Experience-Driven Computational Resource Allocation of Federated Learning by Deep Reinforcement Learning

Yufeng Zhan
Department of Computing
The Hong Kong Polytechnic University
Hong Kong, China
zhanyf1989@gmail.com

Peng Li
School of Computer Science and Engineering
The University of Aizu
Aizuwakamatsu, Japan
pengli@u-aizu.ac.jp

Song Guo
Department of Computing
The Hong Kong Polytechnic University
Hong Kong, China
song.guo@polyu.edu.hk

***Abstract*—Federated learning is promising in enabling large-scale machine learning by massive mobile devices without exposing the raw data of users with strong privacy concerns. Existing work of federated learning struggles for accelerating the learning process, but ignores the energy efficiency that is critical for resource-constrained mobile devices. In this paper, we propose to improve the energy efficiency of federated learning by lowering CPU-cycle frequency of mobile devices who are faster in the training group. Since all the devices are synchronized by iterations, the federated learning speed is preserved as long as they complete the training before the slowest device in each iteration. Based on this idea, we formulate an optimization problem aiming to minimize the total system cost that is defined as a weighted sum of training time and energy consumption. Due to the hardness of nonlinear constraints and unawareness of network quality, we design an experience-driven algorithm based on the Deep Reinforcement Learning (DRL), which can converge to the near-optimal solution without knowledge of network quality. Experiments on a small-scale testbed and large-scale simulations are conducted to evaluate our proposed algorithm. The results show that it outperforms the start-of-the-art by 40% at most.**

***Index Terms*—federated learning, experience-driven, deep reinforcement learning**

I. Introduction

Deep learning has demonstrated significant advantages of using big data to train models for object recognition, classification, and prediction of future events. In many applications, training data are generated by resource-constrained mobile devices (e.g., tablets and smartphones) or Internet-of-Things (IoT) devices. It would be impractical to upload all data to the cloud for centralized model training due to limited network bandwidth. Besides, these devices are owned by individuals that are usually unwilling to share their data due to privacy concerns.

Federated learning has been proposed to let distributed mobile devices at the network edge train machine learning models collaboratively without exposing their own data. Similar to the paradigm of distributed machine learning in data centers, federated learning works in iterations. Within each iteration, mobile devices train local models using their own data, respectively, and then upload the local models, instead of raw data, to a logically centralized parameter server that synthesizes a global model. Since an inception by Google [1], [2], federated learning has attracted great attentions from both academia [3]–[5] and industry [6], [7]. Wang et al. [3] have proposed the adaptive control of the number of training iterations under the resource-constrained edge environment. Tran et al. [4] have studied the federated learning over wireless networks by formulating an optimization problem that captures tradeoff between communication and computation cost.

Although federated learning has shown great promises, there are several critical open challenges unsolved by existing work. First, the major focus of existing work [1], [2], [8] is designing fast learning algorithms with provable convergence speed, but energy efficiency, a critical issue for mobile devices powered by batteries, of federated learning has seldom been studied. Mobile devices may hesitate to join federated learning if the participation incurs quick battery exhaustion. Therefore, it is critical to make a good tradeoff between learning time and energy efficiency, which motivates our work in this paper.

Second, mobile devices are heterogeneous in computation and communication capabilities due to different hardware, software, and network conditions. On the other hand, the iterative scheme of federated learning imposes synchronization among training nodes in each iteration, which opens an unexplored optimization space of computational resource allocation on mobile devices to improve energy efficiency without slowing the training process. For example, if a mobile device has a good network connection, it is unnecessary to run the training program at full speed using all its computational resources. It can lower the CPU-cycle frequency to a certain level for energy saving, as long as its local model uploading is no later than the slowest device in the group.

Finally, most existing work [3], [4] relies on an unrealistic assumption that mobile devices involved in federated learning have stable network connection. However, their network quality changes due to mobility or environmental factors in practice, even within a single training iteration. Without the knowledge of network quality, it would hardly choose the

optimal configuration leading to the minimum completion time of each iteration, which consists of model training time and model uploading time. The unpredictable network quality increases the difficulty of algorithm design.

In this paper, we address the above challenges by formulating the federated learning under dynamic network environment as a stochastic optimal control problem and solve it using a learning-based algorithm. Specifically, at the beginning of each iteration, we observe the learning speeds of mobile devices in the previous iterations. According to the observation, we then decide how many computational resources will be allocated for training on each mobile device based on Deep Reinforcement Learning (DRL). Our objective is to minimize the system cost that is defined as the weighted sum of learning time and total energy consumption. The proposed algorithm requires no knowledge of network quality. The main contributions of this paper are summarized as follows:

- We study a novel mobile computational resource allocation problem for federated learning, which jointly considers the learning time and energy efficiency by exploiting the heterogeneity of mobile devices and their network connections.
- We propose an experience-driven algorithm to solve the problem based on DRL, which can effectively learn the best suitable strategies of mobile computational resource allocation.
- We conduct both experiments on a small-scale testbed and large-scale simulations to evaluate the performance of the proposed algorithm. We compare our approach with the state-of-the-art solutions and the results further demonstrate the superiority of our approach.

The rest of this paper is organized as follows. The motivation of this paper is given in Section II. Section III presents the system model and problem formulation of this paper. We then present the design and implementation details of the proposed DRL-based solution in Section IV. Section V comprehensively evaluates the performance of the proposed approach through experiments and simulations. We discuss related work in Section VI and conclude the paper in Section VII.

II. Motivation

Federated learning [1], [2], [7] is a distributed machine learning approach that enables model training on large amounts of decentralized data held by mobile devices. As an instance of the general approach of "bringing the code to the data, instead of the data to the code", it addresses the fundamental problems of privacy, ownership, and locality of data [6]. The general description of federated learning has been presented in [1], and its theories have been studied in [8]–[10].

The participants in the federated learning consist of a number of mobile devices and the parameter server that usually resides in the cloud. The workflow during each iteration is shown in Figure 1. First, mobile devices download the current global model parameters maintained by the parameter server. Then, each device trains the downloaded model using its local

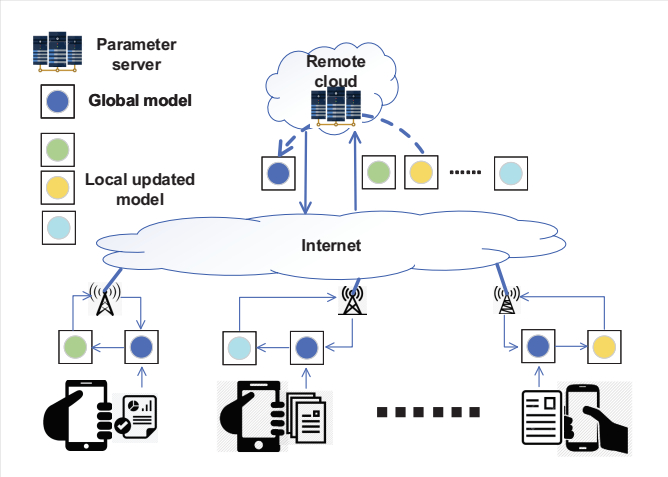


Fig. 1: Illustration of federated learning.

