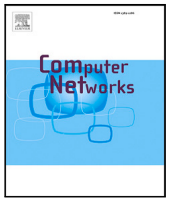
Computer Networks 221 (2023) 109517

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



An efficient federated learning framework for multi-channeled mobile edge network with layered gradient compression

Haizhou Du [a,*], Yijian Chen [a], Xiaojie Feng [a], Qiao Xiang [b], Haoyu Liu [b]

[a] School of Computer Science and Technology, University of Electric Power of Shanghai, Shanghai, China
[b] School of Information Science and Technology, Xiamen University, Xiamen, China

ARTICLE INFO

Keywords:
Federated learning
Mobile edge network
Layered gradient compression
Multi-channel communication
Deep reinforcement learning

ABSTRACT

A fundamental issue for federated learning (FL) is how to achieve efficient training performance under complex dynamic communication environments. This issue can be alleviated by the fact that modern edge devices usually can connect to the edge server via multiple communication channels (*e.g.*, 4G, LTE, and 5G) because multi-channel communication can increase the communication bandwidth and has lower communication costs and energy consumption than a single high-speed communication channel. However, if the communication data cannot be properly allocated to multiple channels in a complex dynamic communication network, multi-channel communication will still waste resources (e.g., bandwidth, battery life, and monetary cost). In this paper, we propose an efficient FL framework called, which consists of two parts, the layered gradient compression (LGC), and a learning-driven control algorithm. Specifically, with LGC, local gradients from a device are coded into several layers, and each layer is sent to the server along a different channel. The FL server aggregates the received layers of local gradients from devices to update the global model and sends the result back to the devices. Furthermore, we prove the convergence of LGC and formally define the problem of resource-efficient with LGC. We then propose a learning-driven algorithm for each device to dynamically adjust its local computation (*i.e.*, the number of local stochastic descent) and communication decisions (*i.e.*, the compression level of different layers and the layer-to-channel mapping) in each iteration. Results from extensive experiments show that significantly reduces the training time and improves the resource utilization (energy consumption and money cost) while achieving a similar test accuracy compared with well-known FL baselines.

1. Introduction

Federated learning has emerged as an efficient solution to analyze and process distributed data for data-driven tasks (*e.g.*, autonomous driving, virtual reality, image classification, etc.) in Mobile Edge Computing (MEC) [1–6]. By performing training tasks at edge devices (*e.g.*, mobile phones and tablets) and aggregating the learned parameters at edge servers, FL significantly reduces the network bandwidth usage of machine learning applications and protects the data privacy of edge devices [7].

However, practically deploying FL in edge networks still faces several difficulties. (1) The communication between devices and the server in dynamic edge networks may be frequently unavailable, congested, and expensive. (2) The resources (*e.g.*, bandwidth and battery life) are always limited in the MEC. These issues can be alleviated by the fact that modern edge devices usually can connect to the edge FL server via multiple communication channels (*e.g.*, 4G, LTE, and 5G) because multi-channel communication can increase the communication bandwidth and has lower communication costs and energy consumption than a single high-speed communication channel. However, if the communication data cannot be properly allocated to multiple channels in a complex dynamic communication network, multi-channel communication will still waste resources (*e.g.*, bandwidth, battery life, and monetary cost).

Several pioneering works have been proposed to improve resource utilization for efficient FL in edge networks [8–10]. However, these studies focus on reducing resource consumption, hindering performance boost in resource utilization and training efficiency. A recently promising solution is to incorporate gradient compression strategies into FL algorithms, which can considerably reduce the communication cost with little impact on learning outcomes [11,12]. However, these compression techniques are not tuned to the underlying communication channel and may not utilize the channel resources to the fullest.

To improve the resource utilization in MEC, we propose an FL framework called Rainbow, which consists of two parts, the layered

* Corresponding author.
E-mail address: duhaizhou@shiep.edu.cn (H. Du).

https://doi.org/10.1016/j.comnet.2022.109517
Received 13 May 2022; Received in revised form 4 December 2022; Accepted 7 December 2022
Available online 10 December 2022

gradient compression and a learning-driven control algorithm. This framework likes a beautiful Rainbow to build a multi-channel link between edge deceives and edge server for layered gradient transmitting with compression. Inspired by the layered coding techniques in video streaming, local gradients from a device are coded into several layers, and each layer can send to a server by a different channel, respectively. The server aggregates the received layers of local gradients from devices to update the global model and sends the results back to devices. We integrate gradient compression and multi-channel communication into FL to alleviate communication and energy bottlenecks. We prove the convergence of LGC in Rainbow and formally define the problem of resource-efficient Rainbow with LGC. To deploy Rainbow in dynamic networks and resource-constrained MEC systems, we then propose a learning-driven algorithm for each device to adaptively adjust its local computation (*i.e.*, the number of local stochastic descent) and communication decisions (*i.e.*, the compression level of different layers and the layer-to-channel mapping) in each iteration.

Our main contributions to this paper are as follows:

- To efficiently utilize the limited resources at edge devices in dynamic edge networks, we propose an efficient FL framework called Rainbow. To the best of our knowledge, we are the first to propose such a layered gradient compression FL framework.
- We provide a rigorous convergence guarantee for LGC from a theoretical perspective and formally define the problem of resource-efficient Rainbow with LGC.
- We also propose a learning-driven control algorithm for each device to adaptively adjust its local computation and communication decisions in each iteration, subject to dynamic edge network and resource constraints.
- We evaluate the performance of Rainbow. Results show that Rainbow significantly reduces the training time and improves resource utilization while achieving a similar accuracy compared with the baselines.

The rest of this paper is organized as follows. In Section 2, we describe the framework of Rainbow and define the problem of resource-efficient Rainbow with LGC. In Section 3, we prove the convergence of LGC in Rainbow. In Section 4, we describe the design and implementation details of the learning-driven control algorithm in Rainbow. We show the experimental results in Section 5, summarize the related works in Section 6, and conclude this work in Section 7.

2. Framework design

In this section, we first describe the main notations used in the paper. Then, we proposed Rainbow framework. Finally, we put forward the problem formulation of resource-efficient Rainbow with LGC.

2.1. Main notations

Table 1 summarizes the main notations used in this paper.

2.2. Framework overview

The framework of Rainbow follows the typical FL pattern and consists of two parts, an edge server and M edge devices. In Rainbow, M edge devices denoted by $\mathcal{M} = \{1, 2, \dots, m, \dots, M\}$ collaboratively train a learning model with an edge server by iterative computation and communication.

To alleviate the communication bottleneck, Rainbow compresses the local computed gradients before transmitting them and sends them through multiple channels. Fig. 1 gives an overview of Rainbow. In Rainbow, each device computes the local gradients (①), compresses the gradients by LGC operator (②) and sends encoded layers of the compressed gradients to the server through multiple channels (③). The server waits until the gradients from all the devices are received. It then aggregates them (④) and dispatches the results to all devices (⑤). Devices use them to update the local model. LGC considers different communication channel links between edge server and edge devices, including 3G, 4G and 5G. Multiple channels are indicated by different colors.

To compress the gradients, we consider the Top_k operator, a sparsification operator in distributed training. For any vector $\mathbf{x} \in \mathbb{R}^D$, $\text{Top}_k(\mathbf{x}) \in \mathbb{R}^D$ is equal to a D-length vector, which has at most k non-zero components whose indices correspond to the indices of the largest k components (in absolute value) of $\mathbf{x}$. And we extend Top_k to $\text{LGC}_\mathbf{k}$ operator for multiple communication channels. Before giving the definition of $\text{LGC}_\mathbf{k}$, we extend Top_k operator to $\text{Top}_{\alpha,\beta}$ $(1 \le \alpha < \beta \le D)$ operator to take the sparsified top-(α, β) gradients. Specifically, for a vector $\mathbf{x} \in \mathbb{R}^D$, $\text{Top}_{\alpha,\beta}(\mathbf{x}) \in \mathbb{R}^D$ and the ith $(i = 1, 2, \dots, D)$ element of $\text{Top}_{\alpha,\beta}(\mathbf{x})$ is defined as

$$\text{Top}_{\alpha,\beta}(\mathbf{x}_i) = \begin{cases} \mathbf{x}_i, & \text{if } thr_\alpha \ge |x_i| > thr_\beta, \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $\mathbf{x}_i$ is the ith element of $\mathbf{x}$ and thr_α is the αth largest absolute value of the elements in $\mathbf{x}$ and thr_β is the βth largest absolute value of the elements in $\mathbf{x}$.

Edge devices can usually connect with multiple communication channels. Considering a device with C channels connected to it, the traffic allocation among these channels is denoted by a vector $\mathbf{k} \in \mathbb{R}^c$. The device codes gradient elements into different layers with $\text{Top}_{\alpha,\beta}$ operator and gets $\{\text{Top}_{\sum_{i=1}^{c-1}\mathbf{k}_i, \sum_{i=1}^{c}\mathbf{k}_i}(\mathbf{x})\}_{c=1}^{C}$. Then each layer is sent to server through different channels. The server collects gradients from all the channels, decodes them and gets $\text{LGC}_\mathbf{k}(\mathbf{x})$. For a vector $\mathbf{x} \in \mathbb{R}^D$, $\text{LGC}_\mathbf{k}(\mathbf{x}) \in \mathbb{R}^d$ and the ith $(i = 1, 2, \dots, d)$ element of $\text{LGC}_\mathbf{k}(\mathbf{x})$ is defined as

$$\text{LGC}_\mathbf{k}(\mathbf{x}) = \sum_{c=1}^{C} \text{Top}_{\sum_{i=1}^{c}\mathbf{k}_i, \mathbf{k}_c}(\mathbf{x}). \tag{2}$$

Unlike previous studies requiring an identical number of local computations and a fixed compression level across all the participants, Rainbow uses an asynchronous operation where the devices synchronize with the master at arbitrary times. Rainbow also allows the participating devices to perform gradient sparsification with different compression coefficients. This helps to accommodate stragglers with poor channel conditions and thus mitigates the impacts of stale updates. By definition, we also allow devices to be equipped with different numbers and types of communication channels.

Since filtering the insignificant gradients will cause a biased estimation of the original gradients, which will damage the model test accuracy. Rainbow uses error accumulation to fix this problem. It was recently shown that using error accumulation (*i.e.*, storing the difference between the computed and compressed gradients and reinserting it at the next iteration) improves both convergence and generalization for compression schemes. This can enable biased gradient compression schemes to reach the target test accuracy.

To state the algorithm, we need the following definition from [13].

Definition 1 (*Gap*). Let $\mathcal{I} = \{t_0, t_1, \dots, t_k\}$, where $t_i < t_{i+1}$ for $i = 0, 1, \dots, k-1$. The gap of $\mathcal{I}$ is defined as $gap(\mathcal{I}) := \max_{i \in \{1,\dots,k\}} (t_i - t_{i-1})$, which is equal to the maximum difference between any two consecutive synchronization indices.

Let $\mathcal{I}_m \subseteq \mathcal{T} := \{1, \dots, T\}$ with $T \in \mathcal{I}_m$ denote a set of indices for which device $m \in \mathcal{M}$ communicates with the server. We assume that $\text{gap}(\mathcal{I}_m) \le H$ holds for every $m \in \mathcal{M}$, which means that there is a unified bound on the maximum delay in each device's update times. Every device $m \in \mathcal{M}$ maintains a local parameter vector $\widehat{\mathbf{w}}_m^{(t)}$ which is updated in each iteration t. If $t \in \mathcal{I}_m$, the error-compensated update $\mathbf{u}_m^{(t)}$ is sent to the server, and device updates its local accumulated error $\mathbf{e}_m^{(t)}$.

Table 1
Summary of main notations.

Notation	Description
m	Device index
n	Channel index
t	Iteration index
f	Global loss function
f_m	Local loss function for device m
L	Assume $f_m(\cdot)$ is L-smooth
μ	Assume $f_m(\cdot)$ is μ-strongly convex
G	Assume $f_m(\cdot)$ has G-bounded second momentum
σ	Assume $f_m(\cdot)$ has σ-bounded variances
$\eta^{(t)}$	Gradient descent step size in iteration t
$\mathbf{w}^{(t)}$	Global model parameter in iteration t
$\widehat{\mathbf{w}}_m^{(t)}$	Local model parameter at device m in iteration t
$\widetilde{\mathbf{w}}_m^{(t)}$	Virtual local model parameter at device m in iteration t
γ_m	Minimum compression coefficient for device m
D	Total number of model parameters
H	Maximum number of local update steps
M	Total number of devices
N	Total number of channels
T	Total number of iterations
B_m	Total resource budget at device m
$\omega_m^{(t)}$	Number of CPU cycles required for computing at device m in round t
θ_m	Energy consumption coefficient of device m
ϵ_i	Energy consumption per byte of channel i
$P_i(w)$	Physical data size of w at channel i
$H_m^{(t)}$	Number of local update steps at device m in round t
$D_{m,n}^{(t)}$	Traffic allocation for channel n at device m in round t
$E_{m,comp}^{(t)}$	Total energy consumption for local computation of device m in round t
$E_{m,comm}^{(t)}$	Energy consumption factor for communication of device m in round t
$\mathcal{H}$	$\{1, 2, \ldots, H\}$
$\mathcal{M}$	$\{1, 2, \ldots, M\}$
$\mathcal{N}$	$\{1, 2, \ldots, N\}$
$\mathcal{T}$	$\{1, 2, \ldots, T\}$

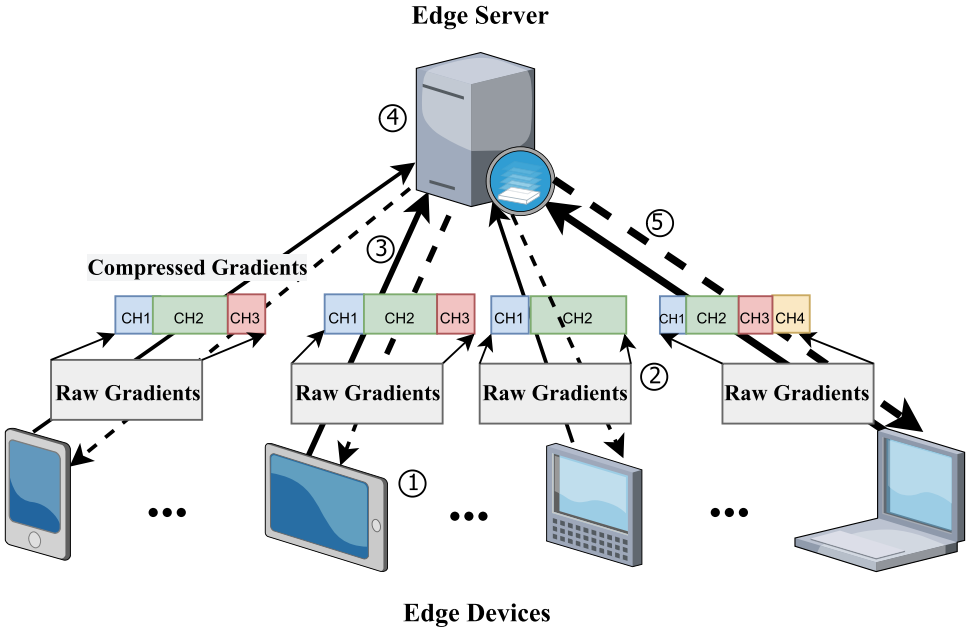


Fig. 1. An overview of Rainbow.

