge nodes: In the simulated non-cooperative game environment, each edge node needs to determine the training strategy $x_{t,k}^n$ when receiving a state $\boldsymbol{s}_{t,k}^n$. Since the action space of each edge node is also continuous, the neural network is also used to represent the policy $\pi_n(x_{t,k}^n|\boldsymbol{s}_{t,k}^n, \boldsymbol{\theta}_n) \rightarrow [0, \infty)$. Edge nodes continuously learn until they reach the Nash equilibrium under the current parameter server's payment strategy.

5) Reward of the parameter server: When applying an action τ_t to the state $\boldsymbol{s}_t$, the parameter server agent receives a reward r_t from the environment. Considering the model formulation in Section III-B, we define the reward r_t as the utility function of the parameter server which satisfies $r_t = u(\tau_t)$.

6) Reward of edge nodes: In the t-th training period, at the k-th game, each edge node determines a training strategy $x_{t,k}^n$. As has formulated in Section III-B, the reward $r_{t,k}^n$ can be defined as $r_{t,k}^n = u_n(x_{t,k}^n, \boldsymbol{x}_{t,k}^{-n}, \tau_t)$.

C. DRL Training Methodology

We train the DRL-based incentive mechanism model based on actor-critic model [40] since the actor-critic architecture

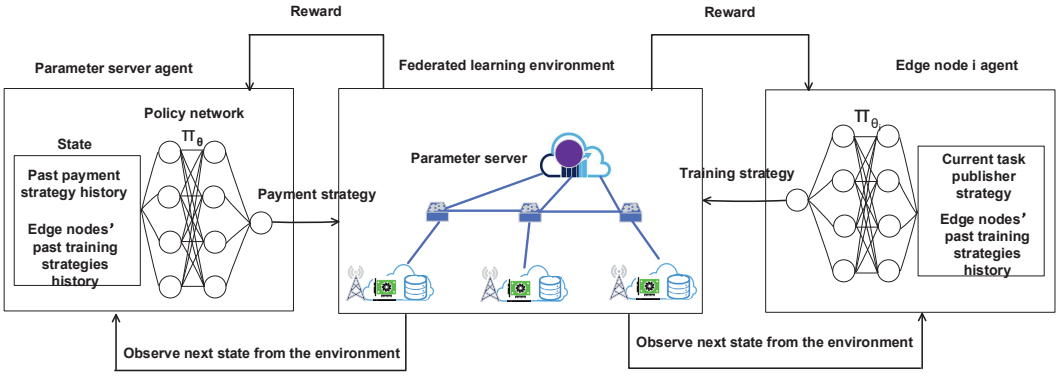


Fig. 3: The workflow of using DRL for incentive mechanism design in federated learning.

well matches our scenario and it also has shown successful applications in many other applications [41]–[43]. For the parameter server, it maintains a policy $\pi(\tau_t|\boldsymbol{s}_t,\boldsymbol{\theta})$ (the actor network) and an estimation of the value function $V(\boldsymbol{s}_t,\boldsymbol{\omega})$ (the critic network), where $\boldsymbol{\theta}$ is the policy parameter and $\boldsymbol{\omega}$ is the parameter of the value function. Each edge node also maintains a policy $\pi_n(x_t^n|\boldsymbol{s}_t^n,\boldsymbol{\theta}_n)$ and an estimate of the value function $V_n(\boldsymbol{s}_t^n,\boldsymbol{\omega}_n)$. In each training period, the parameter server determines a policy τ^t. After all the edge nodes receive the payment issued by the parameter server, each edge node agent continues to take actions $\boldsymbol{x}$ after all edge nodes determining their actions. The edge node agents update both the policy and value functions based on the returns of every D actions. As the training continues, each edge node will learn the optimal training strategy under the payment of τ_t. The parameter server agent continues to determine the payment strategy after receiving the optimal training strategy of edge nodes. It also updates both the policy and value function based on the returns of every D training period. The parameters of the deep neural networks are selected through fine-tuning. Especially, the actor network has two hidden fully-connected layers, each of which contains 200 nodes and 50 nodes, respectively. The critic network also has two hidden fully-connected layers, each of which has 200 nodes and 50 nodes, respectively.