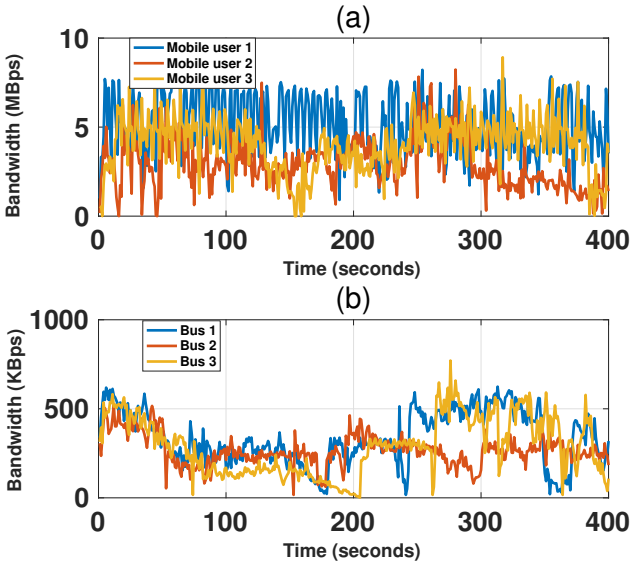


Fig. 2: The dynamics of network bandwidth.

data. The trained models are uploaded to the parameter server that finally synthesizes an updated global model.

Unstable network quality. Most existing work of federated learning assumes stable network connections between parameter server and mobile devices, which unfortunately is impractical because network quality consistently changes. Some evidences are shown in Figure 2. Hooft et al. [11] have conducted bandwidth measurements in 4G networks in and around the city of Ghent, Belgium, during the period of 2015-12-16 to 2016-02-04. We randomly choose 3 mobile devices from their dataset and plot the transmission speeds within 400 seconds in Figure 2(a), where we can observe that network speed changes from less than 1MBps to 9MBps in such a short time. In Figure 2(b), we show the network speeds from the HSDPA dataset [12] that measures the network quality of moving buses in Norway. We have a similar observation that network quality fluctuates between [0, 800KBps].

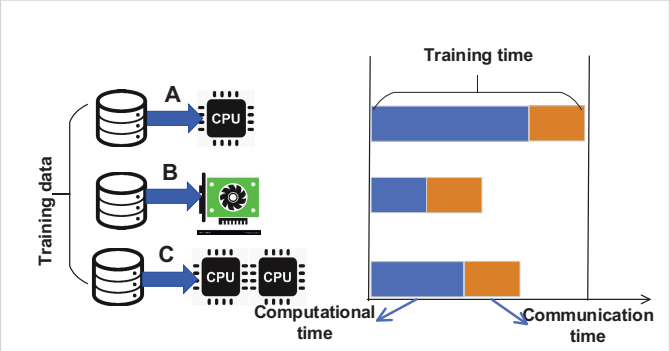


Fig. 3: Illustration of one training iteration in federated learning.

Trading idle time for power saving. We then study how network quality affects the learning speed and energy efficiency of federated learning. As shown in Figure 3, the training time of each mobile device consists of two parts, the computational time of training the model using local data, and the communication time of uploading the trained model to the parameter server. Since the parameter server needs to collect models from all devices to generate an updated global model, the training time of every iteration is determined by the slowest device. As an example shown in Figure 3, device A is the slowest one, but the other two devices are faster in completing the model uploading, leading to unnecessary idle time. This observation motivates us to slow down the computation on devices B and C to reduce CPU/GPU energy consumption, as long as their training time is no later than the slowest one.

The above idea is promising, but facing several challenges. First, as shown in Figure 2, the network quality changes and cannot be accurately predicted in practice. Since it is difficult to estimate the communication time of devices at the beginning of each iteration, we cannot decide how fast they should run training programs. The fast computation may lead to idle time without saving energy. On the other hand, energy efficiency can be improved by slow computation, but leading to longer training time. Second, the problem of optimizing computational resource allocation to achieve short learning time and low energy efficiency is hard, which we will show in Section III. It is difficult to design efficient heuristic algorithms with guaranteed performance.

Instead of struggling with network quality prediction and optimization-based algorithm design, we turn to machine learning techniques for solving federated learning problems. Specifically, we adopt the deep reinforcement learning to decide how fast mobile devices should train their local models.

III. Problem Description

In this section, we first present the system model considered in this paper and then formulate the problem of optimizing learning time and energy efficiency, while guaranteeing learning quality. Some main notations are listed in Table I.

TABLE I: Notations

Variable	Definition
τ	The number of local training times in each iteration
α_i	the effective capacitance coefficient of mobile device i's computing chipset
δ_i^k	the CPU-cycle frequency of mobile device i in the k-th iteration
δ_i^{max}	the CPU-cycle frequency upper bound of mobile device i
c_i	the number of CPU cycles for mobile device i to execute one sample of training data
$\mathcal{D}_i$	mobile device i's local dataset
ξ	the size of deep learning parameters
B_i^k	communication bandwidth of mobile device i in k-th iteration
$t_{i,cmp}^k$	computational time of mobile device i in k-th iteration
$t_{i,com}^k$	communication time of mobile device i in k-th iteration
t^k	the start time of the k-th iteration
K	total number of training iterations
E_i^k	the units of energy consumed by mobile device i in k-th iteration
T_i^k	the units of time consumed by mobile device i in k-th iteration
T^k	the training time of k-th iteration

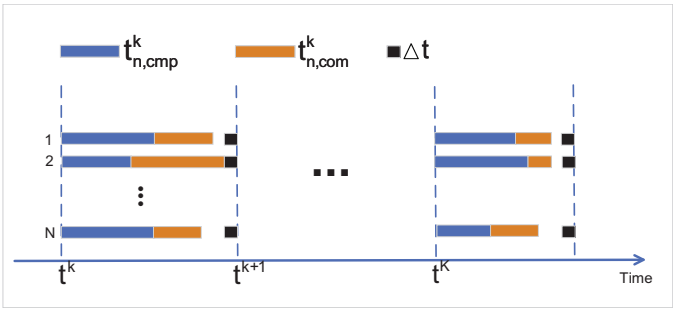


Fig. 4: Illustration of federated learning process.