Upon receiving $\mathbf{g}_m^{(t)}$ from every device $m \in \mathcal{M}$ which sent its gradients, the server aggregates them, updates the global model, and sends the new model $\mathbf{w}^{(t+1)}$ to all the devices; upon receiving $\mathbf{w}^{(t+1)}$, the devices set their local model $\widehat{\mathbf{w}}_m^{(t+1)}$ to be equal to the new global model $\mathbf{w}^{(t+1)}$. The whole process of LGC in Rainbow is summarized in Algorithm 1.

2.3. Computation and communication model

We introduce resource consumption models for both computation and communication in Rainbow. We first define the computation model as the total computation energy consumption of device m at round

Algorithm 1 Layered Gradient Compression

Initialize $\mathbf{w}^{(0)}, \bar{\tilde{\mathbf{w}}}^{(0)}, \mathbf{w}_m^{(0)}, \hat{\mathbf{w}}_m^{(0)}, \mathbf{e}_m^{(0)}, \forall m \in \mathcal{M}$.
for $t = 0$ to $T-1$ **do**
  On Edge Devices:
  for $m \in \mathcal{M}$ in parallel **do**
    Sampling a mini-batch $D_m^{(t)}$ from D_m
    $\hat{\mathbf{w}}_m^{(t+\frac{1}{2})} \leftarrow \hat{\mathbf{w}}_m^{(t)} - \eta^{(t)} \nabla f_m(\hat{\mathbf{w}}_m^{(t)}; D_m^{(t)})$;
    if $t+1 \in \mathcal{I}_m$ **then**
      $\mathbf{u}_m^{(t)} \leftarrow \mathbf{e}_m^{(t)} + \mathbf{w}_m^{(t)} - \hat{\mathbf{w}}_m^{(t+\frac{1}{2})}$
      $\mathbf{g}_m^{(t)} = \mathrm{LGC}_{\mathbf{k}}(\mathbf{u}_m^{(t)})$
      Upload $\mathbf{g}_m^{(t)}$ by multiple channels
      $\mathbf{e}_m^{(t+1)} \leftarrow \mathbf{e}_m^{(t)} + \mathbf{w}_m^{(t)} - \hat{\mathbf{w}}_m^{(t+\frac{1}{2})} - \mathbf{g}_m^{(t)}$
      Receive $\bar{\tilde{\mathbf{w}}}^{(t+1)}$
      $\hat{\mathbf{w}}_m^{(t+1)} \leftarrow \bar{\tilde{\mathbf{w}}}^{(t+1)}$ and $\mathbf{w}_m^{(t+1)} \leftarrow \bar{\tilde{\mathbf{w}}}^{(t+1)}$
    else
      $\hat{\mathbf{w}}_m^{(t+1)} \leftarrow \hat{\mathbf{w}}_m^{(t+\frac{1}{2})}$
      $\mathbf{w}_m^{(t+1)} \leftarrow \mathbf{w}_m^{(t)}$
      $\mathbf{e}_m^{(t+1)} \leftarrow \mathbf{e}_m^{(t)}$
  At Central Server:
  if $t+1 \in \mathcal{I}_m, \forall m \in \mathcal{M}$ **then**
    Collect $\mathbf{g}_m^{(t)}$, $\forall m \in \mathcal{M}$ and $\mathbf{g}^{(t)} \leftarrow \frac{1}{M}\sum_{m=1}^{M} \mathbf{g}_m^{(t)}$
    $\bar{\tilde{\mathbf{w}}}^{(t+1)} \leftarrow \bar{\tilde{\mathbf{w}}}^{(t)} - \mathbf{g}^{(t)}$ and broadcast $\bar{\tilde{\mathbf{w}}}^{(t+1)}$
  else
    $\bar{\tilde{\mathbf{w}}}^{(t+1)} \leftarrow \bar{\tilde{\mathbf{w}}}^{(t)}$
Comment: $\mathbf{w}_m^{(t+\frac{1}{2})}$ denotes an intermediate variable between iterations t and $t+1$

t [14], given by:

$$E_{m,r,comp}^{(t)} = d_m \theta_m (\omega_m^{(t)})^2, \tag{3}$$

where d_m denotes the number of mini-batch data assigned to device m, $\omega_m^{(t)}$ denotes the number of CPU cycles required for computing single mini-batch data in device m at round t, and θ_m denotes an energy consumption coefficient which depends on the computation ability of device m.

Then we define the communication model as the total energy consumption factor for communication of device m at round t, given by:

$$E_{m,r,comm}^{(t)} = \sum_{i}^{n} \epsilon_i P_i(\hat{\mathbf{w}}_m^{(t)}), \tag{4}$$

where ϵ_i denotes the energy consumption per byte of channel i, $P_i(\hat{\mathbf{w}}_m^{(t)})$ denotes the data size of model parameter layer allocated to channel i at round t. The n denotes all available channels.

2.4. Problem formulation

In this part, we define the resource-efficient Rainbow with LGC. Considering that the resources of different mobile devices vary, we formulate the optimization problem to minimize the global loss function under resource constraints as follows.

$$\min_{\{T, H_m^{(t)}, D_{m,n}^{(t)}\}} f(\mathbf{w}^{(T)}), \tag{5}$$

subject to,

$$\sum_{t=1}^{T} \left(E_{m,r,comp}^{(t)} H_m^{(t)} + \sum_{n=1}^{N} E_{m,r,comm}^{(t)} D_{m,n}^{(t)} \right) \le B_{m,r}, \quad \forall m \in \mathcal{M}, \forall r \in \mathcal{R} \tag{6a}$$

$$\sum_{n=1}^{N} D_{m,n}^{(t)} \le D, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \tag{6b}$$

$$H_m^{(t)} \le H, \forall m \in \mathcal{M}, \forall t \in \mathcal{T}, \tag{6c}$$

$H_m^{(t)}$ represents the number of local update steps at device m in round t. $D_{m,n}^{(t)}$ indicates the traffic allocation for channel n at device m in round t. $B_{m,r}$ represents the total budget of resource type r in device m. For each device m, each resource has a budget $B_{m,r}$, where r represents the type of resource, such as one for communication resource (in seconds) and one for computing resource (in joules). $B_{m,r}$ indicates the total budget of resource type r on device m. Note that the difference in resource type units has no impact on the convergence of the objective function.

Since FL is typically deployed in highly dynamic edge networks, a learning-driven method could be helpful to adaptively adjust the local computation and communication decision while satisfying the resource constraints at each epoch in MEC.

3. Convergence analysis of LGC

In this section, we first describe the main results of convergence analysis for LGC in Rainbow and then we prove the convergence results.

3.1. Main results

We consider the following two standard assumptions on the local loss functions $f_m : \mathbb{R}^d \to \mathbb{R}, \forall m \in \mathcal{M}$

Assumption 1 (*Smoothness*). $f_m(\cdot)$ is L-smooth, i.e., for every $\mathbf{w}, \mathbf{w}' \in \mathbb{R}^d$, we have

$$f_m(\mathbf{w}) \le f_m(\mathbf{w}') + \langle \nabla f_m(\mathbf{w}), \mathbf{w}' - \mathbf{w} \rangle + \frac{L}{2} \|\mathbf{w}' - \mathbf{w}\|^2. \tag{7}$$

Assumption 2 (*Bounded Variances and Second Momentum*). For every $\mathbf{w}_m^{(t)} \in \mathbb{R}^d$ and $t \in \mathbb{Z}^+$, there exist constants $\sigma > 0$ and $G \ge \sigma$ such that:

$$\mathbb{E}_{D_m^{(t)} \subset D_m} \left[\| \nabla f_m(\mathbf{w}_m^{(t)}; D_m^{(t)}) - \nabla f_m(\mathbf{w}_m^{(t)}) \|^2 \right] \le \sigma^2, \forall m, \tag{8a}$$

$$\mathbb{E}_{D_m^{(t)} \subset D_m} \left[\left\| \nabla f_m(\mathbf{w}_m^{(t)}; D_m^{(t)}) \right\|^2 \right] \le G^2, \forall m. \tag{8b}$$

We extend Lemma 4 in [12] and get the following lemma.

Lemma 1 (*Memory Contraction*). *Let* $\mathrm{gap}(\mathcal{I}_m) \le H, \forall m \in \mathcal{M}$ *and* $\eta^{(t)} = \frac{\xi}{a+t}$, *where* ξ *is a constant and* $a > \frac{4H}{\gamma}$. *Then there exists a constant* $C \ge \frac{4a\gamma_m(1-\gamma_m^2)}{a\gamma_m - 4H}$, *the following holds for every* $t \in \mathbb{Z}^+$ *and* $m \in \mathcal{M}$:

$$\mathbb{E} \left\| \mathbf{e}_m^{(t)} \right\|_2^2 \le 4 \frac{(\eta^{(t)})^2}{\gamma_m^2} C H^2 G^2. \tag{9}$$

We leverage the perturbed iterate analysis as in [11,15] to provide convergence guarantees for LGC. With the above assumptions, the following Theorem holds for Algorithm 1.

Theorem 1 (*Smooth and Strongly Convex Case with a Decaying Learning Rate*). *Let* $f_m(\mathbf{w})$ *be* L*-smooth and* μ*-strongly convex,* $\forall m \in \mathcal{M}$. *Let* $\{\hat{\mathbf{w}}_m^{(t)}\}_{t=0}^{T-1}$ *be generated according to Algorithm 1 with* $C_m^{(t)}$, *for step sizes* $\eta^{(t)} = 8/\mu(a+t)$ *with* $gap(\mathcal{I}) \le H$, *where* $a > 1$ *is such that we have* $a > \max\{4H/\gamma, 32\kappa, H\}$, $\kappa = L/\mu$. *The following holds*

$$\mathbb{E}[f(\overline{\mathbf{w}}^{(T)})] - f^* \le \frac{La^3}{4S} \|\mathbf{w}^{(0)} - \mathbf{w}^*\|_2^2 + \frac{8LT(T+2a)}{\mu^2 S} A + \frac{128LT}{\mu^3 S} B, \tag{10}$$

where

$$C = \min_{m \in \mathcal{M}} \frac{4a\gamma_m(1-\gamma_m^2)}{a\gamma_m - 4H}, \tag{11a}$$

$$C_1 = \frac{192}{M}\sum_{m=1}^{M}(4-2\gamma_m)(1+\frac{C}{\gamma_m^2}), \tag{11b}$$

$$C_2 = \frac{8}{M}\sum_{m=1}^{M}(4-2\gamma_m)(1+\frac{C}{\gamma_m^2}), \tag{11c}$$

$$A = \frac{\sum_{m=1}^{M}\sigma_m^2}{bM^2}, \tag{11d}$$

$$B = (\frac{3\mu}{2}+3L)(\frac{12CG^2H^2}{\gamma^2}+C_1(\eta^{(t)})^2H^4G^2)+24(1+C_2H^2)LG^2H^2, \tag{11e}$$

$$\overline{\mathbf{w}}^{(T)} = \frac{1}{S}\sum_{t=0}^{T-1}\left[s^{(t)}\left(\frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{w}}_m^{(t)}\right)\right] = \frac{1}{S}\sum_{t=0}^{T-1}s^{(t)}\widehat{\mathbf{w}}^{(t)}, \tag{11f}$$

$$s^{(t)} = (a+t)^2, \tag{11g}$$

$$S = \sum_{t=0}^{T-1}s^{(t)} \geq \frac{T^3}{3}. \tag{11h}$$

Corollary 1. *For gap$(\mathcal{I}_m) \leq H$, $a > \max\{4H/\gamma, 32\kappa, H\}$, $\sigma_{max} = \max_{m\in\mathcal{M}}\sigma_m$, if $\{\mathbf{w}_m^{(t)}\}_{t=0}^{T-1}$ is generated according to Algorithm 1 and using $\mathbb{E}\|\mathbf{w}^{(0)}-\mathbf{w}^*\|_2^2 \leq \frac{4G^2}{\mu^2}$ from Lemma 2 in [16], we have*

$$\begin{aligned}\mathbb{E}[f(\overline{\mathbf{x}}^{(T)})]-f^* \leq& \mathcal{O}\left(\frac{G^2H^3}{\mu^2\gamma^3T^3}\right)+\mathcal{O}\left(\frac{\sigma_{max}^2}{\mu^2bRT}+\frac{H\sigma_{max}^2}{\mu^2bR\gamma T^2}\right)\\ &+\mathcal{O}\left(\frac{G^2}{\mu^3\gamma^2T^2}(H^2+H^4)\right).\end{aligned} \tag{12}$$

3.2. Proof

Inspired by the perturbed iterate analysis framework [15], we define virtual sequences for every device $m \in \mathcal{M}$ and for all $t \geq 0$ as follows:

$$\widetilde{\mathbf{w}}_m^{(0)} := \widehat{\mathbf{w}}_m^{(0)} \quad \widetilde{\mathbf{w}}_m^{(t+1)} := \widetilde{\mathbf{w}}_m^{(t)} - \eta^{(t)}\nabla f_m(\widehat{\mathbf{w}}_m^{(t)}; \mathcal{D}_m^{(t)}). \tag{13}$$

We also define

$$\begin{aligned}\mathbf{q}^{(t)} &:= \frac{1}{M}\sum_{m=1}^{M}\nabla f_m(\widehat{\mathbf{w}}_m^{(t)}; \mathcal{D}_m^{(t)}),\\ \overline{\mathbf{q}}^{(t)} &:= \mathbb{E}_{\mathcal{D}_{\mathcal{M}}^{(t)}}[\mathbf{q}^{(t)}] = \frac{1}{M}\sum_{m=1}^{M}\nabla f_m(\widehat{\mathbf{w}}_m^{(t)}),\\ \widetilde{\mathbf{w}}^{(t+1)} &:= \frac{1}{M}\sum_{m=1}^{M}\widetilde{\mathbf{w}}_m^{(t+1)} = \widetilde{\mathbf{w}}^{(t)} - \eta^{(t)}\mathbf{q}^{(t)},\\ \widehat{\mathbf{w}}^{(t)} &:= \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{w}}_m^{(t)},\\ \mathcal{I}_m &= \left\{t_m^{(i)} : i \in \mathbb{Z}^+, t_m^{(i)} \in \mathcal{T}, \left|t_m^{(i)} - t_m^{(j)}\right| \leq H, \forall|i-j| \leq 1\right\}.\end{aligned} \tag{14}$$

3.2.1. Proof of Theorem 1

Proof. Let $\mathbf{w}^*$ be the minimizer of $f(\mathbf{w})$, therefore we have $\nabla f(\mathbf{w}^*) = 0$. We denote $f(\mathbf{w}^*)$ by f^*. By taking the average of the virtual sequences $\widetilde{\mathbf{w}}_m^{(t+1)} = \widetilde{\mathbf{w}}_m^{(t)} - \eta^{(t)}\nabla f_m(\widehat{\mathbf{w}}_m^{(t)}; \mathcal{D}_m^{(t)})$ for each device $m \in \mathcal{M}$ and defining $\mathbf{q}^{(t)} := \frac{1}{M}\sum_{m=1}^{M}\nabla f_m(\widehat{\mathbf{w}}_m^{(t)}; \mathcal{D}_m^{(t)})$, we get

$$\widetilde{\mathbf{w}}^{(t+1)} = \widetilde{\mathbf{w}}^{(t)} - \eta^{(t)}\mathbf{q}^{(t)}. \tag{15}$$