The policy gradient methods [44] are fundamental to recent breakthroughs in using deep neural networks for updating the DRL model. However, getting good results via policy gradient methods is challenging because they are sensitive to the choice of stepsize. If two small, the progress is hopelessly slow. But too large, the signal is overwhelmed by the noise, or one might see catastrophic drops in performance. They also often have very poor sample efficiency, taking millions (or billions) of timesteps to learn simple tasks [45]. Some works have sought to eliminate these flaws of policy gradient method with approaches like TRPO [46] and ACER [47], by constraining or otherwise optimizing the size of a policy update. However, they are too complicated to implement. PPO algorithm [37] strikes a balance between ease of implementation, sample complexity, and ease of tuning, trying to compute an update at each step that minimizes the cost function while ensuring the deviation from the previous policy is relative. Hence, in this paper, the training process of the parameter server agent and edge node agent employs the state-of-the-art policy optimization approach PPO. Once the actor-critic network is well trained, parameter server and edge nodes can determine their own strategies based on the output of their actor networks, respectively.

During the learning-based incentive mechanism execution, given the observation information as input, the participants utilize their own actor networks to generate actions, and thus the computational complexity is merely based on a fully-connected deep neural network. According to [48], the time complexity of a fully-connected deep neural network is determined by the number of multiplication operations, which is $O(\sum_{f=1}^{F}\epsilon_f\epsilon_{f-1})$, and ϵ is the number of neural units in fully-connected layer f. In our design, we use two-fully hidden layers in the actor networks. Meanwhile, since modern edge nodes are becoming stronger and stronger, they can afford the computational overhead incurred by such a kind of actor networks.

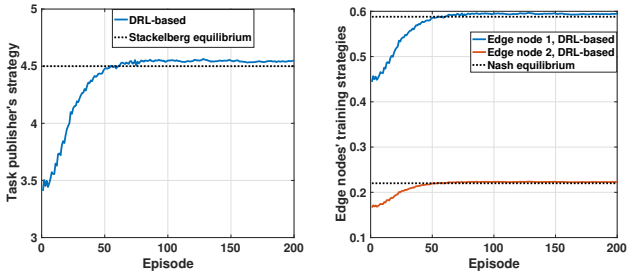
VI. Performance Evaluation

A. Experiment Settings

In this section, extensive experiments are conducted to evaluate the performance of the proposed incentive mechanism in federated learning. We conduct our experiments using Tensorflow 1.9 on Ubuntu 16.04 LTS. The parameters of DRL agent are selected through fine-tuning. Specially, the actor network has two hidden fully-connected layers, each of which contains 200 nodes and 50 nodes, respectively. The critic network also has two hidden fully-connected layers, each of which has 200 nodes and 50 nodes, respectively. We set $D = 20$ and $L = 5$ by default. In this paper, we use $g(X) = 10 * \ln(1+X)$ to denote the benefit of the parameter server in the federated learning.

B. Experiment Results

We first study the convergence of the proposed DRL-based incentive mechanism when there are two edge nodes. We set $c_1^{cmp} = c_2^{cmp} = 1$, $c_1^{com} = 0.5$, $c_2^{com} = 3$ and $\lambda = 10$. As shown in Fig. 4(a), the parameter server's pricing strategy



(a) Strategy of parameter server. (b) Strategies of edge nodes.

Fig. 4: Convergence of DRL-based incentive mechanism.

converges to the Stackelberg equilibrium. In Fig. 4(b) the edge nodes' training strategies also converge to the Nash equilibrium. From Fig. 4, we can see that the incentive mechanism designed in this paper can motivate the edge nodes to participate in the federated learning. Also, the DRL-based incentive mechanism can learn the optimal strategies for the parameter server and edge nodes.