A. System Model

We consider a federated learning instance consisting of a set $\mathcal{N} = \{1, 2, \cdots, N\}$ of mobile devices, which are connected to a centralized parameter server residing in the edge computing site or cloud. Each device $i \in \mathcal{N}$ maintains a dataset $\mathcal{D}_i$, whose size is denoted by D_i. Each data sample in $\mathcal{D}_i$ can be represented by $(\boldsymbol{x}_i^j, y_i^j)$, where the vector $\boldsymbol{x}_i^j$ is the input of the training model and the scalar y_i^j is the desired output.

The learning model. The learning process consists of multiple iterations, as shown in Figure 4. In each iteration, mobile devices first download the deep learning model, denoted by ω, from the parameter server. Since the downlink bandwidth is much larger than that of uplink, the model downloading time is negligible as compared with uploading time [4]. Mobile devices then train the model ω using gradient-descent algorithms fed by their local data, respectively. In order to reduce the communication overhead, each device trains the model by τ times. Since all data samples at a mobile device are with the same size (i.e., image resolution), we let c_i denote the number of CPU cycles used for training a single data sample at device i. It can be measured via profiling execution of training programs [13]. The computational time of mobile device i in

the k-th iteration can be calculated by:

$$t_{i,cmp}^k = \frac{\tau c_i D_i}{\delta_i^k}, \forall i \in \mathcal{N}, 1 \leq k \leq K, \tag{1}$$

where δ_i^k denotes the CPU-cycle frequency of device i in the k-th training iterations.

After the training computation, each device i uploads its trained model to the parameter sever, and the transmission time can be calculated by

$$t_{i,com}^k = \frac{\xi}{B_i^k}, \forall i \in \mathcal{N}, 1 \leq k \leq K, \tag{2}$$

where ξ is the model size and B_i^k is the average transmission speed of device i during the k-th iteration. Since we consider a continuous time model, B_i^k can be expressed as

$$B_i^k = \frac{1}{t^{k+1} - \Delta t_i^k - t_{i,cmp}^k - t^k} \int_{t^k + t_{i,cmp}^k}^{t^{k+1} - \Delta t_i^k} B_t dt, \tag{3}$$

where t^{k+1} is the starting time of the k-th iteration and Δt_i^k is the idle time of device i in the k-th iteration. Hence, the total training time for device i in the k-th iteration is

$$T_i^k = t_{i,cmp}^k + t_{i,com}^k, \forall i \in \mathcal{N}. \tag{4}$$

As shown in Figure 4, we adopt a synchronized learning model, i.e., the next iteration starts only when the parameter server receives the updates from all devices and creates a new global model, which has been shown to be more efficient than asynchronous models [14]. Therefore, the learning time of the k-th iteration can be calculated by:

$$T^k = \max T_i^k, \forall i \in \mathcal{N}. \tag{5}$$

The energy model. According to a widely adopted energy model [15], the energy consumption of device i in the k-th iteration can be calculated by

$$E_i^k = \alpha_i c_i D_i (\delta_i^k)^2 + e_i t_{i,com}^k, \tag{6}$$

where α_i is the effective capacitance coefficient of device i's computing chipset and e_i is the unit energy consumed for transmitting the model, which is independent with δ_i^k.

The loss function. We use loss function to evaluate the training quality. The loss function on training samples of device i is

$$F_i(\boldsymbol{\omega}) = \frac{1}{|\mathcal{D}_i|} \Sigma_{j \in \mathcal{D}_i} f_j(\boldsymbol{\omega}), \tag{7}$$

where $f_j(\boldsymbol{\omega})$ is the loss function of one sample j.

The global loss function over all the distributed datasets is defined by

$$F(\boldsymbol{\omega}) = \frac{\Sigma_{j \in \bigcup_n \mathcal{D}_n} f_j(\boldsymbol{\omega})}{|\bigcup_n \mathcal{D}_n|} = \frac{\sum_{n=1}^N D_n F_n(\boldsymbol{\omega})}{\sum_{n=1}^N D_n}. \tag{8}$$

B. Problem Formulation

The learning time and energy efficiency are contradictory objectives in the optimization problem. If mobile devices run training programs at their full CPU speeds, the total learning time can be reduced, but leading to high energy consumption. On the other hand, if they lower CPU-cycle frequency for power saving, the learning time is prolonged. In our paper, we study to minimize a system cost function, which makes a tradeoff between these two metrics, while guaranteeing a certain level of learning quality represented by the loss function. The problem formulation is as follows.

$$\min \sum_{k=1}^K (T^k + \lambda \sum_{i=1}^N E_i^k) \tag{9}$$

$$F(\boldsymbol{\omega}) \leq \epsilon, \tag{10}$$

$$t^{k+1} = t^k + T^k, \forall 1 \leq k \leq K, \tag{11}$$

$$0 \leq \delta_i^k \leq \delta_i^{max}, \forall 1 \leq i \leq N, 1 \leq k \leq K. \tag{12}$$

$$(1) - (8).$$

As different federated learning tasks have different preferences on learning time and energy efficiency, we define a non-negative parameter λ to adjust the preference in the objective function. A large λ indicates that the parameter server is not particularly concerned about time. On the other hand, more efforts are made to achieve fast federated learning model training under a small λ. In general, this definition of objective is quite general as it allows the parameter server to model varying federated learning preferences on different contributing factors. To guarantee the training quality, the global loss function should be less than ϵ, as shown in constraint (10). The $(k+1)$-th iteration begins only after the slowest edge node completes its training in the k-th iteration, which is represented by constraint (11). The constraint (12) indicates that the CPU-cycle frequency should be no greater than δ_n^{max}.

Solving the above problem is quite difficult due to nonlinear constraints and unawareness of future network quality. Therefore, we propose a data-driven approach to solve the problem, which will be elaborated in the next section.

IV. Algorithm Design

In this section, we first give an overview about our algorithm design and then elaborate on the DRL model and training algorithm design.

A. Overview

We illustrate the proposed architecture in Figure 5, which contains two parts: federated learning system and DRL agent. The DRL agent is deployed in the parameter server. In this work, we design the DRL agent based on actor-critic method including an actor network and a critic network, which is the core of the federated learning system. At the beginning of each iteration, the DRL agent can determine the suitable CPU-cycle frequency for each mobile device.