Define $\mathcal{D}_{\mathcal{M}}^{(t)}$ as the set of random sampling of the mini-batches at each worker $\left\{\mathcal{D}_1^{(t)}, \mathcal{D}_2^{(t)}, \ldots, \mathcal{D}_M^{(t)}\right\}$ and let $\overline{\mathbf{q}}^{(t)} = \mathbb{E}_{\mathcal{D}_{\mathcal{M}}^{(t)}}[\mathbf{q}^{(t)}] = \frac{1}{M}\sum_{m=1}^{M}\nabla f_m(\widehat{\mathbf{w}}_m^{(t)})$. From Eq. (15) we can get

$$\begin{aligned}\left\|\widetilde{\mathbf{w}}^{(t+1)}-\mathbf{w}^*\right\|^2 &= \left\|\widetilde{\mathbf{w}}^{(t)}-\eta^{(t)}\mathbf{q}^{(t)}-\mathbf{w}^*-\eta^{(t)}\overline{\mathbf{q}}^{(t)}+\eta^{(t)}\overline{\mathbf{q}}^{(t)}\right\|^2\\ &= \left\|\widetilde{\mathbf{w}}^{(t)}-\mathbf{w}^*-\eta^{(t)}\overline{\mathbf{q}}^{(t)}\right\|^2+(\eta^{(t)})^2\left\|\mathbf{q}^{(t)}-\overline{\mathbf{q}}^{(t)}\right\|^2\\ &\quad-2\eta^{(t)}\left\langle\widetilde{\mathbf{w}}^{(t)}-\mathbf{w}^*-\eta^{(t)}\overline{\mathbf{q}}^{(t)}, \mathbf{q}^{(t)}-\overline{\mathbf{q}}^{(t)}\right\rangle.\end{aligned} \tag{16}$$

Taking the expectation w.r.t. the sampling $\mathcal{D}_{\mathcal{M}}^{(t)}$ at time t (conditioning on the past) and noting that last term in Eq. (16) becomes zero gives:

$$\mathbb{E}_{\mathcal{D}_{\mathcal{M}}^{(t)}}\left\|\widetilde{\mathbf{w}}^{(t+1)}-\mathbf{w}^*\right\|^2 = \left\|\widetilde{\mathbf{w}}^{(t)}-\mathbf{w}^*-\eta^{(t)}\overline{\mathbf{q}}^{(t)}\right\|^2+(\eta^{(t)})^2\left\|\mathbf{q}^{(t)}-\overline{\mathbf{q}}^{(t)}\right\|^2. \tag{17}$$

It follows from the Jensen's inequality and independence that $\mathbb{E}_{\mathcal{D}_{\mathcal{M}}^{(t)}}\|\mathbf{q}^{(t)}-\overline{\mathbf{q}}^{(t)}\|^2 \leq \frac{\sum_{m=1}^{M}\sigma_m^2}{bM^2}$. This gives

$$\mathbb{E}_{\mathcal{D}_{\mathcal{M}}^{(t)}}\left\|\widetilde{\mathbf{w}}^{(t+1)}-\mathbf{w}^*\right\|^2 = \left\|\widetilde{\mathbf{w}}^{(t)}-\mathbf{w}^*-\eta^{(t)}\overline{\mathbf{q}}^{(t)}\right\|^2+(\eta^{(t)})^2\frac{\sum_{m=1}^{M}\sigma_m^2}{bM^2}. \tag{18}$$

Now we bound the first term on the RHS. Using μ-strong convexity and L-smoothness of f, together with some algebraic manipulations provided in Lemma 14 of [12], we arrive at

$$\begin{aligned}\mathbb{E}\|\widetilde{\mathbf{w}}^{(t+1)}-\mathbf{w}^*\|_2^2 \leq& (1-\frac{\mu\eta^{(t)}}{2})\mathbb{E}\|\widetilde{\mathbf{w}}^{(t)}-\mathbf{w}^*\|_2^2-\frac{\eta^{(t)}\mu}{2L}(\mathbb{E}\left[f\left(\widehat{\mathbf{w}}^{(t)}\right)\right]-f^*)\\ &+\eta^{(t)}(\frac{3\mu}{2}+3L)\mathbb{E}\|\widehat{\mathbf{w}}^{(t)}-\widetilde{\mathbf{w}}^{(t)}\|_2^2\\ &+\frac{3\eta^{(t)}L}{M}\sum_{m=1}^{M}\mathbb{E}\|\widehat{\mathbf{w}}^{(t)}-\widehat{\mathbf{w}}_m^{(t)}\|_2^2\\ &+(\eta^{(t)})^2\frac{\sum_{m=1}^{M}\sigma_m^2}{bM^2}.\end{aligned} \tag{19}$$

Now we have to bound the deviation of local sequences $\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)}-\widehat{\mathbf{w}}_m^{(t)}\right\|_2^2$ and the difference between the virtual and true sequences $\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)}-\widetilde{\mathbf{w}}^{(t)}\right\|_2^2$. We show these below in Lemmas 2 and 3.

Lemma 2 (*Contracting Local Sequence Deviation*)**.** *Let gap$\left(\mathcal{I}_m\right) \leq H$ holds for every $m \in \mathcal{M}$. For $\widehat{\mathbf{w}}_t^{(r)}$ generated according to Algorithm 1 with decaying learning rate $\eta^{(t)}$ and letting $\widehat{\mathbf{w}}^{(t)} = \frac{1}{M}\sum_{m=1}^{M}\widehat{\mathbf{w}}_m^{(t)}$, we have the following bound on the deviation of the local sequences:*

$$\frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)}-\widehat{\mathbf{w}}_m^{(t)}\right\|_2^2 \leq 8\left(1+C''H^2\right)(\eta^{(t)})^2G^2H^2, \tag{20}$$

where $C'' = \frac{8}{M}\sum_{m=1}^{M}(4-2\gamma_m)\left(1+\frac{C}{\gamma_m^2}\right)$ and C is a constant satisfying $C \geq \min_{m\in\mathcal{M}}\frac{4a\gamma_m(1-\gamma_m^2)}{a\gamma_m-4H}$.

Lemma 3 (*Contracting Distance Between Virtual and True Sequence*)**.** *Let gap$\left(\mathcal{I}_m\right) \leq H$ holds for every $m \in \mathcal{M}$. If we run Algorithm 1 with a decaying learning rate $\eta^{(t)}$, then we have the following bound on the difference between the true and virtual sequences:*

$$\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)}-\widetilde{\mathbf{w}}^{(t)}\right\|_2^2 \leq C'(\eta^{(t)})^2H^4G^2+12C\frac{(\eta^{(t)})^2}{\gamma^2}G^2H^2, \tag{21}$$

where $C' = 192C'' = \frac{8}{M}\sum_{m=1}^{M}(4-2\gamma_m)\left(1+\frac{C}{\gamma_m^2}\right)$ and C is a constant satisfying $C \geq \min_{m\in\mathcal{M}}\frac{4a\gamma_m(1-\gamma_m^2)}{a\gamma_m-4H}$.

Substituting the bounds from Eqs. (20) (21) into Eq. (19) yields

$$\begin{aligned}&\mathbb{E}\|\widetilde{\mathbf{w}}^{(t+1)}-\mathbf{w}^*\|_2^2\\ &\leq(1-\frac{\mu\eta^{(t)}}{2})\mathbb{E}\|\widetilde{\mathbf{w}}^{(t)}-\mathbf{w}^*\|_2^2-\frac{\eta^{(t)}\mu}{2L}\varepsilon^{(t)}\\ &+\eta^{(t)}(\frac{3\mu}{2}+3L)[C'(\eta^{(t)})^2H^4G^2+12C\frac{(\eta^{(t)})^2}{\gamma^2}G^2H^2]\\ &+\frac{(3\eta^{(t)}L)}{M}[8\left(1+C''H^2\right)(\eta^{(t)})^2G^2H^2]\\ &+(\eta^{(t)})^2\frac{\sum_{m=1}^{M}\sigma_m^2}{bM^2}.\end{aligned} \tag{22}$$

Employing a slightly modified result than Lemma 3.3 in [11] with $a^{(t)} = \mathbb{E}\left\|\widetilde{\mathbf{w}}^{(t)} - \mathbf{w}^*\right\|_2^2$, $A = \frac{\sum_{m=1}^M \sigma_m^2}{bM^2}$ and $B = (\frac{3\mu}{2} + 3L)(\frac{12CG^2H^2}{\gamma^2} + C_1(\eta^{(t)})^2H^4G^2) + 24(1+C_2H^2)LG^2H^2$, we have

$$a^{(t+1)} \leq \left(1 - \frac{\mu\eta^{(t)}}{2}\right)a^{(t)} - \frac{\mu\eta^{(t)}}{2L}\epsilon^{(t)} + (\eta^{(t)})^2 A + (\eta^{(t)})^3 B.$$

For $\eta^{(t)} = \frac{8}{\mu(a+t)}$ and $s^{(t)} = (a+t)^2$, $S = \sum_{t=0}^{T-1} s^{(t)} \geq \frac{T^3}{3}$, we have

$$\frac{\mu}{2LS}\sum_{t=0}^{T-1} s^{(t)}\epsilon^{(t)} \leq \frac{\mu a^3}{8S}a^{(0)} + \frac{4T(T+2a)}{\mu S}A + \frac{64T}{\mu^2 S}B.$$

From convexity, we can finally write

$$\mathbb{E}f\left(\overline{\mathbf{w}}^{(T)}\right) - f^* \leq \frac{La^3}{4S}a^{(0)} + \frac{8LT(T+2a)}{\mu^2 S}A + \frac{128LT}{\mu^3 S}B.$$

where $\overline{\mathbf{w}}^{(T)} := \frac{1}{S}\sum_{t=0}^{T-1}\left[s^{(t)}\left(\frac{1}{M}\sum_{m=1}^M \widehat{\mathbf{w}}_m^{(t)}\right)\right] = \frac{1}{S}\sum_{t=0}^{T-1} s^{(t)}\widehat{\mathbf{w}}^{(t)}$. This completes the proof of Theorem 1.

3.2.2. Proof of Lemma 2

Proof. Fix a time t and consider any device $m \in \mathcal{M}$. Let $t_m \in \mathcal{I}_m$ denote the last synchronization step until time t for the mth worker. Define $t_0' := \min_{m\in\mathcal{M}} t_m$. We need to upper-bound $\frac{1}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)} - \widehat{\mathbf{w}}_m^{(t)}\right\|^2$. Note that for any M vectors $\mathbf{u}_1, \ldots, \mathbf{u}_M$, if we let $\overline{\mathbf{u}} = \frac{1}{M}\sum_{i=1}^M \mathbf{u}_i$, then $\sum_{i=1}^n \left\|\mathbf{u}_i - \overline{\mathbf{u}}\right\|^2 \leq \sum_{i=1}^M \left\|\mathbf{u}_i\right\|^2$. We use this in the first inequality below.

$$\begin{aligned}
&\frac{1}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)} - \widehat{\mathbf{w}}_m^{(t)}\right\|^2 \\
&= \frac{1}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t)} - \overline{\widehat{\mathbf{w}}}^{(t_0')} - \left(\widehat{\mathbf{w}}^{(t)} - \overline{\widehat{\mathbf{w}}}^{(t_0')}\right)\right\|^2 \\
&\leq \frac{1}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t)} - \overline{\widehat{\mathbf{w}}}^{(t_0')}\right\|^2 \\
&\leq \frac{2}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t)} - \widehat{\mathbf{w}}_m^{(t_m)}\right\|^2 + \frac{2}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t_m)} - \overline{\widehat{\mathbf{w}}}^{(t_0')}\right\|^2.
\end{aligned} \tag{23}$$

We bound both the terms separately. For the first term:

$$\begin{aligned}
\mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t)} - \widehat{\mathbf{w}}_m^{(t_m)}\right\|^2 &= \mathbb{E}\left\|\sum_{j=t_m}^{t-1}\eta^{(j)}\nabla f_{\mathcal{D}_m^{(j)}}\left(\widehat{\mathbf{w}}_m^{(j)}\right)\right\|^2 \\
&\leq (t-t_m)\sum_{j=t_m}^{t-1}\mathbb{E}\left\|\eta^{(j)}\nabla f_{\mathcal{D}_m^{(j)}}\left(\widehat{\mathbf{w}}_m^{(j)}\right)\right\|^2 \\
&\leq (t-t_m)^2(\eta^{(t_m)})^2G^2 \\
&\leq 4(\eta^{(t)})^2H^2G^2.
\end{aligned} \tag{24}$$

The last inequality Eq. (24) uses $\eta^{(t_m)} \leq 2\eta^{(t_m+H)} \leq 2\eta^{(t)}$ and $t - t_m \leq H$. To bound the second term of Eq. (32), note that we have

$$\bar{\widehat{\mathbf{w}}}_m^{t_m} = \bar{\widehat{\mathbf{w}}}^{(t_0')} - \frac{1}{M}\sum_{s=1}^M\sum_{j=t_0'}^{t_m-1}\mathbb{1}\left\{j+1 \in \mathcal{I}_s\right\}g_s^{(j)}. \tag{25}$$

Note that $\widehat{\mathbf{w}}_m^{(t_m)} = \overline{\widehat{\mathbf{w}}}_m^{(t_m)}$, because at synchronization steps, the local parameter vector becomes equal to the global parameter vector. Using this, the Jensen's inequality, and that $\left\|\mathbb{1}\left\{j+1\in\mathcal{I}_s\right\}g_s^{(j)}\right\|^2 \leq \left\|g_s^{(j)}\right\|^2$, we can upper-bound Eq. (25) as

$$\mathbb{E}\left\|\widehat{\mathbf{w}}_m^{t_m} - \overline{\widehat{\mathbf{w}}}^{(t_0')}\right\|^2 \leq \frac{(t_m - t_0')}{M}\sum_{s=1}^M\sum_{j=t_0'}^{t_m}\mathbb{E}\left\|g_s^{(j)}\right\|^2. \tag{26}$$

Now we bound $\mathbb{E}\left\|g_s^{(j)}\right\|^2$ for any $j \in \{t_0', \ldots, t_m\}$ and $s \in \mathcal{M}$: Since $\mathbb{E}\left\|C_m^{(t)}(\mathbf{u})\right\|^2 \leq B\|\mathbf{u}\|^2$ holds for every $\mathbf{u}$, with $B = (4-2\gamma_m)$,[1] we have for any $s \in \mathcal{M}$ that

$$\begin{aligned}
\mathbb{E}\left\|\mathbf{g}_s^{(j)}\right\|^2 &\leq B\mathbb{E}\left\|\mathbf{e}_s^{(j)} + \mathbf{w}_s^{(j)} - \widehat{\mathbf{w}}_s^{(j+\frac{1}{2})}\right\|^2 \\
&\leq 2B\mathbb{E}\left\|\mathbf{e}_s^{(j)}\right\|^2 + 2B\mathbb{E}\left\|\mathbf{w}_s^{(j)} - \widehat{\mathbf{w}}_s^{(j+\frac{1}{2})}\right\|^2.
\end{aligned} \tag{27}$$