We then study the influence of training cost. Specifically, we study the training cost by changing edge node 1's unit communication cost (i.e., c_1^{com}) from 0.5 to 2.5. The unit computational costs of edge nodes are $c_1^{cmp} = 1$ and $c_2^{cmp} = 1$, respectively. The unit communication cost of node 2 is $c_2^{com} = 3$. As shown in Fig. 5(a), we observe that the parameter server decreases its payment as the increasing of training cost. For example, when the training cost of node 1 is 1.5, the parameter server makes the payment of 4.5 to incentivize the edge nodes to participate in the federated learning. However, when the unit training cost of node 1 increases to 3.5, a lower payment of 2.5 will be made to edge nodes. In Fig. 5(b), we observe that the edge nodes' participation level decreases when the training cost increases. For example, when edge node 1's training cost is 1.5, edge node 1 participates the federated learning with participation level of 0.6. But when the training cost increases to 3.5, edge node 1 participates in the federated learning with only 0.18 participation level. As shown in Fig. 5(c), it is expected that the parameter server's utility decreases as the training costs of edge nodes increase. In Fig. 5(c), we compare the DRL-based incentive mechanism with the random and greedy approaches. In random approach, the parameter server determines the payment in each training round randomly, while in greedy the parameter server determines its strategy according to the best strategy in the past training periods. In the actual scenario, the parameter server does not know any prior information about the edge nodes and the relationship between model accuracy and the amount of training data. Therefore, the parameter server could only determines the payment to the edge nodes randomly. In Fig. 5(c), we can find that the DRL-based incentive mechanism is much better than the baselines.

Fig. 6 shows the impact of number of edge nodes. In this experiment, we randomly choose edge node's unit communication cost c_n^{com} and unit computation cost c_n^{cmp} within $(0, 2]$. As shown in Fig. 6(a), we observe that the parameter server's utility increases when the number of edge nodes increases. For example, when there are only 2 edge nodes, the parameter server's utility is nearly 3.2. However, when the number of edge nodes increases to 10, a higher utility of 7.2 can be obtained by the parameter server. In Fig. 6(b), we observe that the average utility of edge nodes decreases as more and more edge nodes participate in the federated learning. That is, although the parameter server increases its payment to incentivize more edge nodes to participate in the federated learning in Fig. 6(c), it leads to more competition among edge nodes. Therefore, each edge node could obtain less reward from the parameter server. For example, the average utility of edge nodes decreases by 94.8% as the number of edge nodes changes from 2 to 10.

VII. Conclusion

In this paper, we have studied the distributed machine learning on edge nodes, federated learning, in which the training model is distributed to participating edge nodes performing training tasks on their local data. Though it has the benefit of data privacy, it is not clear how the incentive mechanism impacts on the utility of the parameter server. We address this issue by providing an incentive mechanism based on Stackelberg game approach. First, we analyze the uniqueness of the Nash equilibrium in the second-stage of the Stackelberg game and the uniqueness of the Stackelberg equilibrium in the first-stage. Second, due to the unique challenges of unshared information and difficulties of contribution evaluation in federated learning, we propose the DRL-based incentive mechanism to address these issues. Finally, numerical experiments have been done to further demonstrate the efficiency of the DRL-based incentive mechanism as compared with the baseline approaches.

Acknowledgment

This work was supported by the General Research Fund of the Research Grants Council of Hong Kong (PolyU 152221/19E), National Natural Science Foundation of China (Grant 61872310), JSPS Grants-in-Aid for Scientific Research JP19K20258, Basic Research-Free Exploration Project of Shenzhen under Grant JCYJ20170818103849343, and China Postdoctoral Science Foundation 2019M661709.

References

[1] S. Yao, Y. Zhao, A. Zhang, S. Hu, H. Shao, C. Zhang, L. Su, and T. Abdelzaher, "Deep learning for the internet of things," *Computer*, vol. 51, no. 5, pp. 32–41, May 2018.

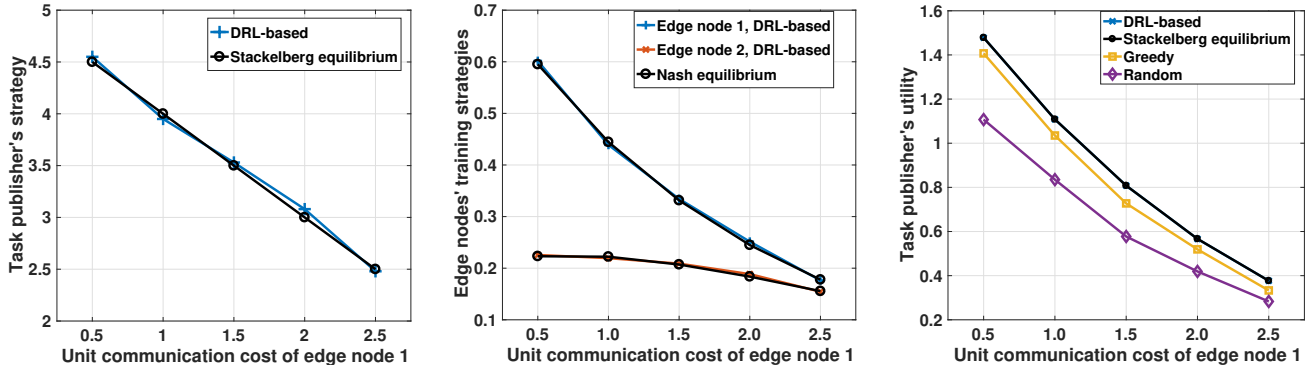
[2] P. Liu, Y. Ding, and T. Fu, "Optimal throwboxes assignment for big data multicast in vdtns," *Wireless Networks*, pp. 1–11, 2019.