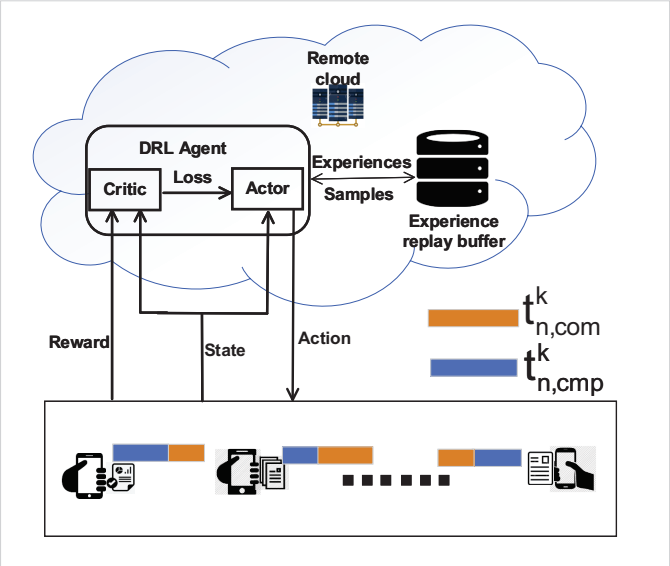


Fig. 5: The architecture of the proposed DRL-based control framework.

Different from many existing edge resource allocation approaches using predefined rules or model-based heuristics, DRL strives to learn a general action decision from past experiences based on the current state and the given reward. Specially, as illustrated in Figure 5, a DRL agent interacts with the federated learning system that defines the rules, restrictions and reward mechanism. At the beginning of k-th iteration, the agent observes a state of the federated learning system and determines an action accordingly. When the chosen action is done, the federated learning system transmits to the next state and the agent receives a reward. If the agent continues this process, it will get accumulated rewards after every action is done. The details of function design of the DRL agent, such as state, action and reward, will be presented in the following subsections. The objective of the DRL agent is to find the best policy mapping a state to an action that maximizes the expected discounted accumulated reward.

B. DRL Model Design

We leverage some advanced RL techniques, including actor-critic method [16]–[18] and the proximal policy optimization (PPO) algorithm [19], for solving the CPU-cycle frequency control problem in federated learning system. We introduce the design details as follows.

1) State Space: We consider a practical scenario with dynamic network bandwidth, but the network conditions are reasonably stable on short timescales and usually do not change drastically during a short time slot h (tens of seconds) [20], [21]. Recall the problem formulation in Section III, network bandwidth is the main factor affecting system cost. Since future network bandwidth is related to historical bandwidth information [20], [22], we set the state space in DRL formulation consists of several past bandwidth historical information which can be defined as $\boldsymbol{s}_k = (\boldsymbol{B}_1^k, \boldsymbol{B}_2^k, \cdots, \boldsymbol{B}_N^k)$. Here, $\boldsymbol{B}_i^k = (B_i(\lfloor t^k/h \rfloor), B_i(\lfloor t^k/h \rfloor - 1), \cdots, B_i(\lfloor t^k/h \rfloor - H))$ and $B_i(j)$ is the bandwidth of mobile device i in the j-th time slot.

2) Action Space: An action in the k-th round is defined as $\boldsymbol{a}_k =< \delta_i^k >, \forall i \in \mathcal{N}$, where $\delta_i^k \in (0, \delta_i^{max}]$. Given the continuous values of the CPU-cycle frequency δ_i^k, there are infinite $\{state, action\}$ pairs so that we can not store them in a tabular form and solve the problem using value-based methods, e.g., Q-learning and SARSA. To address this issue, we use a neural network to represent the policy π, where the adjustable parameters of the neural network is referred to as the policy parameters $\boldsymbol{\theta}_a$. Then, the policy can be represented as $\pi(\boldsymbol{a}_k|\boldsymbol{s}_k;\boldsymbol{\theta}_a) \in [0,1]$, indicating the probability of taking the action $\boldsymbol{a}_k$ at current state $\boldsymbol{s}_k$.

3) Reward: When applying an action $\boldsymbol{a}_k$ under the state $\boldsymbol{s}_k$, the DRL agent receives a reward r_k from the federated learning system in the k-th iteration. Considering the optimization objective (9), we craft the reward to achieve the mininum system cost. Specifically, in the k-th iteration, the mobile device trains the model using its own data with the specified CPU-cycle frequency δ_i^k determined by DRL agent. With time $t_{i,cmp}^k$, the mobile user i completes the federated learning model updates, and starts to upload the new parameters to the parameter server with time $t_{i,com}^k$. After all the mobile devices completing the new parameters upload, the DRL agent can obtain the system cost in current iteration. We define the reward r_k in the round k as

$$r_k = -T^k - \lambda \sum_{i=1}^{N} E_i^k, \forall 1 \leq i \leq N, 1 \leq k \leq K. \qquad (13)$$

Note that the design of state, action and reward is critical to the success of DRL method. In the federated learning system, the actions and rewards are straightforward. However, there are different ways of defining states because federated learning system is a very complex system which includes various information. We choose a simple and clean way in our design, and the experimental results in Section V show that our design can achieve superior performance.

C. DRL Training Methodology

We train the DRL agent by using the actor-critic method [16]–[18], which well matches our context and has been successfully applied in many other fields. There are lots of policy optimization approach such as DPG [23], A2C [24], TRPO [25], PPO, and etc. PPO strikes a balance between ease of implementation, sample complexity, and ease of tuning, trying to compute an update that minimizes the objective function while ensuring the deviation from the previous policy is relatively small. Hence, in this work, the policy optimization process of DRL agent employs the PPO algorithm.

The DRL agent maintains an experience replay buffer $\boldsymbol{\mathcal{D}}$, a policy $\pi(\boldsymbol{a}_k|\boldsymbol{s}_k;\boldsymbol{\theta}_a)$ (the actor network) and an estimate of the value function $V(\boldsymbol{s}_k;\boldsymbol{\theta}_v)$ (the critic network), where $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_v$ are the parameters of the actor network and critic network, respectively. Since the parameter server can access mobile devices' information, hence we can train the DRL agent in an

offline manner. Algorithm 1 shows the detailed offline training procedure of the DRL agent.