We can directly use Lemma 4 in [12] to bound the first term in Eq. (27) as $\mathbb{E}\left\|\mathbf{e}_j^{(s)}\right\|^2 \leq 4C\frac{(\eta^{(j)})^2}{\gamma_m^2}H^2G^2$. In order to bound the second term of Eq. (27), note that $\mathbf{w}_s^{(j)} = \widehat{\mathbf{w}}_s^{(t_s)}$, which implies that $\left\|\mathbf{w}_s^{(j)} - \widehat{\mathbf{w}}_s^{(j+\frac{1}{2})}\right\|^2 = \left\|\sum_{l=t_s}^j \eta^{(l)}\nabla f_{\mathcal{D}_s^{(l)}}\left(\widehat{\mathbf{w}}_l^{(s)}\right)\right\|^2$ Taking expectation yields $\mathbb{E}\left\|\mathbf{w}_s^{(j)} - \widehat{\mathbf{w}}_s^{(j+\frac{1}{2})}\right\|^2 \leq 4(\eta^{(t_s)})^2H^2G^2 \leq 4(\eta^{(t_0')})^2H^2G^2$, where in the last inequality we used that $t_0' \leq t_s$. Using these in Eq. (27) gives

$$\mathbb{E}\left\|g_s^{(j)}\right\|^2 \leq 8B\left(1 + \frac{C}{\gamma_m^2}\right)(\eta^{(t_0')})^2H^2G^2. \tag{28}$$

Since $t_0' \leq t \leq t_0' + H$, we have $\eta^{(t_0')} \leq 2\eta^{(t_0'+H)} \leq 2\eta^{(t)}$. Putting the bound on $\mathbb{E}\left\|\mathbf{g}_s^{(j)}\right\|^2$ (after substituting $\eta^{(t_0')} \leq 2\eta^{(t)}$ in Eq. (28) in Eq. (26) gives

$$\mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t_m)} - \overline{\widehat{\mathbf{w}}}^{(t_0')}\right\|^2 \leq 32\frac{1}{M}\left[\sum_{m=1}^M(4-2\gamma_m)\left(1+\frac{C}{\gamma_m^2}\right)\right](\eta^{(t)})^2H^4G^2. \tag{29}$$

Putting this and the bound from Eq. (24) back in Eq. (23) gives

$$\begin{aligned}
&\frac{1}{M}\sum_{m=1}^M \mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)} - \widehat{\mathbf{w}}_m^{(t)}\right\|^2 \\
&\leq 8(\eta^{(t)})^2H^2G^2 + 64\frac{1}{M}\left[\sum_{m=1}^M(4-2\gamma_m)\left(1+\frac{C}{\gamma_m^2}\right)\right](\eta^{(t)})^2H^4G^2 \\
&\leq 8\left[1 + 8\left[\frac{1}{M}\sum_{m=1}^M(4-2\gamma_m)\left(1+\frac{C}{\gamma_m^2}\right)\right]H^2\right](\eta^{(t)})^2H^2G^2.
\end{aligned} \tag{30}$$

This completes the proof of Lemma 2.

3.2.3. Proof of Lemma 3

Proof. Fix a time t and consider any device $m \in \mathcal{M}$. Let $t_m \in \mathcal{I}_m$ denote the last synchronization step until time t for the mth device. Define $t_0' := \min_{m\in\mathcal{M}} t_m$. We want to bound $\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{w}}^{(t)}\right\|^2$. By definition $\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{w}}^{(t)} = \frac{1}{M}\sum_{m=1}^M\left(\widehat{\mathbf{w}}_m^{(t)} - \widetilde{\mathbf{w}}_m^{(t)}\right)$. By the definition of virtual sequences and the update rule for $\widehat{\mathbf{w}}_m^{(t)}$, we also have $\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{w}}^{(t)} = \frac{1}{M}\sum_{m=1}^M\left(\widehat{\mathbf{w}}_m^{(t_m)} - \widetilde{\mathbf{w}}_m^{(t_m)}\right)$. This can be written as

$$\begin{aligned}
\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{w}}^{(t)} = &\left[\frac{1}{M}\sum_{m=1}^M\widehat{\mathbf{w}}_m^{(t_m)} - \bar{\widehat{\mathbf{w}}}^{(t_0')}\right] + \left[\bar{\widehat{\mathbf{w}}}^{(t_0')} - \bar{\widehat{\mathbf{w}}}^{(t)}\right] \\
&+ \left[\bar{\widehat{\mathbf{w}}}^{(t)} - \frac{1}{M}\sum_{m=1}^M\widetilde{\mathbf{w}}_m^{(t_m)}\right].
\end{aligned} \tag{31}$$

Applying Jensen's inequality and taking expectation gives

$$\begin{aligned}
\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)} - \widetilde{\mathbf{w}}^{(t)}\right\|^2 \leq &\left[\frac{3}{M}\sum_{m=1}^M\mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t_m)} - \bar{\widehat{\mathbf{w}}}^{(t_0')}\right\|^2\right] + \left[3\mathbb{E}\left\|\bar{\widehat{\mathbf{w}}}^{(t_0')} - \bar{\widehat{\mathbf{w}}}^{(t)}\right\|^2\right] \\
&+ \left[3\mathbb{E}\left\|\bar{\widehat{\mathbf{w}}}^{(t)} - \frac{1}{M}\sum_{m=1}^M\widetilde{\mathbf{w}}_m^{(t_m)}\right\|^2\right].
\end{aligned} \tag{32}$$

[1] This can be seen as follows: $\mathbb{E}\|C_m^{(t)}(\mathbf{u})\|^2 \leq 2\mathbb{E}\|\mathbf{u} - C_m^{(t)}(\mathbf{u})\|^2 + 2\|\mathbf{u}\|^2 \leq 2(1-\gamma_m)\|\mathbf{u}\|^2 + 2\|\mathbf{u}\|^2$

We bound each of the three terms of Eq. (32) separately. We have upper-bounded the first term earlier in Eq. (32), which is

$$\mathbb{E}\left\|\widehat{\mathbf{w}}_m^{(t_m)}-\bar{\widetilde{\mathbf{w}}}^{(t_0')}\right\|^2 \leq 32B\left(1+\frac{C}{\gamma^2}\right)(\eta^{(t)})^2H^4G^2, \tag{33}$$

where $B=(4-2\gamma)$. To bound the second term of Eq. (32), note that

$$\begin{aligned}\bar{\widetilde{\mathbf{w}}}^{(t)} &= \bar{\widetilde{\mathbf{w}}}^{(0)}-\frac{1}{M}\sum_{m=1}^{M}\sum_{j=0}^{t_m-1}\mathbb{1}\left\{j+1\in\mathcal{I}_m\right\}\mathbf{g}_m^{(t)}\\ &= \bar{\widetilde{\mathbf{w}}}^{(t_0')}-\frac{1}{M}\sum_{m=1}^{M}\sum_{j=t_0'}^{t_m-1}\mathbb{1}\left\{j+1\in\mathcal{I}_m\right\}\mathbf{g}_m^{(j)}.\end{aligned} \tag{34}$$

By applying Jensen's inequality, using $\left\|\mathbb{1}\left\{j+1\in\mathcal{I}_m\right\}\mathbf{g}_m^{(j)}\right\|^2 \leq \left\|\mathbf{g}_m^{(j)}\right\|^2$, and taking expectation, we can upper-bound Eq. (34) as

$$\mathbb{E}\left\|\bar{\widetilde{\mathbf{w}}}^{(t_0')}-\bar{\widetilde{\mathbf{w}}}^{(t)}\right\|^2 \leq \frac{\left(t_m-t_0'\right)}{M}\sum_{m=1}^{M}\sum_{j=t_0'}^{t_m}\mathbb{E}\left\|\mathbf{g}_m^{(j)}\right\|^2. \tag{35}$$

Using the bound on $\mathbb{E}\left\|\mathbf{g}_m^{(j)}\right\|^2$'s from Eq. (29) gives

$$\mathbb{E}\left\|\bar{\widetilde{\mathbf{w}}}^{(t_0')}-\bar{\widetilde{\mathbf{x}}}^{(t)}\right\|^2 \leq 32B\left(1+\frac{C}{\gamma^2}\right)(\eta^{(t)})^2H^4G^2. \tag{36}$$

To bound the last term of Eq. (32), note that

$$\widetilde{\mathbf{w}}_m^{(t_m)} = \bar{\widetilde{\mathbf{w}}}^{(0)}-\sum_{j=0}^{t_m-1}(\eta^{(j)})\nabla f_{\mathcal{D}_m^{(j)}}\left(\widehat{\mathbf{w}}_m^{(j)}\right). \tag{37}$$

From Eq. (34) and Eq. (37), we can write

$$\begin{aligned}&\bar{\widetilde{\mathbf{w}}}^{(t)}-\frac{1}{M}\sum_{m=1}^{M}\widetilde{\mathbf{w}}_m^{(t_m)}\\ &=\frac{1}{M}\sum_{m=1}^{M}\left[\sum_{j=0}^{t_m-1}\eta^{(j)}\nabla f_{\mathcal{D}_m^{(j)}}\left(\widehat{\mathbf{w}}_m^{(j)}\right)-\sum_{j=0}^{t_m-1}\mathbb{1}\left\{j+1\in\mathcal{I}_m\right\}\mathbf{g}_m^{(j)}\right].\end{aligned} \tag{38}$$

Let $t_m^{(1)}$ and $t_m^{(2)}$ be two consecutive synchronization steps in $\mathcal{I}_m$. Then, by the update rule of $\widehat{\mathbf{w}}_m^{(t)}$, we have $\widehat{\mathbf{w}}_m^{(t_m^{(1)})}-\widehat{\mathbf{w}}_m^{(t_m^{(2)}-\frac{1}{2})}=\sum_{j=t_m^{(1)}}^{t_m^{(2)}-1}\nabla f_{\mathcal{D}_m^{(j)}}\left(\widehat{\mathbf{w}}_m^{(j)}\right)$. Since $\mathbf{w}_m^{(t_m^{(1)})}=\widehat{\mathbf{w}}_m^{(t_m^{(1)})}$ and the devices do not modify their local $\mathbf{w}_m^{(t)}$'s in between the synchronization steps, we have $\mathbf{w}_m^{(t_m^{(2)}-1)}=\mathbf{w}_m^{(t_m^{(1)})}=\widehat{\mathbf{w}}_m^{(t_m^{(1)})}$. Therefore, we can write

$$\mathbf{w}_m^{(t_m^{(2)}-1)}-\widehat{\mathbf{w}}_m^{(t_m^{(2)}-\frac{1}{2})}=\sum_{j=t_m^{(1)}}^{t_m^{(2)}-1}\nabla f_{\mathcal{D}_m^{(j)}}\left(\widehat{\mathbf{w}}_m^{(j)}\right). \tag{39}$$

Using Eq. (39) for every consecutive synchronization steps, we can equivalently write Eq. (38) as

$$\begin{aligned}&\bar{\widetilde{\mathbf{w}}}^{(t)}-\frac{1}{M}\sum_{m=1}^{M}\widetilde{\mathbf{w}}_m^{(t_m)}\\ &=\frac{1}{M}\sum_{m=1}^{M}\left[\sum_{j:j+1\in\mathcal{I}_m,j\leq t_m-1}\left(\mathbf{w}_m^{(j)}-\widehat{\mathbf{w}}_m^{(j+\frac{1}{2})}-\mathbf{g}_m^{(j)}\right)\right]\\ &=\frac{1}{M}\sum_{m=1}^{M}\mathbf{e}_m^{(t_m)}\\ &=\frac{1}{M}\sum_{m=1}^{M}\mathbf{e}_m^{(t)}.\end{aligned} \tag{40}$$

In the last inequality, we used the fact that the devices do not update their local memory in between the synchronization steps. For the reasons given in the proof of Lemma 2, we can directly apply Lemma 4 in [12] to bound the local memories and obtain $\mathbb{E}\left\|\frac{1}{M}\sum_{m=1}^{M}\mathbf{e}_m^{(t)}\right\|^2 \leq \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left\|\mathbf{e}_m^{(t)}\right\|^2 \leq 4C\frac{(\eta^{(t)})^2}{\gamma^2}G^2H^2$. This implies

$$\mathbb{E}\left\|\bar{\widetilde{\mathbf{w}}}^{(t)}-\frac{1}{M}\sum_{m=1}^{M}\widetilde{\mathbf{w}}_m^{(t_m)}\right\|^2 \leq 4C\frac{(\eta^{(t)})^2}{\gamma^2}G^2H^2. \tag{41}$$

Putting the bounds from (89), (92), and (97) in (88) and using $B=(4-2\gamma)$ give

$$\mathbb{E}\left\|\widehat{\mathbf{w}}^{(t)}-\widetilde{\mathbf{w}}^{(t)}\right\|^2 \leq 192(4-2\gamma)\left(1+\frac{C}{\gamma^2}\right)(\eta^{(t)})^2H^4G^2+12C\frac{(\eta^{(t)})^2}{\gamma^2}G^2H^2. \tag{42}$$

This completes the proof of Lemma 3.

4. The design of a learning-driven control algorithm

The problem defined in (5) under constraints (6a), (6b) and (6c) is a mixed integer programming problem, which is in general NP-hard. Additionally, the communication channel selection and resource allocation strive to mini-mize (5), which also can be reformulated as a Markov decision process [17,18]. As such, we utilize the recent advances in machine learning [19] to design a control mechanism using deep reinforcement learning (DRL) algorithm for local update adaption and communication channel selection associated with its wireless connectivity and resource constraints. In this section, we propose a learning-driven control algorithm for Rainbow to achieve resource-efficient FL. We first introduce the workflow of the DRL algorithm and then describe how to solve the formulated problem with a DRL procession.

4.1. Deep reinforcement learning mechanism

Unlike some traditional approaches using predefined rules or model-based heuristics, the DRL-driven method aims to learn a general action set based on the current system state and the given reward. This is critical for deploying Rainbow in a highly dynamic environment.

The workflow of the DRL method is illustrated in Fig. 2. At each epoch t, for each device m, it measures its state $\mathbf{s}_m^{(t)}$, computes the corresponding reward $r_m^{(t)}$, and chooses its action $a_m^{(t)}$ based on its policy $\pi_m^{(t)}$. After device m updates its state to $\mathbf{s}_m^{(t+1)}$ at the next epoch $t+1$, it puts the tuple $\langle \mathbf{s}_m^{(t)}, \mathbf{a}_m^{(t)}, r_m^{(t)}, \mathbf{s}_m^{(t+1)}\rangle$ in a replay buffer for experience accumulation. A critic-network then reads from the replay buffer and updates the policy to $\pi_m^{(t+1)}$ together with the optimizer. In particular, $\pi_m^{(t+1)}$ is updated to maximize the accumulative rewards $\mathbf{R}_m^{(t)}=\sum_{t=0}^{\infty}\gamma_m^{(t)}r_m^{(t)}$, where $\gamma\in(0,1]$ is a discount factor of future rewards.

4.2. Model design

To implement the formulated problem using DRL techniques, we specify the state space, the action space, and the reward function below.

4.2.1. State space

The state of each agent contains the current resource consumption of each type of resource. We denote the state space $\mathcal{S}_m=\{\mathbf{s}_m^{(t)},\forall t\in\mathcal{T}\}$. And we define $s_m^{(t)}$ as follows

$$s_m^{(t)}=\langle \mathbf{E}_{m,\text{comm}}^{(t)}, \mathbf{E}_{m,\text{comp}}^{(t)}\rangle, \tag{43}$$

where

$$\mathbf{E}_{m,\text{comm}}^{(t)}=\langle E_{m,1,\text{comm}}^{(t)},\cdots,E_{m,r,\text{comm}}^{(t)},\cdots,E_{m,R,\text{comm}}^{(t)}\rangle, \tag{44a}$$

$$\mathbf{E}_{m,\text{comp}}^{(t)}=\langle E_{m,1,\text{comp}}^{(t)},\cdots,E_{m,r,\text{comp}}^{(t)},\cdots,E_{m,R,\text{comp}}^{(t)}\rangle. \tag{44b}$$

The state variables are described as follows.