[3] P. Li, T. Miyazaki, K. Wang, S. Guo, and W. Zhuang, "Vehicle-assist resilient information and network system for disaster management," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 3, pp. 438–448, July 2017.

[4] B. A. y. Arcas, G. Andrew, D. Bacon, K. Bonawitz, and et al., "Federated learning: Collaborative machine learning without centralized training data," https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, april 6, 2017.

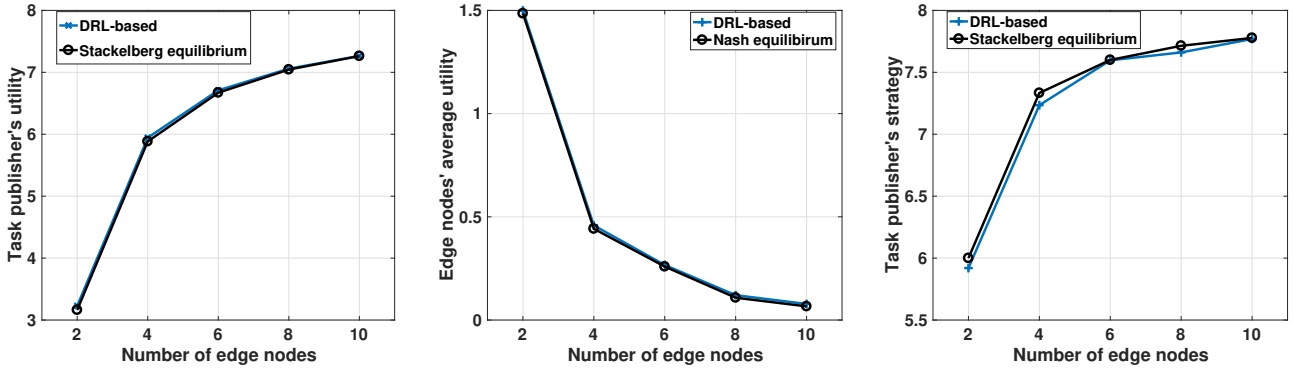
[5] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. of ACM MobiCom*, 2012, pp. 173–184.

[6] X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, "Truthful incentive mechanisms for crowdsourcing," in *Proc. of IEEE INFOCOM*, 2015, pp. 2830–2838.



(a) Strategy of parameter server. (b) Strategy of edge nodes. (c) Parameter server's utility.

Fig. 5: Performance of the incentive mechanism when varying the unit communication cost of edge node 1.



(a) Parameter server's utility. (b) Edge nodes' average utility. (c) Strategy of parameter server.

Fig. 6: Performance of the incentive mechanism when varying the number of edge nodes.

[7] X. Zhang, Z. Yang, Z. Zhou, H. Cai, L. Chen, and X. Li, "Free market of crowdsourcing: Incentive mechanism design for mobile sensing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3190–3200, 2014.

[8] D. Zhao, X.-Y. Li, and H. Ma, "How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint," in *Proc. of IEEE INFOCOM*, 2014, pp. 1213–1221.

[9] Z. Feng, Y. Zhu, Q. Zhang, L. M. Ni, and A. V. Vasilakos, "Trac: Truthful auction for location-aware collaborative sensing in mobile crowdsourcing," in *Proc. of IEEE INFOCOM*, 2014, pp. 1231–1239.