The DRL agent training procedure starts with randomly initialize the parameters of the actor network and critic network (Line 1). Before the DRL agent training, we should load the real-world network dataset and mobile devices' information, this will construct a simulated training environment of federated learning system. In order to train the DRL agent efficiently, we need to use another policy $\boldsymbol{\theta}_a^{old}$ (Line 2) to sample the federated learning environment. By this way, the DRL agent can repeatedly use the experience sampled by $\boldsymbol{\theta}_a^{old}$ multiple times. The federated learning system randomly selects a start time t^1 (Line 6). Then, the DRL agent constructs the initial state by each mobile device's past bandwidth historical information (Line 6-10). Then the DRL agent starts to execute CPU-cycle frequency control. At the beginning of the k-th iteration in federated learning, the DRL agent feeds the state $\boldsymbol{s}_k$ into the policy network $\pi(\cdot|\boldsymbol{s}_k;\boldsymbol{\theta}_a^{old})$ and derive action $\boldsymbol{a}_k$ (Line 12). After the mobile devices receiving the action from the DRL agent, they train the deep learning model with the specified CPU-cycle frequency determined by $\boldsymbol{a}_k$ (Line 13). The k-th iteration ends with the parameter server receiving all the updates from the mobile devices. Then the DRL agent can calculate the reward r_k obtained in the k-th iteration (Line 14), and the federated learning system moves to the next state $\boldsymbol{s}_{k+1}$ (Line 15). At the same time, the experience in the k-th iteration is stored in the experience replay buffer $\boldsymbol{\mathcal{D}}$ (Line 16). When the experience replay buffer is full (Line 16), we can update the DRL agent with the experience in $\boldsymbol{\mathcal{D}}$, where the actor network is updated by the PPO approach. After the DRL agent learning the information from the experience in $\boldsymbol{\mathcal{D}}$ with M times, the newly parameters of actor network $\boldsymbol{\theta}_a$ are assigned to the policy $\boldsymbol{\theta}_a^{old}$ to do the next sampling (Line 22). Meanwhile, the experience replay buffer is emptied (Line 23).

Algorithm 1 The offline DRL agent training procedure

1: Randomly initialize actor network $\pi(\cdot)$ and critic network $V(\cdot)$ with weights $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_v$, respectively;
2: Load the real-world network dataset;
3: Initialize experience replay buffer $\boldsymbol{\mathcal{D}}$ and mobile devices' information such as c_i, D_i, α_i and etc.;
4: $\boldsymbol{\theta}_a^{old} \leftarrow \boldsymbol{\theta}_a$;
5: **for** $episode = 1, 2, \cdots$, **do**
6: Randomly select a federated learning start time t^1;
7: **for** $i \in \mathcal{N}$ **do**
8: $\boldsymbol{B}_i^1 = (B_i(\lfloor t^1/h \rfloor), B_i(\lfloor t^1/h \rfloor - 1), \cdots, B_i(\lfloor t^1/h \rfloor - H))$;
9: **end for**
10: $\boldsymbol{s}_1 = (\boldsymbol{B}_1^1, \boldsymbol{B}_2^1, \cdots, \boldsymbol{B}_N^1)$;
11: **for** $k = 1, 2, \cdots, K$ **do**
12: Derive action $\boldsymbol{a}_k$ by feeding $\boldsymbol{s}_k$ into the policy network $\pi(\cdot|\boldsymbol{s}_k, \boldsymbol{\theta}_a^{old})$;
13: Mobile devices train the model with the specified CPU-cycle frequency determined by action $\boldsymbol{a}_k$;
14: Calculate the reward r_k by (13);
15: Update the state of federated learning from $\boldsymbol{s}_k$ to $\boldsymbol{s}_{k+1}$;
16: Store transition sample $(\boldsymbol{s}_k, \boldsymbol{a}_k, r_k, \boldsymbol{s}_{k+1})$ into $\boldsymbol{\mathcal{D}}$;
17: **if** $k\%|\boldsymbol{\mathcal{D}}| == 0$ **then**
18: **for** $m = 1, 2, \cdots, M$ **do**
19: Update $\boldsymbol{\theta}_a$ using PPO;
20: Update the critic network $V(\cdot)$ by minimizing the loss function: $\min_{\boldsymbol{\theta}_v} \frac{1}{|\boldsymbol{\mathcal{D}}|} \sum_{j=1}^{|\boldsymbol{\mathcal{D}}|} [r_j + \gamma V(\boldsymbol{s}_{j+1}; \boldsymbol{\theta}_v) - V(\boldsymbol{s}_j; \boldsymbol{\theta}_v)]^2$;
21: **end for**
22: $\boldsymbol{\theta}_a^{old} \leftarrow \boldsymbol{\theta}_a$;
23: Clear the experience replay buffer $\boldsymbol{\mathcal{D}}$
24: **end if**
25: **end for**
26: **end for**

V. Performance Evaluation

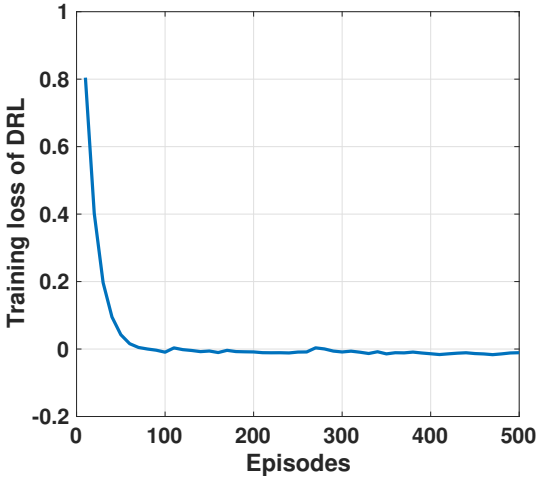
In this section, we compare our proposed algorithm with the state-of-the-art approaches via trace-driven experiments. We first describe experimental settings, followed by experimental results and analysis.

A. Experimental Settings

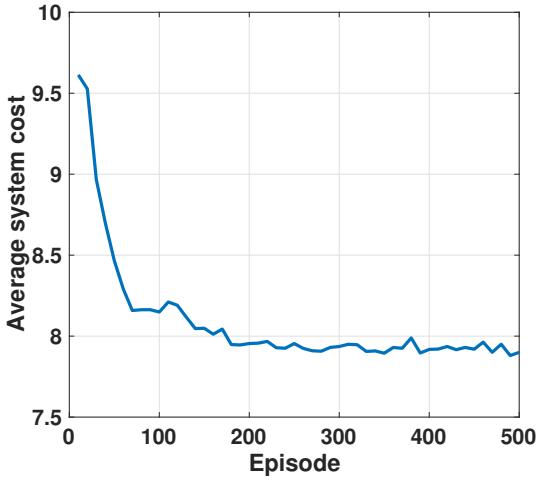
We use real-world traces of 4G/LTE networks [26] to construct an evaluation environment. The trace dataset contains bandwidth measurements of 4G networks along several routes in and around the city of Ghent, Belgium, during the period of 2015-12-16 to 2016-02-04. The data is collected by Huawei P8 Lite smartphones running in 6 scenarios: walking, bicycles, buses, trams, trains, and cars. We randomly select three walking datasets in our experiments.