- $E_{m,r,\text{comm}}^{(t)}$ represents the consumption factor for communication of resource r at device m in round t.
- $E_{m,r,\text{comp}}^{(t)}$ represents total consumption for local computation of resource r at device m in round t.

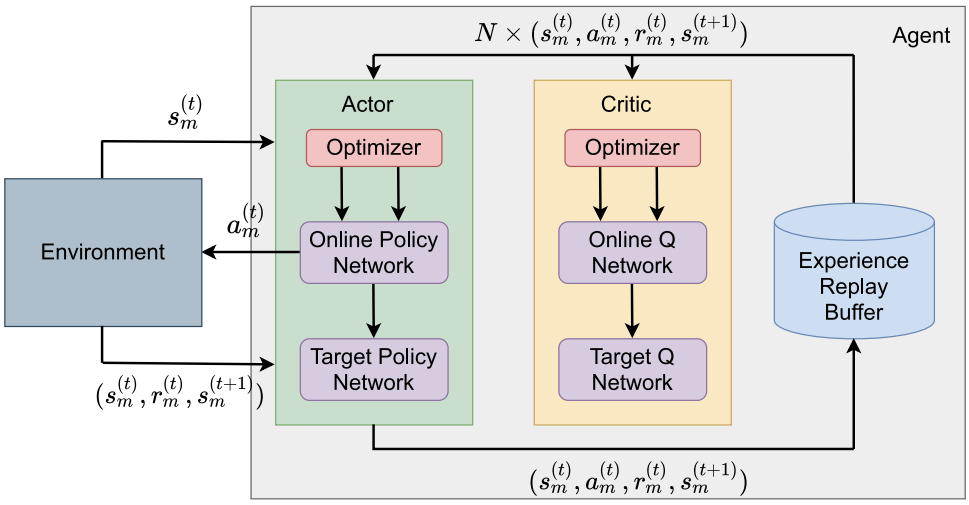


Fig. 2. The workflow of the DRL algorithm.

4.2.2. Action space

Each device m has an action space denoted as $\mathcal{A}_m = \left\{\mathbf{a}_m^{(t)}, \forall t \in \mathcal{T}\right\}$. On receiving state $s_m^{(t)}$, the agent m needs to choose its local computation and communication decisions. Specifically, an action can be represented as

$$a_m^{(t)} = \langle H_m^{(t)}, \mathbf{D}_m^{(t)} \rangle, \tag{45}$$

where $\mathbf{D}_m^{(t)} = \langle D_{m,1}^{(t)}, \ldots, D_{m,n}^{(t)}, \ldots, D_{m,N}^{(t)} \rangle$.

The action variables are described as follows.

- $H_m^{(t)}$ represents the number of local iterations at device m in round t.
- $D_{m,n}^{(t)}$ represents the number of gradient entries sent through channel n at device m in round t.

4.2.3. Reward function

At each training epoch t, the agent m will get a reward $r\langle s_m^{(t)}; a_m^{(t)}; s_m^{(t+1)} \rangle$ under a certain state $s_m^{(t)}$ after executing action $a_m^{(t)}$. For the sake of clarity, we denote global loss function $f(\mathbf{w}^{(t)})$ as $\varepsilon^{(t)}$ and local loss function $f(\mathbf{w}_m^{(t)})$ as $\varepsilon_m^{(t)}$. The objective function of this work is to minimize the global loss function $\varepsilon^{(T)} = \sum_{m=1}^{M} \varepsilon_m^{(T)}$ under resource constraints. Hence, we minimize the loss function denoted by $\varepsilon_m^{(T)}$ for each device m under its resource constraints. We first define the utility function over resource r at device m in iteration t as follows:

$$U_{m,r}^{(t)} = \frac{\delta^{(t)}}{\epsilon_{m,r}^{(t)}}, \tag{46}$$

where

$$\delta_m^{(t)} = \varepsilon_m^{(t)} - \varepsilon_m^{(t-1)}, \tag{47a}$$

$$\epsilon_{m,r}^{(t)} = \left(E_{m,r,comp}^{(t)} H_m^{(t)} + \sum_{n=1}^{N} E_{m,r,comm}^{(t)} D_{m,n}^{(t)} \right). \tag{47b}$$

Then we define the reward function as the weighted averaging utility function over R types of resources at device m in iteration t as follows:

$$r\langle s_m^{(t)}; a_m^{(t)}; s_m^{(t+1)} \rangle = \sum_{r=1}^{R} \alpha_r \frac{U_{m,r}^{(t+1)}}{U_{m,r}^{(t)}}, \tag{48}$$

where α_r is the weight of utility function $U_{m,r}^{(t)}$.

4.3. DRL algorithm details

In our framework, each device dynamically decides its number of local iterations, gradient compression ratio and traffic allocation among different channels based on the state-of-the-art Deep Deterministic Policy Gradient (DDPG) algorithm [20]. Specifically, the algorithm maintains a parameterized critic function and actor function. As shown in Fig. 2, the critic function $Q(\mathbf{s}_i(t), \mathbf{a}_i(t)|\theta_i^Q(t))$ is implemented by a Deep Q-Network (DQN) where $\theta_i^Q(t)$ denotes the weight vector of DQN. The actor function $\pi(\mathbf{s}_i(t)|\theta_i^\pi(t))$ is implemented by DNN where $\theta_i^\pi(t)$ is the weight vector of the DNN. If the agent under state $\mathbf{s}_i(t)$ takes an action $\mathbf{a}_i(t)$ at epoch t, the Q value of the critic function will be returned as follows.

$$Q(\mathbf{s}_i(t), \mathbf{a}_i(t)) = \mathbb{E}\left[\mathbf{R}_i | \mathbf{s}_i(t), \mathbf{a}_i(t)\right], \tag{49}$$

where $\mathbf{R}_i = \sum_{k=t}^{T} \gamma_i(t) r_i\langle \mathbf{s}_i(t), \mathbf{a}_i(t) \rangle$. Let $y_i(t)$ be the target value at epoch t. It can be evaluated as

$$y_i(t) = r_i\langle \mathbf{s}_i(t), \mathbf{a}_i(t) \rangle + \gamma_i(t) Q(\mathbf{s}_i(t+1), \pi(\mathbf{s}_i(t+1)|\theta_i^\pi(t))|\theta_i^Q(t)), \tag{50}$$

where $\gamma_i(t)$ denotes the discount factor for future rewards at edge device i at epoch t. According to [21], the actor-network can be updated by applying the chaining rule to the expected cumulative reward $\mathbf{R_i(t)}$ with respect to the actor vector θ_i^π.

$$\nabla_{\theta^\pi} \mathcal{R}_i = \mathbb{E}(\nabla_{\mathbf{a}} Q(\mathbf{s}_i, \mathbf{a}_i|\theta^Q)|_{\mathbf{s}_i=\mathbf{s}_i(t), \mathbf{a}_i=\pi(\mathbf{s}_i(t))} \cdot \nabla_{\theta^\pi} \pi(\mathbf{s}|\theta^\pi)_{\mathbf{s}_i=\mathbf{s}_i(t)}), \tag{51}$$

where $\pi(\mathbf{s}_i(t)) = \arg\max_{\mathbf{a}_i(t)} Q(\mathbf{s}_i(t), \mathbf{a}_i(t))$ and stands for the greedy policy of the off-policy algorithm.

Going beyond the original DDPG, which applies a uniform sampling method for the experience replay and hence ignores the significance of transition samples in the replay buffer, we adopt a prioritized experience replay technique under a critic–actor framework. In the replay buffer, transition tuples are marked with different priorities to achieve better performance. During each epoch, a tuple sampled by a priority-based method will be obtained to compute its temporal difference error (TD-Error), denoted as ε, using the critic function. For a tuple with index j, with state $\mathbf{s}_i\langle t; j \rangle$, and action $\mathbf{a}_i\langle t; j \rangle$, the TD-Error function can be calculated as

$$\varepsilon_i\langle t; j \rangle = y_i\langle t; j \rangle - Q(\mathbf{s}_i\langle t; j \rangle, \mathbf{a}_i\langle t; j \rangle), \tag{52}$$

where $y_i\langle t; j \rangle$ is defined by Eq. (50) with the j tuple. For simplicity, let $\nabla_{\mathbf{a}} Q_i\langle t; j \rangle = \nabla_{\mathbf{a}} Q(\mathbf{s}, \mathbf{a})|_{\mathbf{s}=\mathbf{s}_i\langle t;j \rangle, \mathbf{a}=\pi(\mathbf{s}_i\langle t;j \rangle)}$ according to Eq. (51). Combining with its absolute value $\left|\nabla_{\mathbf{a}} Q_i\langle t; j \rangle\right|$, we can define the priority of a given tuple as follows.

$$p_i(t; j) = \varrho \cdot \left|\varepsilon_i\langle t; j \rangle\right| + (1 - \varrho) \cdot \left|\nabla_{\mathbf{a}_i(t)} Q\langle t; j \rangle\right|, \tag{53}$$

where ϱ is a constant controlling the relative importance of TD-Error versus that of the Q gradient. For any tuple j in the replay buffer, its sampling probability is then

$$\Pr_i\langle t; j\rangle = \frac{\left(p_i\langle t; j\rangle\right)^x}{\sum_{n=1}^{H}\left(p_i\langle t; n\rangle\right)^x}, \tag{54}$$

where H is the size of the replay buffer and the exponent x plays the role of the sampling distribution. The importance-sampling weight of tuple j is then computed as

$$w_i\langle t; j\rangle = \frac{\left(H \cdot \Pr_i\langle t; j\rangle\right)^{-x}}{\max\left\{w_i\langle t; 1\rangle, \ldots, w_i\langle t; H\rangle\right\}}, \tag{55}$$

and the accumulated weight changes for the critic and actor-network are computed as

$$\Delta_{\theta_i^Q(t)} = \Delta_{\theta_i^Q(t)} + w_i\langle t; j\rangle\varepsilon_i\langle t; j\rangle\nabla_{\theta_i^Q(t)}Q(\mathbf{s}_i\langle t; j\rangle, \mathbf{a}_i\langle t; j\rangle), \tag{56a}$$

and

$$\Delta_{\theta_i^\pi(t)} = \Delta_{\theta_i^\pi(t)} + w_i\langle t; j\rangle\nabla_{\theta_i^\pi(t)}Q(\mathbf{s}_i\langle t; j\rangle, \mathbf{a}_i\langle t; j\rangle). \tag{56b}$$

The procedure of DRL-based control algorithm is summarized Algorithm 2. All computations in Algorithm 2 are input–output processes, giving each line the computational complexity of $\mathcal{O}(1)$. Due to our experience replay mechanism, the $\Delta_{\theta_i^Q(t)}$ and $\Delta_{\theta_i^\pi(t)}$ of each round will be accumulated multiple times until every sample in the replay buffer is being computed. Thus, the time complexity of Algorithm 2 is $\mathcal{O}(HT)$. The DRL control algorithm needs space to store actor network, critic network and reply buffer, so its space complexity is $\mathcal{O}(n)$. Notice the LGC with DRL-based control algorithm is a practical improved version of LGC, which means it shares the same convergence result with LGC.

Algorithm 2 The DRL-based algorithm

1: **Initialize:** Critic function $Q_i(0)$ with weight $\theta_i^Q(0)$ and actor function with weight $\theta_i^\pi(0)$. Target function $Q_i'(0)$, $\pi_i'(0)$ with $\theta_i^{Q'}(0) = \theta_i^Q(0)$, and $\theta_i^{\pi'}(0) = \theta_i^\pi(0)$. The size of replay buffer H, $p_i(0; 1) = 1$, and total episodes T. $\Delta_{\theta_i^Q(0)} = \Delta_{\theta_i^\pi(0)} = 0$.
2: **for** $t = 1$ to T **do**
3: Get the state $\mathbf{s}_i(t)$ and take an action $\mathbf{a}_i(t)$.
4: Feedback $\mathbf{s}_i(t+1)$, compute $r_i(t)$ via Equation (46).
5: Put the tuple $\langle \mathbf{s}_i(t), \mathbf{a}_i(t), r_i(t), \mathbf{s}_i(t+1)\rangle$ into the replay buffer.
6: **for all** $j = 1$ to H **do**
7: Sample a tuple j accord with Equation (54).
8: Compute importance-sampling of tuple j by Equation (55).
9: Calculate TD-Error of tuple j via Equation (52).
10: Compute the expected cumulative by Equation (51).
11: Update the priority of tuple j by Equation (53).
12: Accumulate the weight changes of critic and actor functions via Equation (56a) and (56b).
13: Update the critic and actor function, and their target functions.
14: Reset $\Delta_{\theta_i^Q(t)}$, $\Delta_{\theta_i^\pi(t)}$ to zero.

5. Evaluation

We describe the implementation of Rainbow and verify its performance in this section. We first clarify our environment settings and then show the experimental results.

5.1. Experiment settings

5.1.1. Environment

We conduct the experiments in a distributed environment using two servers with GPUs (a server with 4 GeForce RTX 2080 Ti and a server with 8 T T4). We implement the framework with FedML [22] and set the number of devices to 32. We put devices on these GPUs, except one for the placement of the server. We simulated an wireless FDMA communication environment with three channels, including 3G, 4G and 5G channels, with data rates of 2 Mbps, 500 Mbps and 1 Gbps respectively [23].

5.1.2. Baseline

To illustrate the effectiveness of Rainbow (LGC with DRL), we implement LGC with DRL control algorithm and compare Rainbow with the following FL baselines.

- FedSGD [24] with single-channel communication performs a fixed number of local computations in each round and aggregates the gradients in a centralized and synchronous paradigm with a single communication channel.
- FedSGD with multiple channel communication performs a fixed number of local computations in each round and aggregates the gradients in a centralized and synchronous paradigm with multiple communication channels.
- LGC without DRL performs a fixed number of local computations and compresses the gradients with the same communication decisions for each round.

We use these methods as baselines to show the benefits of Rainbow.

5.1.3. Datasets and models

The experiments are conducted over three different models (*i.e.*, LR [25], CNN [26] and LSTM [27] which are implemented by open source framework FedML [22]) and three real datasets (*i.e.*, MNIST, CIFAR10, and Shakespeare).

- LR is trained over the MNIST dataset [28], which is composed of 60,000 handwritten digits for training and 10,000 for testing.
- CNN is trained over the CIFAR10 dataset [29], which consists of 50,000 training images and 10,000 validation images in 10 classes.
- LSTM is trained over the Shakespeare dataset [22], which includes 40,000 lines from a variety of Shakespeare's plays.