[10] Y. Liu, C. Xu, Y. Zhan, Z. Liu, J. Guan, and H. Zhang, "Incentive mechanism for computation offloading using edge computing: A stackelberg game approach," *Computer Networks*, vol. 129, pp. 399–409, 2017.

[11] Y. Zhan, C. H. Liu, Y. Zhao, J. Zhang, and J. Tang, "Free market of multi-leader multi-follower mobile crowdsensing: An incentive mechanism design by deep reinforcement learning," *IEEE Transactions on Mobile Computing*, 2019.

[12] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: an efficient dynamic resource scheduler for deep learning clusters," in *Proc. of ACM EuroSys*, 2018, pp. 1–14.

[13] Y. Zhan, S. Guo, P. Li, K. Wang, and Y. Xia, "Big data analytics by crowdlearning: Architecture and mechanism design," *IEEE Network*, 2019.

[14] H. Tian, M. Yu, and W. Wang, "Continuum: A platform for cost-aware, low-latency continual learning," in *Proc. of ACM SoCC*, 2018, pp. 26–40.

[15] A. Grover, A. Kapoor, and E. Horvitz, "A deep hybrid model for weather forecasting," in *Proc. of ACM SIGKDD*, 2015, pp. 379–386.

[16] S. E. Haupt and B. Kosovic, "Big data and machine learning for applied weather forecasts: Forecasting solar power for utility operations," in *Proc. of IEEE SSCI*, 2015, pp. 496–501.

[17] C. Li, S. Li, and Y. T. Hou, "A general model for minimizing age of information at network edge," in *Proc. of IEEE INFOCOM*, 2019, pp. 118–126.

[18] S. K. Kaul, R. D. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. of IEEE INFOCOM*, 2012, pp. 2731–2735.

[19] S. K. Kaul, M. Gruteser, V. Rai, and J. B. Kenney, "Minimizing age of information in vehicular networks," in *Proc. of IEEE SECON*, 2011, pp. 350–358.

[20] K. Ota, M. S. Dao, V. Mezaris, and F. G. De Natale, "Deep learning for mobile multimedia: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 3s, pp. 1–22, 2017.

[21] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[22] 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 of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2019.

[23] L. Gu, D. Zeng, W. Li, S. Guo, A. Y. Zomaya, and H. Jin, "Intelligent vnf orchestration and flow scheduling via model-assisted deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, 2019.

[24] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Network*, vol. 33, no. 3, pp. 26–33, 2019.

[25] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. of AISTATS*, 2017, pp. 1273–1282.

[26] N. H. Tran, W. Bao, A. Zomaya, N. Minh N.H., and C. S. Hong, "Federated learning over wireless networks: Optimization model design and analysis," in *Proc. of IEEE INFOCOM*, 2019, pp. 1387–1395.

[27] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, 2018.

[28] J. Mills, J. Hu, and G. Min, "Communication-efficient federated learning for wireless edge intelligence in iot," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[29] Y. Zhan, P. Li, and G. Song, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *Proc. of IPDPS*, 2020.

[30] J. Xu, J. Xiang, and D. Yang, "Incentive mechanisms for time window dependent tasks in mobile crowdsensing," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 6353–6364, 2015.

[31] P. Li and S. Guo, "Incentive mechanisms for device-to-device communications," *IEEE Network*, vol. 29, no. 4, pp. 75–79, 2015.

[32] T. Wang, Y. Xu, C. Withanage, L. Lan, S. D. Ahipa?ao?lu, and C. A. Courcoubetis, "A fair and budget-balanced incentive mechanism for energy management in buildings," *IEEE Transactions on Smart Grid*, vol. 9, no. 4, pp. 3143–3153, 2018.

[33] J. Liu, W. Wang, D. Li, S. Wan, and H. Liu, "Role of gifts in decision making: An endowment effect incentive mechanism for offloading in the iov," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6933–6951, 2019.

[34] Y. Zhan, Y. Xia, and J. Zhang, "Incentive mechanism in platform-centric mobile crowdsensing: A one-to-many bargaining approach," *Computer Networks*, vol. 132, pp. 40–52, 2018.

[35] H. Zhu and Y. Jin, "Multi-objective evolutionary federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[36] R. B. Myerson, *Game theory*. Harvard university press, 2013.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[38] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[39] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[40] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. of NIPS*, 2000, pp. 1008–1014.