Similar with the settings in [4], we set the size of training data held by mobile device as a uniform distribution within 50-100 MB. The number of CPU cycles used for training a single data sample, which is denoted by c_n, is uniformly distributed within 10-30 cycles/bit. The maximum CPU-cycle frequency δ_n^{max} is uniformly distributed within 1.0-2.0 GHz. We compare our proposed algorithm with two state-of-the-art approaches.

- Heuristic [3]: At the beginning of each training iteration of federated learning, since the last iteration is just ended, the parameter server could know all the mobile devices' bandwidth information. Hence, the parameter server can determine the mobile device's CPU-cycle frequency in current iteration with the bandwidth in the last iteration.
- Static [4]: different from heuristic, in this work, the authors assume that the network is static, and determine the optimal CPU-cycle frequency at the beginning of federated learning. In order to implement this approach, we randomly select some bandwidth data from the dataset, and determine the CPU-cycle frequency for each mobile device according to the average value of these bandwidth data. Then, in each federated learning training iteration, the mobile devices will use the consistent CPU-cycle



(a) Training loss.



(b) Average system cost.

Fig. 6: Training convergence of DRL agent.

frequency directly.

B. Evaluation Results

As illustrated in Section 4.3, our proposed algorithm contains two stages, offline training and and online reasoning. We evaluate the performance of two stages respectively and show the results as follows.

1) Offline DRL Training: We first study the training loss and system cost in a testbed consisting of $N = 3$ mobile devices. As shown in Figure 6(a), the training loss decreases quickly in the earlier stages of DRL training process. This is because the DRL agent has no information of the federated learning environment which causes a large loss value. Thanks to the deep neural network and PPO, the loss can be rapidly minimized. After less than 200 episodes, the training loss becomes stabilized, which means that the DRL agent learns to adapt the federated learning environment. In Figure 6(b), we can observe that the average system cost decreases over time. That is because the DRL agent is learning a better policy that can well control the CPU-cycle frequency. When reaching around 200 episodes, the system cost starts to saturate with slight fluctuations.

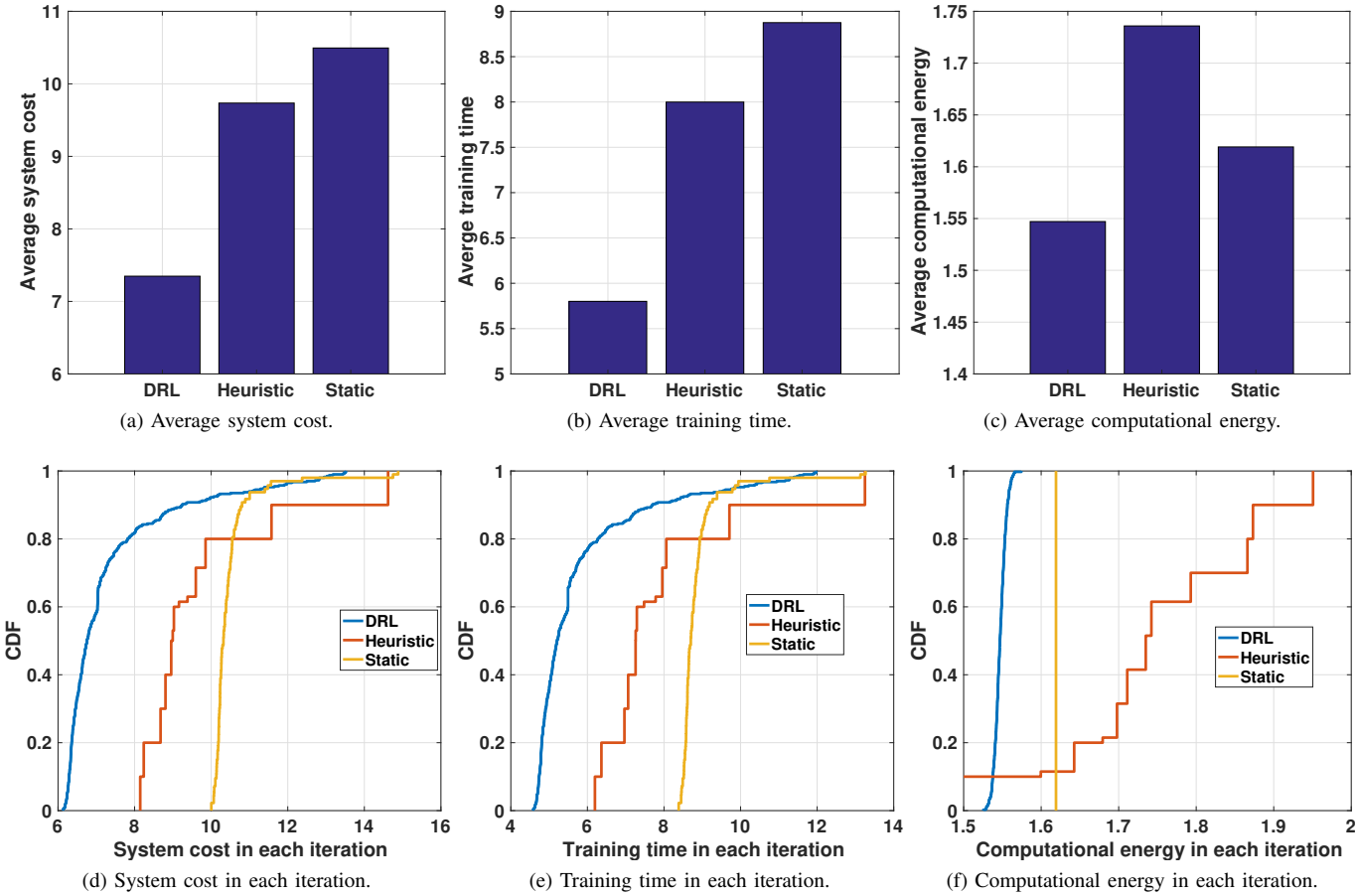
2) Online DRL Reasoning: After adequate offline training, the DRL model is saved for reasoning. During reasoning, we only use the trained actor network to generate its action $\boldsymbol{a}_k$, given its own state $\boldsymbol{s}_k$. Then, the parameter server receives a reward r_k, and changes its state to $\boldsymbol{s}_{k+1}$. The experimental results after 400 iterations are shown in Figure 7.