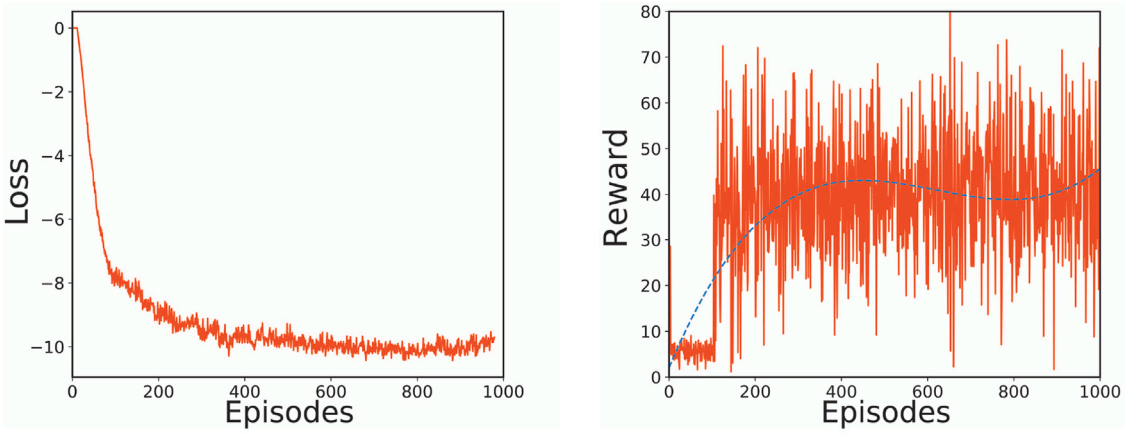
5.1.4. Performance metrics

In our experiments, we mainly adopt the following metrics to evaluate the performance of our proposed framework.

- **Training loss** measures the difference between the predicted values and the actual values. The performance of both the DRL and the training models are all evaluated.
- **Reward** is the return of the reward function during one DRL training episode.
- **Model accuracy** is the proportion of correctly classified samples to all samples in the dataset.
- **Energy consumption** caused by local computation and communication, which indicates the battery usage.
- **Money cost** denotes the money spent on the training procedure.
- **Time consumption** denotes the time spent until training terminates, which is used to evaluate the model training speed.
- **Communication traffic** denotes the volume of gradients during the model training.

5.1.5. Hyperparameters setting

Our first application is an LR on the MNIST dataset, where we set the training batch size to 128, learning rate to 0.01, and momentum to 0.9. In the second application, we train CNN using the CIFAR10 dataset, with the batch size of 128, learning rate of 0.001, momentum of 0.9, and weight decay 1e−5. The third application is an LSTM model with the Shakespeare dataset, where we set the batch size to 64, learning rate to 0.01, and momentum to 0.9. For each application, we randomly partition its training dataset across devices with the same random seed.



(a) Loss (b) Reward

Fig. 3. Convergence curves of DRL training.

Table 2
Energy consumption for different communication channels.

Channel type	Mean (J/MB)	Standard deviation
3G	1296	0.033
4G	2.2×1296	0.033
5G	$2.5 \times 2.2 \times 1296$	0.033

By default, we set the initial compression ratio to 50× and consider three different communication channels (3G, 4G, and 5G) for Rainbow. To quantify the energy cost for different channels, we adopt a Gaussian distribution with mean and standard deviation values [8]. The energy consumption ϵ at each channel is given in Table 2.

5.2. Experiment results

We set a fixed random seed to ensure our experimental results are reproducible, and all the results in the following figures are the average of multiple repeated experiments.

5.2.1. Results of DRL training

The DRL training is conducted simultaneously with the FL procedure. In Fig. 3(a), we first observe the loss change with the increasing episode in DRL. The loss decreases quickly in the earlier stages of DRL training because the DRL agent has no information about network conditions and the FL training leads to a high training loss. Thanks to the efficient exploration and the experience replay, the reward will rapidly decrease with the model training procession. Fig. 3(b) shows the change of reward.The blue curve donates the smoothed curve of the reward value. Specifically, as the smoothed curve shows the reward value increases with the epoch because the DRL model can learn a better policy to achieve a better reward.

5.2.2. Results of model performance

We compare LGC with/without DRL to FedSGD (single/multiple channels) with different datasets and models. We first examine on the image classification task. Figs. 4(a) and 4(b) are the training loss and test accuracy of LR on MNIST dataset. The convergence curves of FedSGD with single-channel communication (red) coincide with the one of FedSGD with multiple channel communication (green) as all gradients aggregated by multi-channel FedSGD is exactly the same as that of single-channel FedSGD and as we use the same random seed to generate the same dataset partitioning, gradient update and same initial model parameters With LGC (orange, blue), the convergence curves closely follow the ones of FedSGD (green, red).

Table 3
Energy consumption for different communication channels.

Model	Method	Loss	Acc.
LR	LGC (with DRL)	1.723	83.62%
	LGC (without DRL)	1.720	83.61%
	FedSGD (single channel)	1.718	83.62%
	FedSGD (multi channel)	1.718	83.62%
CNN	LGC (with DRL)	0.678	79.21%
	LGC (without DRL)	0.677	79.22%
	FedSGD (single channel)	0.678	79.23%
	FedSGD (multi channel)	0.678	79.23%
LSTM	LGC (with DRL)	1.612	55.02%
	LGC (without DRL)	1.689	54.21%
	FedSGD (single channel)	1.553	52.54%
	FedSGD (single channel)	1.553	52.54%

When scaling to the large-scale dataset, Figs. 5(a) and 5(b) show the convergence curves of CNN on CIFAR10 dataset. The convergence curves of FedSGD with single-channel communication (red) fully match the ones of FedSGD with multiple channel communication (green) as LR on MNIST. The training loss and test accuracy of LGC (orange and blue) closely match the ones of FedSGD (green and red).

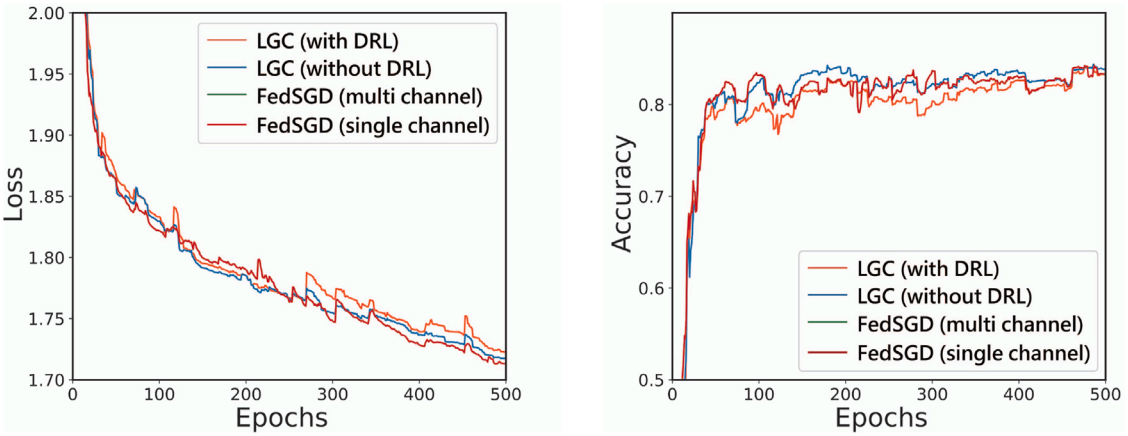
Figs. 6(a) and 6(b) show the training loss and test accuracy of LSTM on the Shakespeare dataset. The learning curve of LGC without DRL (blue) is worse than FedSGD (red, green) due to gradient sparsity. With DRL (orange), the learning curve converges slightly faster, and the accuracy is much closer to FedSGD.

Table 3 summarizes the best training loss and test accuracy of the LR, CNN, and LSTM datasets. We can find that LGC convergences with a similar rate to FedSGD and LGC in Rainbow has little impact on convergence and accuracy.

5.2.3. Results of resource consumption

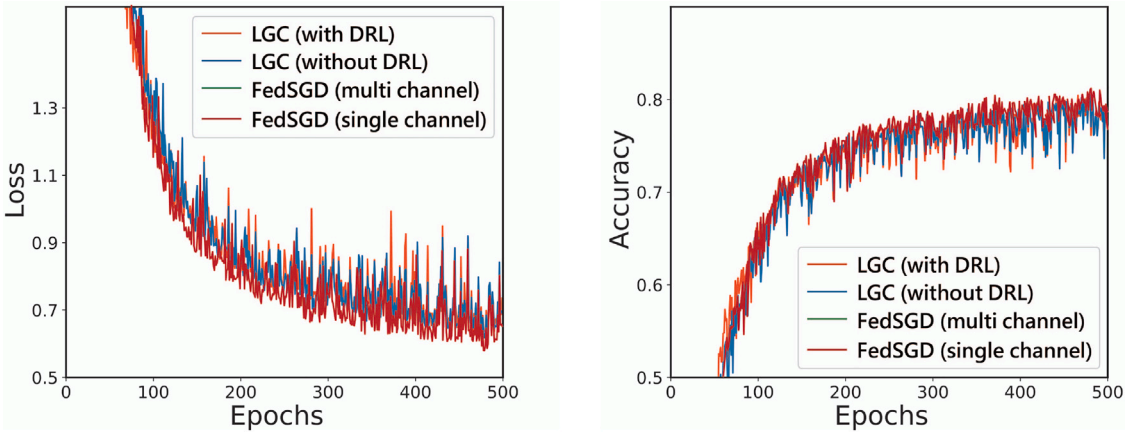
We also compare Rainbow to baselines with energy consumption and money cost. We calculate the cost of communication using the product of the amount of communication data and the cost per unit of communication data. We refer the average cost for 1 GB data on 3G is $25 compared to 4G on $17 and $13 on 5G [30]. By the results from Figs. 7, 8 and 9, Rainbow can greatly reduce the energy consumption and money cost when achieving the target accuracy for LR on MNIST, CNN on CIFAR10, and LSTM on Shakespeare, respectively.

Multiple channel communication (orange, blue, green) is more resource-efficient than single-channel communication (red) since slower channels (e.g., 3G and 4G) can save energy and money costs



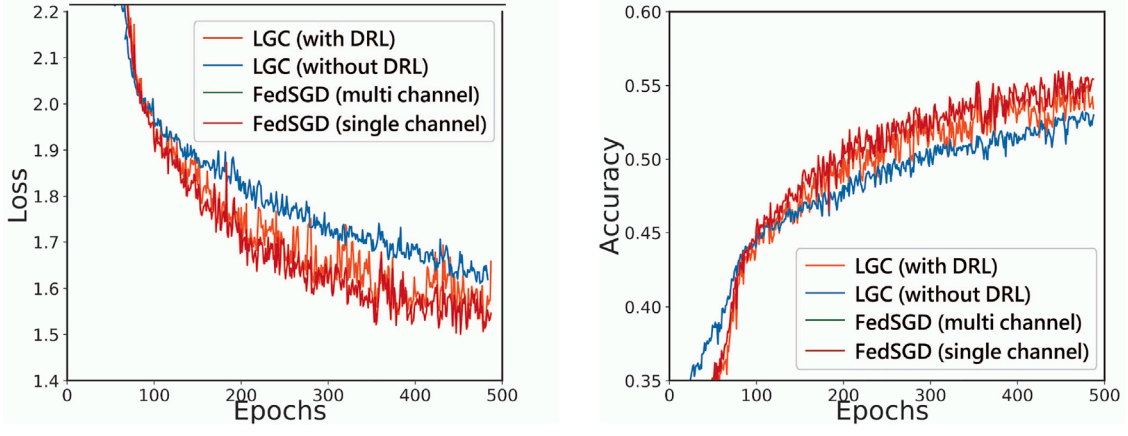
(a) Loss (b) Accuracy

Fig. 4. Convergence curves of different mechanisms with LR on MNIST. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



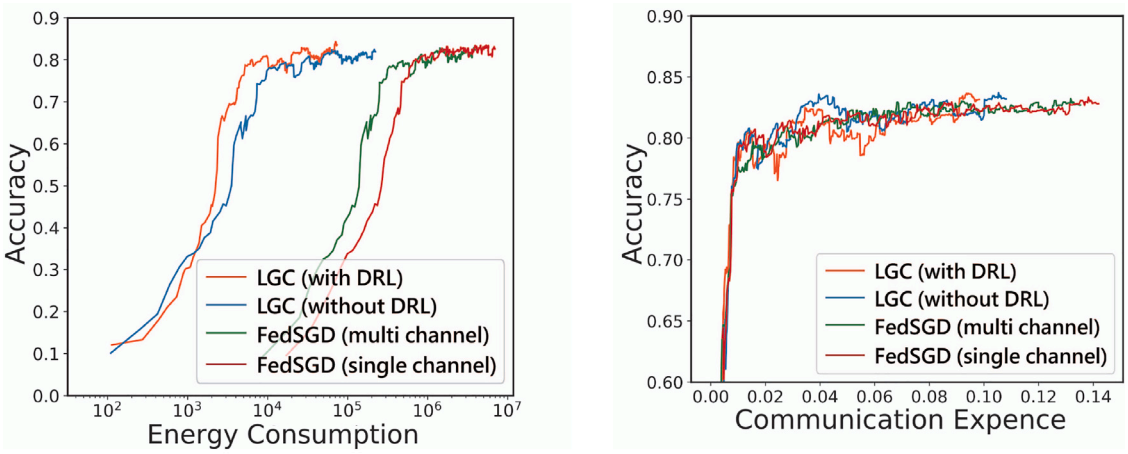
(a) Loss (b) Accuracy

Fig. 5. Convergence curves of different mechanisms with CNN on CIFAR10. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



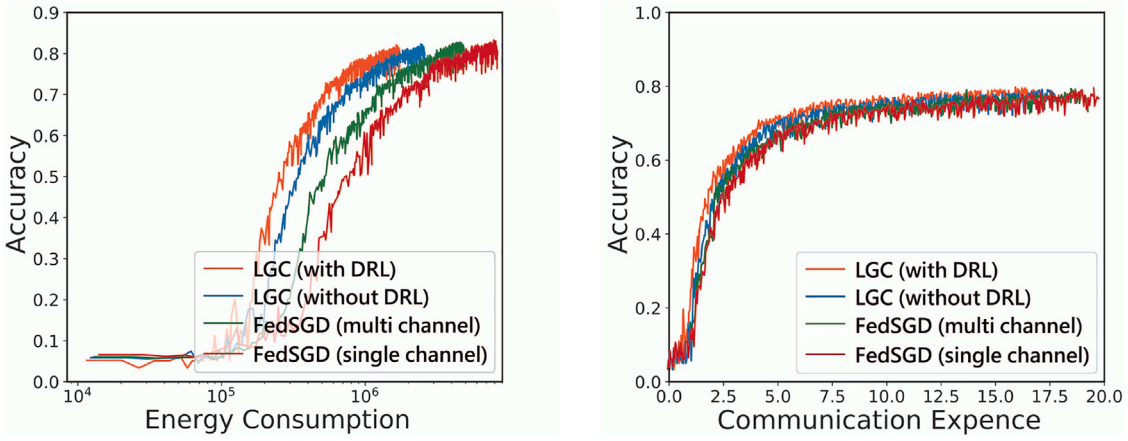
(a) Loss (b) Accuracy

Fig. 6. Convergence curves of different mechanisms with LSTM on Shakespeare. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



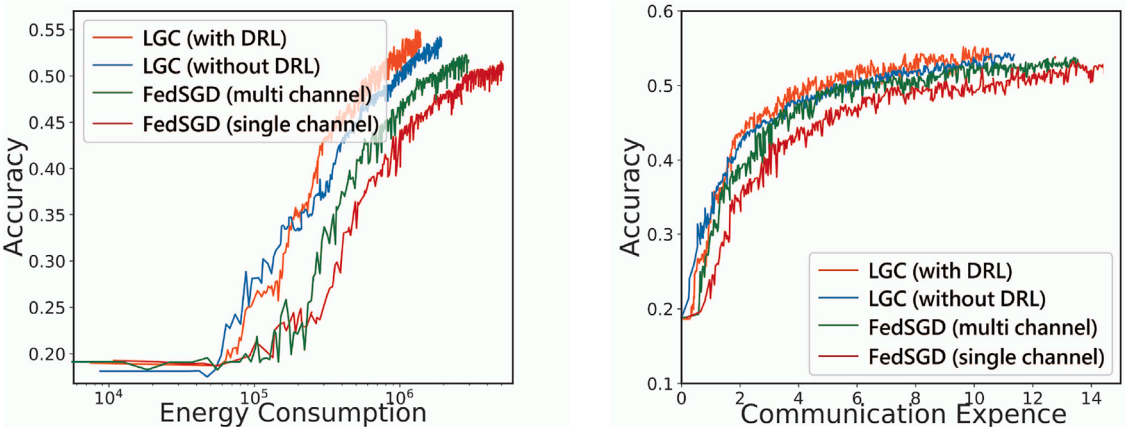
(a) Energy consumption (b) Monetary cost

Fig. 7. Resource Consumption of different mechanisms with LR on MNIST.