[41] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," in *Proc. of ICLR*, 2017.

[42] Z. Xu, J. Tang, C. Yin, Y. Wang, and G. Xue, "Experience-driven congestion control: When multi-path tcp meets deep reinforcement learning," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1325–1336, 2019.

[43] C. H. Liu, Z. Chen, and Y. Zhan, "Energy-efficient distributed mobile crowd sensing: A deep learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1262–1276, 2019.

[44] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. of NeurIPS*, 2000, pp. 1057–1063.

[45] OpenAI, "Proximal policy optimization," https://openai.com/blog/openai-baselines-ppo/, july 20, 2017.

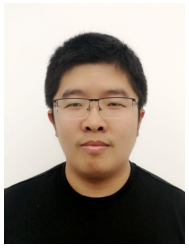
[46] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. of ICML*, 2015, pp. 1889–1897.

[47] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT press, 2016.

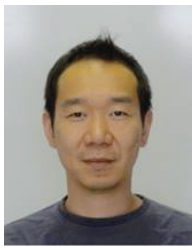
Yufeng Zhan received his Ph.D. degree in the School of Automation from Beijing Institute of Technology, China, in 2018. He is currently a Post-Doc in the Department of Computing with The Hong Kong Polytechnic University. His research interests include mobile computing, machine learning and networked control systems.

Peng Li received his BS degree from Huazhong University of Science and Technology, China, in 2007, the MS and PhD degrees from the University of Aizu, Japan, in 2009 and 2012, respectively. Dr. Li is currently an Associate Professor in the University of Aizu, Japan. His research interests mainly focus on cloud computing, Internet-of-Things, big data systems, as well as related wired and wireless networking problems. Dr. Li has published over 100 technical papers on prestigious journals and conferences. He won the Young Author Award of IEEE Computer Society Japan Chapter in 2014. He won the Best Paper Award of IEEE TrustCom 2016. He supervised students to win the First Prize of IEEE ComSoc Student Competition in 2016. Dr. Li serves as the guest editor of several international journal special issues and he is the editor of IEICE Transactions on Communications. He is a member of IEEE.



Zhihao Qu received his B.S. and Ph.D. degree in computer science from Nanjing University, Nanjing, China, in 2009, and 2018, respectively. He is currently an assistant researcher in the College of Computer and Information, Hohai University, China. His research interests are mainly in the areas of wireless networks, edge computing, and distributed machine learning.

Deze Zeng is currently a Full Professor in School of Computer Science, China University of Geosciences, Wuhan, China. He received his Ph.D. and M.S. degrees in computer science from University of Aizu, Aizu-Wakamatsu, Japan, in 2013 and 2009, respectively. He received his B.S. degree from School of Computer Science and Technology, Huazhong University of Science and Technology, China in 2007. His current research interests mainly focus on edge computing, and related technologies like network function virtualization, machine learning, and IoT. He has authored 3 books and over 100 papers in refereed journals and conferences in these areas. He also received 3 best paper awards from IEEE/ACM conferences and the IEEE Systems Journal Annual Best Paper Award of 2017. He serves in editorial boards of Journal of Network and Computer Applications and guest editors of many prestigious journals. He has been the in organization or program committees of many international conferences including ICPADS, ICA3PP, CollaberateCom, MobiQuitous, ICC, Globecom. He is a member of IEEE, and senior member of CCF.



Song Guo received his Ph.D. in computer science from the University of Ottawa. He is currently a full professor in the Department of Computing with The Hong Kong Polytechnic University. Prior to joining PolyU, he was a full professor with the University of Aizu, Japan. His research interests are mainly in the areas of cloud and green computing, big data, wireless networks, and cyber-physical systems. He has published over 300 conference and journal papers in these areas and received multiple best paper awards from IEEE/ACM conferences. His research has been sponsored by JSPS, JST, MIC, NSF, NSFC, and industrial companies. He has served as an editor for several journals, including IEEE TPDS, IEEE TETC, IEEE TGCN, IEEE Communications Magazine, and Wireless Networks. He has been actively participating in international conferences as general chair and TPC chair. He is a senior member of IEEE, a senior member of ACM, and an IEEE Communications Society Distinguished Lecturer.