In Figure 7(a), we can see that DRL-based approach achieves the best performance. With DRL-based approach, the average system cost is 7.25, while the average system cost in heuristic and static are 9.74 and 10.5, respectively. The average system cost of the two state-of-the-art approaches is 35% higher than DRL-based approach. In order to see the system cost clearly, we show the cumulative distribution function (CDF) of the system cost in each iteration in Figure 7(d), where we can see that over 80% of the system cost in DRL-based approach is less than 8. However, the system cost in each training iteration is more than 8 under heuristic and static approaches.

Figure 7(b) shows the average training time of federated learning in each iteration. We can observe that DRL-based approach makes the federated learning training process very fast. For example, the heuristic approach is 38% slower than DRL-based approach. Figure 7(e) shows the CDF of the training time in each iteration. Nearly 80% of the training time in each iteration is less than 6 by using DRL-based approach. However, in heuristic and static approaches, the training time in each iteration is over 6.

As shown in Figure 7(c), the DRL-based approach also consumes the minimum computational energy as compared with the other two approaches. In Figure 7(f), energy consumption in each training iteration ranges from 1.5 to 1.6 for DRL. However, in heuristic, over 80% of the energy consumption is more than 1.7 in each training iteration. In static, the mobile device implements the computation with the same CPU-cycle frequency. Therefore, in each training iteration, federated learning costs computational energy with 1.62. Figure 7 verifies the finding in the motivation that even though mobile device invests more computing power, it can not necessarily accelerate the convergence rate of federated learning.

In order to evaluate the scalability of the proposed DRL-based approach. We conduct simulations with 50 mobile devices. Since the dataset we used in this paper does not have enough data, we randomly select five walking datasets and let each mobile device randomly select one dataset. In this group of simulation, we set $\lambda = 0.1$, and all the other parameters are the same as in the testbed experiment. As shown in Figure 8, with more mobile devices participating in the federated learning, our DRL-based approach can also obtain the best performance as compared with the state-of-the-art. For example, as shown in Figure 8, the system cost of each iteration in DRL is almost less than 12. However, in heuristic and static, the system cost in each iteration is larger than 14 and 16, respectively. The average system cost in DRL



(a) Average system cost. (b) Average training time. (c) Average computational energy.

(d) System cost in each iteration. (e) Training time in each iteration. (f) Computational energy in each iteration.

Fig. 7: Performance of federated learning.

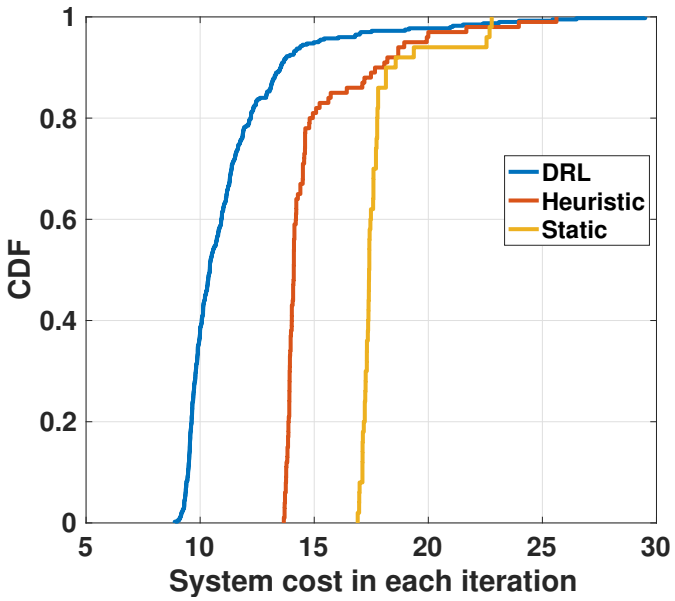


Fig. 8: The system cost in each iteration with 50 mobile devices.

is 11.2, while in heuristic is 14.3 and in static is 17.3.

VI. Related Works

Exiting works on edge computing mainly focus on generic applications, where solutions have been proposed for application offloading [27]–[30], workload scheduling [31]–[33], and service migration triggered by user mobility [34]–[36]. However, few works address the relationship among communication, computation, and training accuracy for machine learning applications, which is important for optimizing the performance of machine learning tasks.

A. Federated Learning

As a natural extension of distributed learning, federated learning moves the training computation from centralized cloud to devices at the 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 [7] and later has attracted great research attentions. Since training computation is distributed among edge nodes, communication becomes the main bottleneck. In [1], Mcmahan et al. have presented a practical model for the federated learning based on model averaging and conducted an extensive empirical evaluation. Wang et al. [3] have studied how to optimize the number of local iterations in each global iteration and the number

of total iterations of federated learning in a resource constrained mobile edge computing environment. Tran et al. [4] have studied the federated learning over wireless networks by formulating on optimization problem that capture tradeoff between communication and computation cost. They propose algorithms to solve the problem by decompose and transform the original problems into three convex sub-problems. Li et al. [37] have designed FedProx to address the heterogeneity of federated networks, which allows for more robust convergence than traditional methods across a suit of federated datasets. Considering participants with heterogenous resource, Nishio et al. [38] have proposed a participant selection scheme for federated learning.

Security and privacy issues are also the main concerns in federated learning. Bhowmich et al. [39] have analyzed local privacy protection in federated learning, which is significant for the practical adoption of private procedure. A testing methodology for quantitatively assessing the risk in federated learning has been proposed in [40]. In [41], Wang et al. have given the first attempt to explore user-level privacy leakage against the federated learning by the attack from a malicious server. A Byzantine gradient descent method has been proposed in [5] to defense the adversarial attacks. Bonawitz et al. [42] have proposed a communication-efficient and failure-robust protocol for secure aggregation of high-dimensional data.