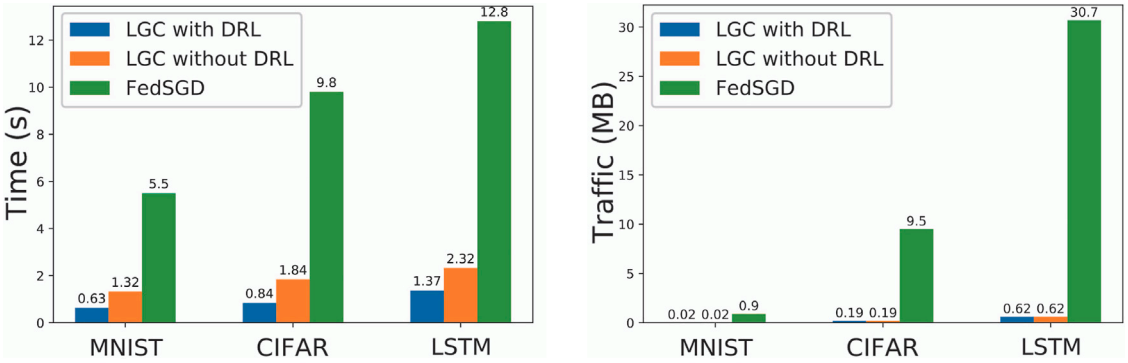
(a) Energy consumption (b) Monetary cost

Fig. 8. Resource Consumption of different mechanisms with CNN on CIFAR10.



(a) Energy consumption (b) Monetary cost

Fig. 9. Resource Consumption of different mechanisms with LSTM on Shakespeare.



(a) Time (b) Communication

Fig. 10. Time and communication comparisons of one iteration.

instead of faster channels (e.g., 5G). The reason for the significant performance improvement of LGC (orange, blue) under the resource budgets is that LGC performs gradient compression, which significantly reduces the network traffic. DRL (orange) further improves resource utilization since the DRL-driven control algorithm makes Rainbow dynamically adjust its local computation and communication decisions.

Fig. 10 depicts the iteration time and the network footprint of one iteration under different mechanisms. We measure the actual iteration time to calculate the time communication. The iteration time includes computation time and communication time of edge devices. LGC successfully reduces the network traffic by order of magnitude, which is beneficial to saving energy consumption and communication expense. LGC also saves the training time due to gradient compression. With DRL, Rainbow further reduces the training time because the DRL-driven control algorithm makes Rainbow dynamically adjust its local computation and communication decisions.

5.2.4. Varying number of devices

We use the CNN model and CIFAR10 dataset to analyze the effect of the number of devices on the training results. Figs. 11(a) and 11(b) show the training loss and test accuracy. Figs. 11(c) and 11(d) show the energy consumption and money cost. The convergence curves and resource curves of the different methods are very close. Results show that varying number of devices has little impact on the model performance and resource utilization.

5.2.5. Varying number of channels

We use the LSTM model and Shakespeare dataset to analyze the effect of the number of channels on the training results. Figs. 12(a) and 12(b) show the training loss and test accuracy. The convergence curve of Rainbow with three channels (orange) is the best, and the convergence curve of Rainbow with one channel (green) is the worst. The convergence curve of variant channels (red) (each device with different number of communication channels (1/2/3)) is slightly worse than that of three channels (orange) and better than that of two channels (blue). Figs. 12(c) and 12(d) show the energy consumption and money cost. The resource curve of Rainbow with three channels (orange) is the best, and the resource curve of Rainbow with one channel (green) is the worst. The resource curve of variant channels (red) is slightly worse than that of three channels (orange) and better than that of two channels (blue).

6. Related works

The unique characteristics of FL in dynamic networks lead to mainly practical issues in FL implementation, i.e., (i) communication cost, (ii) resource allocation, and (iii) decision making. In this section, we review related works that address each of these issues.

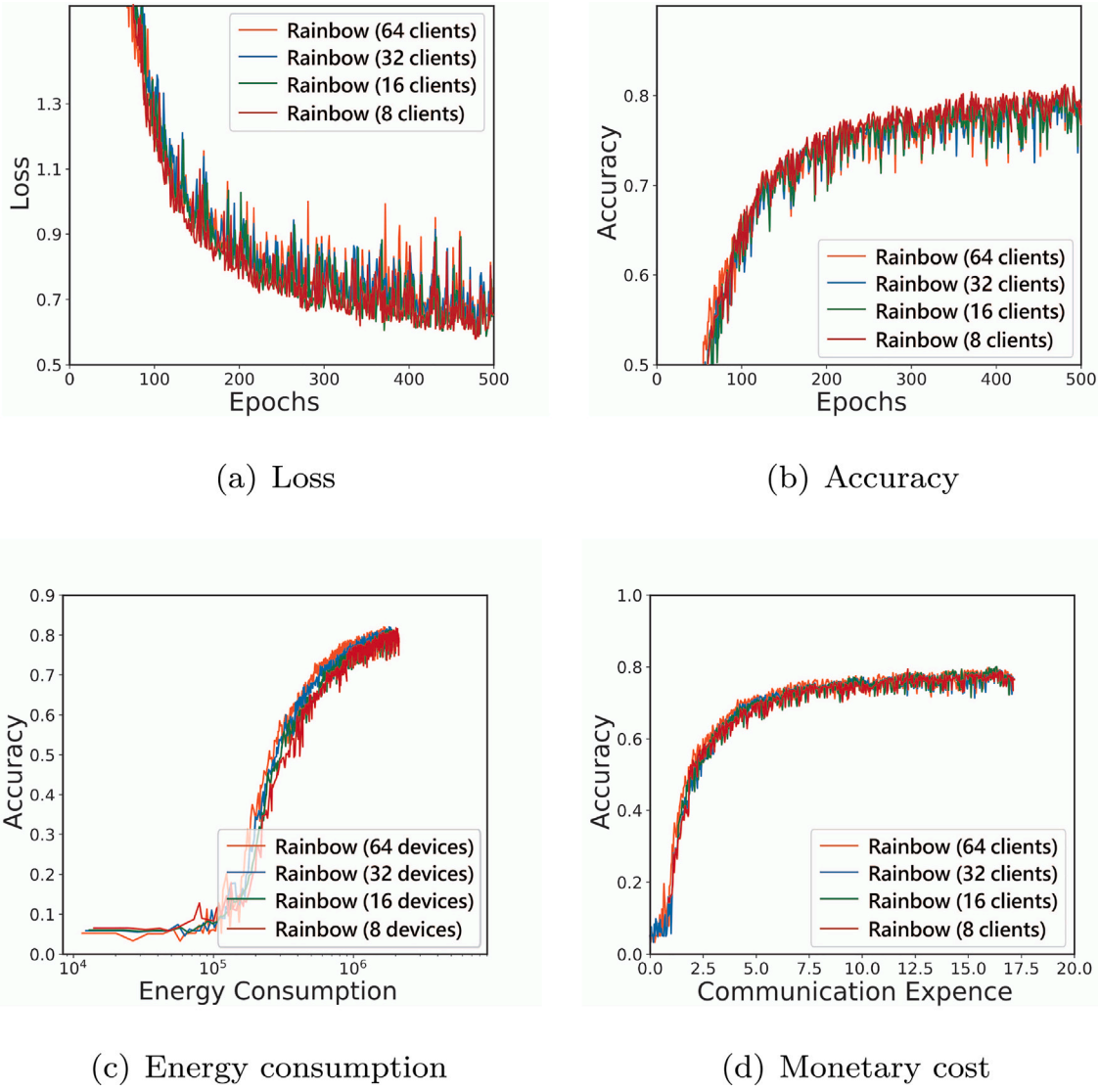
6.1. Communication cost

6.1.1. Local computation

Some recent works propose to perform more computation on edge devices before each global aggregation to reduce the number of communication rounds needed for the model training [24,31,32]. [24] proposed to do more local updates between communication for global aggregation to reduce instances of communication. [31] transferred a learning-inspired two-stream model for FL participants to learn from the fixed global model to accelerate training convergence. [32] introduced MEC-inspired edge server-assisted FL that aids in intermediate parameter aggregation to reduce instances of communication. However, these approaches may increase computation costs and delay convergence when global aggregation is infrequent. The tradeoff between these sacrifices and communication cost reduction thus has to be well-managed.

6.1.2. Gradient compression

To reduce the network traffic per round, some other works let each participant communicate the compressed gradients rather than original gradients for every global synchronization by quantization [33,34] or sparsification [11,12,35]. Quantization-based methods limit the number of bits to represent floating-point numbers. QSGD [34] and TernGrad [33] compresses gradients into multi-levels and 3-levels respectively with stochastic quantization to ensure that the compressed stochastic gradient remains an unbiased approximation to the true gradient. 1BitSGD [36] was proposed to utilize only the sign of each gradient component. An error feedback scheme was introduced during quantization to compensate for the quantization error from the last iteration. Initiated by a 1BitSGD, a large number of recent studies revisited the idea of biased quantization [37–40]. Sparsification-based methods reduce the number of non-zero entries in the stochastic gradient. An unbiased stochastic sparsification method was proposed in [41] that balances sparsity and variance by solving constrained linear programming. Another Top-k sparsification schemes keep only very few coordinates of the stochastic gradient by considering only the coordinates with the largest magnitudes [42–45]. In contrast to the unbiased schemes, it is clear that such methods can only work by using some kind of error feedback or accumulation mechanisms [11,45,46], as otherwise, certain coordinates could simply never be updated. This category is the closest to our work. However, we note that these proposed sparsification methods are not a general design for FL with layered gradient compression.



(a) Loss (b) Accuracy

(c) Energy consumption (d) Monetary cost

Fig. 11. Scalability in number of devices with CNN on CIFAR10.

6.2. Resource allocation

6.2.1. Adaptive aggregation

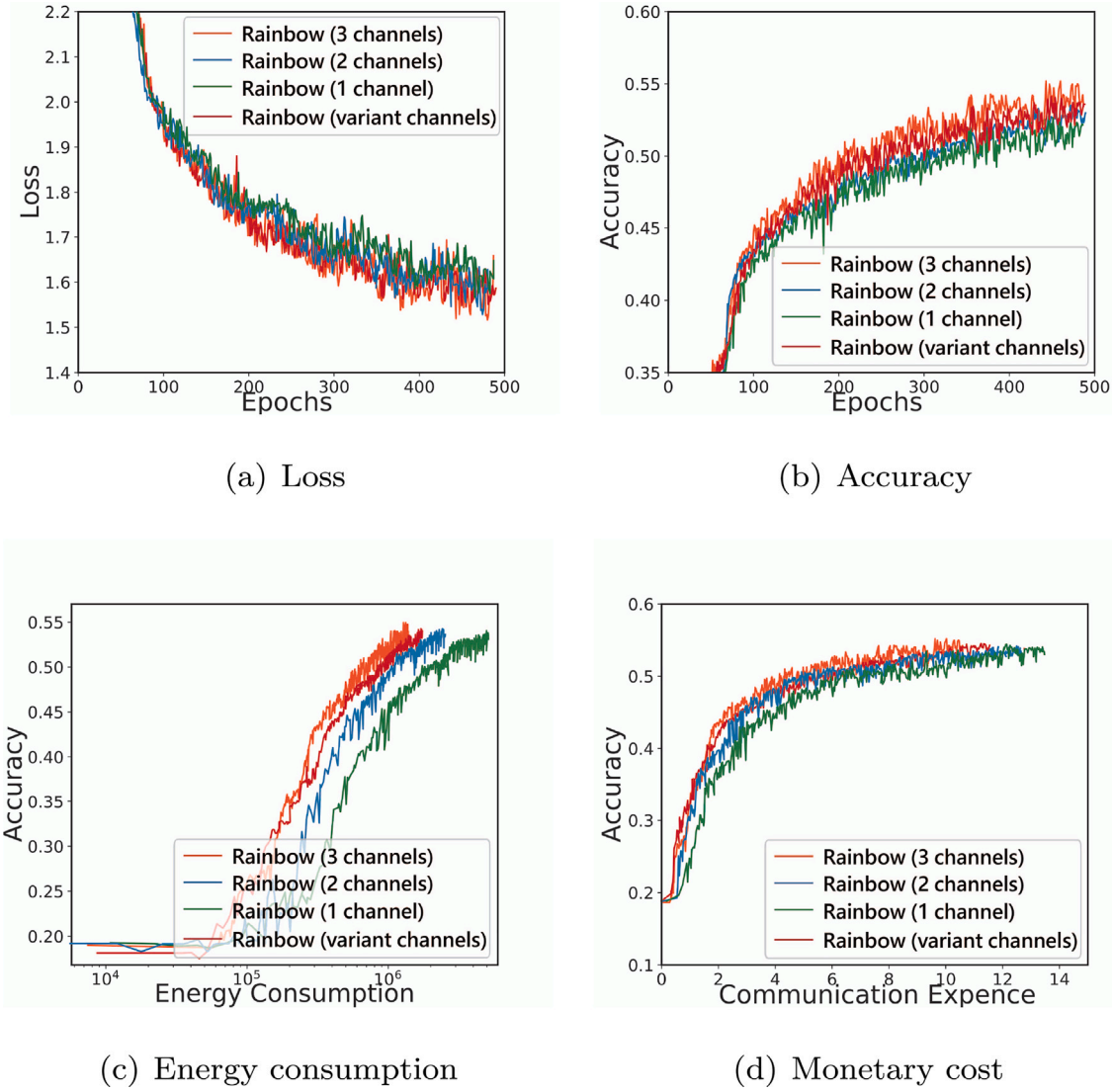
In recent works, adaptive adjustment of global aggregation frequency has been investigated to increase training efficiency subject to resource constraints [8,47]. [8] proposed to use adaptive global aggregation frequency based on resource constraints. [47] proposed asynchronous FL where model aggregation occurs whenever local updates are received by the FL server. [48] proposed a service delay reduction method over mobile devices by jointly selecting the number of local training iterations and gradient quantization levels. [49] proposed a 3-layer hierarchical FL time minimization model, minimize the hierarchical FL time by optimizing number of local computations and the number of local aggregations. While properly managing resources to enable FL in mobile edge networks, these studies ignore reducing resource consumption intrinsically by themselves. Moreover, they hinder the substantial boost in training efficiency and resource utilization.

6.2.2. Joint communication techniques and resource management

Even though computation capabilities of mobile devices have grown rapidly, many devices still face a scarcity of radio resources [50,51]. Some recent research has focused on using joint optimization to properly manage communication and computational resources [52–54]. [52] proposes a joint resolution considering energy efficiency, latency and accuracy trade-off for FL in Metaverse. [53,54] minimize the energy consumption in wireless powered edge computing systems by jointly optimizing time-slot assignment, computation task allocation, etc. [55] consider a joint optimization of local computation and network latency to optimize a weighted sum of energy consumption and total completion time. Given that local model transmission is an integral part of FL, there also has been a growing number of studies that focus on developing novel wireless communication techniques for efficient FL [56,57]. Yang et al. [58]consider the tradeoff between local computation delay and wireless transmission to minimize the global communication latencies in FL. However, signal distortion can lead to a drop in accuracy, and scalability is also an issue when large heterogeneous networks are involved. On a higher level, wireless technologies, such as IEEE 802.11a and 5G, provide multiple non-overlapping channels. The available network capacity can be increased by using multiple channels, and devices can be equipped with multiple interfaces to utilize the available channels.

6.3. Decision making

In recent years, DRL has shown potential for decision making in many applications. [59] first used Deep Q-Network to learn policies



(a) Loss (b) Accuracy

(c) Energy consumption (d) Monetary cost

Fig. 12. Scalability in number of channels with LSTM on Shakespeare. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

from sensor input. In this work, the authors proposed the experience replay and target network technologies to improve the stability and the performance. To reduce the observed over-estimations, [60] have designed the Double Deep Q-Network and achieved better performance on several games. [61], Schaul et al. 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 [21] and PPO [62]. Towards this direction, DeepRM [63] leveraged DRL to solve the online multi-resource job placement problem and can achieve better performance as compared with the heuristic algorithms. In [64], 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. [65] developed a novel model-free approach that can learn to well control a distributed stream data processing system from its experience rather than accurate and mathematically solvable system models. Liu et al. [66] leveraged a neural network-integrated convolutional neural network for feature extraction and then made the decision under the guidance of a multi-agent deep deterministic policy gradient method in a fully distributed mobile crowdsensing system. [67] proposed a FL framework utilizing DRL and distributed DRL to optimize edge caching and computing, which has advantages in terms of performance and cost balance. Different from these previous works, we consider the resource utilization in the FL and apply the DRL technique to accommodate a dynamic network environment resource-effectively.

7. Conclusion

This paper proposes a framework called Rainbow, which can reduce the training time and improve the resource utilization for multi-channel communication in MEC. Inspired by the layer coding technology in video streaming, the layered gradient compression and a learning-driven control algorithm are proposed. First, the local gradient from the device is encoded into multiple layers, and each layer is sent to the server along a different channel, and the server aggregates the local gradient received from the device to update the global model and send the results back to the device. Second, we propose a learning-driven control algorithm to adaptively adjust its local computation and communication decisions in each iteration. Finally, we evaluate the performance of the Rainbow framework. The results show that Rainbow significantly reduces training time and improves resource utilization while achieving similar model accuracy to compare with other baselines.

CRediT authorship contribution statement

Haizhou Du: Conceptualization, Methodology, Writing. **Yijian Chen:** Experiment setup, Editing, Revising. **Xiaojie Feng:** Data curation, Writing – original draft, Software. **Qiao Xiang:** Writing – review & editing. **Haoyu Liu:** Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work is supported in part by the NSFC Award 62172345, Open Research Projects of Zhejiang Lab 2022QA0AB05.

References

[1] J. Liu, H. Xu, Y. Xu, Z. Ma, Z. Wang, C. Qian, H. Huang, Communication-efficient asynchronous federated learning in resource-constrained edge computing, Comput. Netw. 199 (2021) 108429.
[2] H. Jiang, M. Liu, B. Yang, Q. Liu, J. Li, X. Guo, Customized federated learning for accelerated edge computing with heterogeneous task targets, Comput. Netw. 183 (2020) 107569.
[3] S. Niknam, H.S. Dhillon, J.H. Reed, Federated learning for wireless communications: Motivation, opportunities, and challenges, IEEE Commun. Mag. 58 (6) (2020) 46–51.
[4] D. Verma, S. Julier, G. Cirincione, Federated AI for building AI solutions across multiple agencies, 2018, arXiv preprint arXiv:1809.10036.
[5] S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, When edge meets learning: Adaptive control for resource-constrained distributed machine learning, in: IEEE Conference on Computer Communications, 2018, pp. 63–71.
[6] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. 10 (2) (2019) 1–19.
[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H.B. McMahan, et al., Towards federated learning at scale: System design, 2019, arXiv preprint arXiv:1902.01046.
[8] S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1205–1221.
[9] N.H. Tran, W. Bao, A. Zomaya, M.N. Nguyen, C.S. Hong, Federated learning over wireless networks: Optimization model design and analysis, in: IEEE Conference on Computer Communications, IEEE, 2019, pp. 1387–1395.
[10] M. Chen, H.V. Poor, W. Saad, S. Cui, Convergence time optimization for federated learning over wireless networks, IEEE Trans. Wireless Commun. 20 (4) (2020) 2457–2471.
[11] S.U. Stich, J.-B. Cordonnier, M. Jaggi, Sparsified SGD with memory, in: Advances in Neural Information Processing Systems, 2018, pp. 4447–4458.
[12] D. Basu, D. Data, C. Karakus, S. Diggavi, Qsparse-local-SGD: distributed SGD with quantization, sparsification and local computations, in: Advances in Neural Information Processing Systems, 2019, pp. 14668–14679.
[13] S.U. Stich, Local SGD converges fast and communicates little, 2018, arXiv preprint arXiv:1805.09767.
[14] M. Chen, Z. Yang, W. Saad, C. Yin, H.V. Poor, S. Cui, A joint learning and communications framework for federated learning over wireless networks, IEEE Trans. Wireless Commun. 20 (1) (2020) 269–283.
[15] H. Mania, X. Pan, D. Papailiopoulos, B. Recht, K. Ramchandran, M.I. Jordan, Perturbed iterate analysis for asynchronous stochastic optimization, SIAM J. Optim. 27 (4) (2017) 2202–2229.
[16] A. Rakhlin, O. Shamir, K. Sridharan, Making gradient descent optimal for strongly convex stochastic optimization, 2011, arXiv preprint arXiv:1109.5647.
[17] H. Xu, M. Chen, Z. Meng, Y. Xu, L. Wang, C. Qiao, Decentralized machine learning through experience-driven method in edge networks, IEEE J. Sel. Areas Commun. (2021).
[18] W. Zhang, D. Yang, W. Wu, H. Peng, N. Zhang, H. Zhang, X. Shen, Optimizing federated learning in distributed industrial IoT: A multi-agent approach, IEEE J. Sel. Areas Commun. 39 (12) (2021) 3688–3703.
[19] Y. Tang, S. Agrawal, Y. Faenza, Reinforcement learning for integer programming: Learning to cut, in: ICML, PMLR, 2020, pp. 9367–9376.
[20] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[21] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, 2014, pp. 387–395.
[22] C. He, S. Li, J. So, X. Zeng, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, et al., FedML: a research library and benchmark for federated machine learning, 2020, arXiv preprint arXiv:2007.13518.
[23] E. Ezhilarasan, M. Dinakaran, A review on mobile technologies: 3G, 4G and 5G, in: 2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM), IEEE, 2017, pp. 369–373.
[24] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[25] S.L. Gortmaker, Theory and methods–applied logistic regression by David W. Hosmer Jr and Stanley Lemeshow, Contemp. Sociol. 23 (1) (1994) 159.
[26] S. Albawi, T.A. Mohammed, S. Al-Zawi, Understanding of a convolutional neural network, in: 2017 International Conference on Engineering and Technology (ICET), IEEE, 2017, pp. 1–6.
[27] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[29] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, Handb. Syst. Autoimmune Dis. 1 (4) (2009).
[30] The cost of 5G technology: will it be expensive?, 2022, https://centum.com/en/the-cost-of-5g-technology-will-it-be-expensive/, Accessed: 2022-08-30.
[31] X. Yao, C. Huang, L. Sun, Two-stream federated learning: Reduce the communication costs, in: 2018 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2018, pp. 1–4.
[32] L. Liu, J. Zhang, S. Song, K.B. Letaief, Client-edge-cloud hierarchical federated learning, in: ICC 2020-2020 IEEE International Conference on Communications (ICC), IEEE, 2020, pp. 1–6.
[33] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, H. Li, TernGrad: ternary gradients to reduce communication in distributed deep learning, in: Advances in Neural Information Processing Systems, 2017, pp. 1509–1519.
[34] D. Alistarh, D. Grubic, J. Li, R. Tomioka, M. Vojnovic, QSGD: communication-efficient SGD via gradient quantization and encoding, in: Advances in Neural Information Processing Systems, 2017, pp. 1709–1720.
[35] J. Wangni, J. Wang, J. Liu, T. Zhang, Gradient sparsification for communication-efficient distributed optimization, in: Annual Conference on Neural Information Processing Systems, 2018, pp. 1306–1316.
[36] F. Seide, H. Fu, J. Droppo, G. Li, D. Yu, 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns, in: Annual Conference of the International Speech Communication Association, 2014, pp. 1058–1062.
[37] J. Wu, W. Huang, J. Huang, T. Zhang, Error compensated quantized SGD and its applications to large-scale distributed optimization, in: International Conference on Machine Learning, 2018, pp. 5325–5333.
[38] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, A. Anandkumar, SignSGD: Compressed optimisation for non-convex problems, in: International Conference on Machine Learning, 2018, pp. 559–568.
[39] J. Bernstein, J. Zhao, K. Azizzadenesheli, A. Anandkumar, SignSGD with majority vote is communication efficient and fault tolerant, 2018, arXiv preprint arXiv:1810.05291.
[40] S. Zheng, Z. Huang, J. Kwok, Communication-efficient distributed blockwise momentum SGD with error-feedback, in: Advances in Neural Information Processing Systems, 2019, pp. 11446–11456.
[41] J. Wangni, J. Wang, J. Liu, T. Zhang, Gradient sparsification for communication-efficient distributed optimization, in: Advances in Neural Information Processing Systems, 2018, pp. 1299–1309.
[42] N. Strom, Scalable distributed DNN training using commodity GPU cloud computing, in: Annual Conference of the International Speech Communication Association, 2015, pp. 1488–1492.
[43] A.F. Aji, K. Heafield, Sparse communication for distributed gradient descent, in: Conference on Empirical Methods in Natural Language Processing, 2017, pp. 440–445.
[44] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, K. Gopalakrishnan, AdaComp: adaptive residual gradient compression for data-parallel distributed training, in: AAAI Conference on Artificial Intelligence, 2018, pp. 2827–2835.

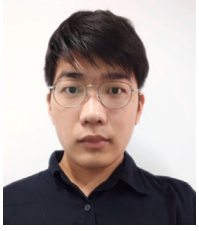
[45] Y. Lin, S. Han, H. Mao, Y. Wang, W.J. Dally, Deep gradient compression: Reducing the communication bandwidth for distributed training, 2017, arXiv preprint arXiv:1712.01887.
[46] S.P. Karimireddy, Q. Rebjock, S.U. Stich, M. Jaggi, Error feedback fixes signSGD and other gradient compression schemes, in: International Conference on Machine Learning, 2019, pp. 3252–3261.
[47] M.R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, M. Kopp, Asynchronous federated learning for geospatial applications, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2018, pp. 21–28.
[48] R. Chen, D. Shi, X. Qin, D. Liu, M. Pan, S. Cui, Service delay minimization for federated learning over mobile devices, 2022, arXiv preprint arXiv:2205.09868.
[49] C. Liu, T.J. Chua, J. Zhao, Time minimization in hierarchical federated learning, 2022, arXiv preprint arXiv:2210.04689.
[50] M.I. Jordan, J.D. Lee, Y. Yang, Communication-efficient distributed statistical inference, J. Amer. Statist. Assoc. (2018).
[51] Y. Wang, J. Zhao, Mobile edge computing, metaverse, 6G wireless communications, artificial intelligence, and blockchain: Survey and their convergence, 2022, arXiv preprint arXiv:2209.14147.
[52] X. Zhou, C. Liu, J. Zhao, Resource allocation of federated learning for the metaverse with mobile augmented reality, 2022, arXiv preprint arXiv:2211.08705.
[53] S. Mao, J. Wu, L. Liu, D. Lan, A. Taherkordi, Energy-efficient cooperative communication and computation for wireless powered mobile-edge computing, IEEE Syst. J. (2020).
[54] J. Feng, W. Zhang, Q. Pei, J. Wu, X. Lin, Heterogeneous computation and resource allocation for wireless powered federated edge learning systems, IEEE Trans. Commun. 70 (5) (2022) 3220–3233.
[55] X. Zhou, J. Zhao, H. Han, C. Guet, Joint optimization of energy consumption and completion time in federated learning, in: 2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS), IEEE, 2022, pp. 1005–1017.
[56] M.M. Amiri, D. Gündüz, Federated learning over wireless fading channels, IEEE Trans. Wireless Commun. 19 (5) (2020) 3546–3557.
[57] K. Yang, T. Jiang, Y. Shi, Z. Ding, Federated learning via over-the-air computation, IEEE Trans. Wireless Commun. 19 (3) (2020) 2022–2035.
[58] Z. Yang, M. Chen, W. Saad, C.S. Hong, M. Shikh-Bahaei, H.V. Poor, S. Cui, Delay minimization for federated learning over wireless communication networks, 2020, arXiv preprint arXiv:2007.03462.
[59] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[60] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: AAAI Conference on Artificial Intelligence, 2016, pp. 2094–2100.
[61] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, 2015, arXiv preprint arXiv:1511.05952.
[62] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, 2017, arXiv preprint arXiv:1707.06347.
[63] H. Mao, M. Alizadeh, I. Menache, S. Kandula, Resource management with deep reinforcement learning, in: Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 2016, pp. 50–56.
[64] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C.H. Liu, D. Yang, Experience-driven networking: A deep reinforcement learning based approach, in: IEEE Conference on Computer Communications, IEEE, 2018, pp. 1871–1879.
[65] T. Li, Z. Xu, J. Tang, Y. Wang, Model-free control for distributed stream data processing using deep reinforcement learning, 2018, arXiv preprint arXiv:1803.01016.
[66] C.H. Liu, Z. Chen, Y. Zhan, Energy-efficient distributed mobile crowd sensing: A deep learning approach, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1262–1276.
[67] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, M. Chen, In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning, IEEE Netw. 33 (5) (2019) 156–165.

Haizhou Du received his Ph.D. degree in computer science and technology from Tongji University, Shanghai, China. His research interests include Machine Learning, Data Management, Distributed System.

Yijian Chen received B.S. degree from Shanghai University of Electric Power in 2019, currently studying for master's degree in school of computer science and technology, Shanghai University of Electric Power. Main research interests are edge computing and federated learning.

Xiaojie Feng received the B.S. degree in computer science and technology from Huaiyin Normal University, China, in 2019. He received his master's degree in the School of Computer Science and Technology, Shanghai University of Electric Power in 2022. His main research interests are distributed deep learning and federated learning.

Qiao Xiang is a professor in the Department of Computer Science at School of Informatics, Xiamen University. He received the Master and Ph.D. Degrees from the Department of Computer Science at Wayne State University, respectively. His research interests include computer networks, formal methods in distributed systems, cyber–physical systems and machine learning systems.



Haoyu Liu received the B.S degree from Northeastern University, China, in 2021. He is currently studying for a master's degree in Department of Computer Science and Technology, Xiamen University. His research interests include network optimization and federated learning.