B. Deep Reinforcement Learning

In recent years, DRL has shown amazing potentials in many applications [43]. Minh et al. [43] first used Deep Q-Network to learn policies from sensor input for decision making. In this work, the authors have proposed the experience replay and target network technologies to improve the stability and the performance. In order to reduce the observed over-estimations, Van Hasselt et al. [44] have designed the Double Deep Q-Network and achieved better performance on several games. In [45], Schaul et al. have developed a framework for prioritizing experience which can replay important transitions more frequently. There have been recent works on applying state-of-the-art policy gradient optimization techniques such as DPG [23] and PPO [19]. Towards this direction, DeepRM [46] leveraged DRL to solve the online multi-resource job placement problem and can achieve better performance as compared with the heuristic algorithms. In [47], Xu et al. first proposed to leverage emerging DRL for enabling model-free control in communication networks and presented a novel and highly effective DRL-based control framework, DRL-TE, for a fundamental networking problem. Li et al. [48] developed a novel model-free approach that can learn to well control a distributed stream data processing systems from its experience rather than accurate and mathematically solvable system models. Edics [49] leveraged neural network integrated convolutional neural network for feature extraction and then made decision under the guidance of multi-agent deep deterministic policy gradient method in a fully distributed mobile crowd sensing system. Chen et al. [50] developed a two-level DRL technique to solve complex online control problems for traffic optimizations in datacenters. In order to handle the complexity and scale of the scheduling problem on distributed compute clusters, Mao et al. [51] designed a scalable neural neural network architecture which combines the graph neural network and reinforcement learning together to make scheduling decisions. In [52], Shen et al. proposed the DeepAPP, which learned a model-free predictive neural network from historical APP usage data. Meanwhile, an online updating strategy was designed to adapt the predictive network to the time-varying APP usage behavior. Different from these aforementioned works, we consider the computational resource control problem in the federated learning. To the best our knowledge, this is the first to translate the federated learning to a control-theoretic problem and apply experience-driven DRL technique to cost-effectively accommodate the network dynamics.

VII. Conclusion

Our paper is motivated by recent debates surrounding the design of efficient federated learning mechanism. As the mobile devices with the high speed may be dragged down by the slow speed mobile devices, blindly increasing the computational speed not only can not accelerate the federated learning convergence rate, but also will increase energy consumption. In this paper, we propose to improve energy efficiency of federated learning by carefully controlling the CPU-cycle frequency. Due to the hardness of the control problem and the unawareness of the network quality, it prompt us to use the learning methods to solve this problem. Therefore, we design the experience-driven method to solve the control problem based on DRL. We train the DRL agent based on the real-world network datasets. The final trace-driven experiments further demonstrate the superiority of DRL-based approach compared to the state-of-the-art solutions.

Acknowledgment

This research was financially supported by the General Research Fund of the Research Grants Council of Hong Kong (PolyU 152221/19E), the National Natural Science Foundation of China (Grant 61872310), JSPS Grants-in-Aid for Scientific Research grant number JP19K20258. Peng Li and Song Guo are the corresponding authors.

References

[1] 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.

[2] J. Konecný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.

[3] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. of IEEE INFOCOM*, 2018, pp. 63–71.

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

[5] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proc. of ACM SIGMETRICS*, vol. 1, no. 2, pp. 1–25, 2017.

[6] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *Proc. of SysML*, 2019, pp. 1–15.

[7] B. A. y. Arcas, G. Andrew, D. Bacon, and et al., "Federated learning: Collaborative machine learning without centralized training data," http://timmurphy.org/2009/07/22/line-spacing-in-latex-documents/, accessed April 6, 2017.

[8] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[9] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[10] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *Proc. of ICLR*, 2018.

[11] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Communications Letters*, vol. 20, no. 11, pp. 2177–2180, 2016.

[12] "Hsdpa dataset," http://home.ifi.uio.no/paalh/dataset/hsdpa-tcp-logs, accessed December 01, 2014.

[13] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. of USENIX HotCloud*, 2010, pp. 1–7.

[14] J. Chen, R. Monga, S. Bengio, and R. Józefowicz, "Revisiting distributed synchronous SGD," *CoRR*, vol. abs/1604.00981, 2016.

[15] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *VLSI Signal Processing*, vol. 13, no. 2-3, pp. 203–221, 1996.

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

[17] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

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

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

[20] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. of ACM SIGCOMM*, 2015, pp. 325–338.

[21] Y. Zhang and N. Duffield, "On the constancy of internet path properties," in *Proc. of ACM IMW*, 2001, pp. 197–211.

[22] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326–340, 2014.

[23] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. of ICML*, 2014, pp. 387–395.

[24] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc of ICML*, 2016, pp. 1928–1937.

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

[26] "4g/lte dataset," https://users.ugent.be/ jvdrhoof/dataset-4g/, accessed 2016.

[27] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[28] L. Tong and W. Gao, "Application-aware traffic scheduling for workload offloading in mobile clouds," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[29] Y. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Proc. of IEEE INFOCOM*, 2017, pp. 1–9.

[30] 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.

[31] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[32] H. Tan, Z. Han, X. Li, and F. C. M. Lau, "Online job dispatching and scheduling in edge-clouds," in *Proc. of IEEE INFOCOM*, 2017, pp. 1–9.

[33] P. Li, S. Guo, T. Miyazaki, X. Liao, H. Jin, A. Y. Zomaya, and K. Wang, "Traffic-aware geo-distributed big data analytics with predictable job completion time," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1785–1796, June 2017.

[34] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2017.

[35] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017.

[36] 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.

[37] T. Li, A. K. Sahu, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *CoRR*, vol. abs/1812.06127, 2018.

[38] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. of IEEE ICC*. IEEE, 2019, pp. 1–7.

[39] A. Bhowmick, J. Duchi, J. Freudiger, G. Kapoor, and R. Rogers, "Protection against reconstruction and its applications in private federated learning," *arXiv preprint arXiv:1812.00984*, 2018.

[40] N. Carlini, C. Liu, J. Kos, Ú. Erlingsson, and D. Song, "The secret sharer: Measuring unintended neural network memorization & extracting secrets," *arXiv preprint arXiv:1802.08232*, 2018.

[41] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. of IEEE INFOCOM*. IEEE, 2019, pp. 2512–2520.

[42] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. of ACM CCS*, 2017, pp. 1175–1191.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, and et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[44] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. of AAAI*, 2016, pp. 2094–2100.

[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. of ICLR*, 2016, pp. 1–21.

[46] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. of ACM HotNets*, 2016, pp. 50–56.

[47] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *Proc. of IEEE INFOCOM*, 2018, pp. 1871–1879.

[48] T. Li, Z. Xu, J. Tang, and Y. Wang, "Model-free control for distributed stream data processing using deep reinforcement learning," *Proceedings of the VLDB Endowment*, vol. 11, no. 6, pp. 705–718, 2018.

[49] 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.

[50] L. Chen, J. Lingys, K. Chen, and F. Liu, "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proc. of ACM SIGCOMM*, 2018, pp. 191–205.

[51] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proc. of ACM SIGCOMM*, 2019, pp. 270–288.

[52] Z. Shen, K. Yang, W. Du, X. Zhao, and J. Zou, "Deepapp: a deep reinforcement learning framework for mobile application usage prediction," in *Proc. of ACM SenSys*, 2019, pp. 153–165.