Contents lists available at ScienceDirect

Computer Networks

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

Communication-efficient asynchronous federated learning in resource-constrained edge computing

Jianchun Liu [a], Hongli Xu [b,*], Yang Xu [b,*], Zhenguo Ma [b], Zhiyuan Wang [b], Chen Qian [c], He Huang [d]

[a] *School of Data Science, University of Science and Technology of China, Hefei, Anhui, 230027, China*
[b] *School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, 230027, China*
[c] *Department of Computer Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064, USA*
[d] *School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, 215006, China*

ARTICLE INFO

Keywords:
Federated learning
Edge computing
Communication-efficient
Asynchronous

ABSTRACT

Federated learning (FL) has been widely used to train machine learning models over massive data in edge computing. However, the existing FL solutions may cause long training time and/or high resource (*e.g.*, bandwidth) cost, and thus cannot be directly applied for resource-constrained edge nodes, such as base stations and access points. In this paper, we propose a novel communication-efficient asynchronous federated learning (CE-AFL) mechanism, in which the parameter server will aggregate the local model updates only from a certain fraction α, with $0 < \alpha < 1$, of all edge nodes by their arrival order in each epoch. As a case study, we design efficient algorithms to determine the optimal value of α for two cases of CE-AFL, single learning task and multiple learning tasks, under bandwidth constraints. We formally prove the convergence of the proposed algorithm. We evaluate the performance of our algorithm with experiments on Jetson TX2, deep learning workstation and extensive simulations. Both experimental results and simulation results on the classical models and datasets show the effectiveness of our proposed mechanism and algorithms. For example, CE-AFL can reduce the training time by about 69% while achieving similar accuracy, and improve the accuracy of the trained models by about 18% under resource constraints, compared with the state-of-the-art solutions.

1. Introduction

With the rapid development of Internet of Things, a massive amount of data are generated from physical worlds each day [1,2]. Under traditional solutions, these data will be forwarded to the remote cloud through core networks for training or processing, which will lead to massive bandwidth consumption. As a result, it is increasingly attractive to encourage local data processing and push more computation to the edge, also called *edge computing* [3]. It motivates the application of federated learning (FL), which implements distributed machine learning at the network edges [4–6].

As shown in Fig. 1, a federated learning system is usually composed of one or more parameter servers (a server group) and a large number of workers (*e.g.*, edge nodes), following the typical parameter server architecture [7]. Each parameter server is controlled by a manager and maintains a partition of the globally shared parameters. For simplicity, we assume one single parameter server, and the solution can be easily extended to the case of multiple parameter servers. Each worker is responsible for computing local statistics such as gradients by training the local data, and communicates only with the parameter server. Specifically, workers will send the local updates to the parameter server, and receive the global updated model from the parameter server. Since the workers will expose not their training data but the trained model to the parameter server, federated learning can efficiently protect users' privacy.

To implement highly efficient federated learning in edge computing, we should take into account the following constraints and factors.

- **Resource Constraints:** The bandwidth between edge nodes and remote parameter servers is usually constrained [8]. However, edge nodes may frequently forward and receive the updated models, which requires an enormous bandwidth cost [9]. For example, the size of parameters in the AlexNet model is about 60M [10]. Given the bandwidth of 1GB, the network is easily congested because of frequent transmission of local and global models. Meanwhile, compared with the cloud platform, edge nodes are

* Corresponding authors.
E-mail addresses: xuhongli@ustc.edu.cn (H. Xu), xuyangcs@ustc.edu.cn (Y. Xu).

https://doi.org/10.1016/j.comnet.2021.108429
Received 27 March 2021; Received in revised form 17 July 2021; Accepted 16 August 2021
Available online 26 August 2021

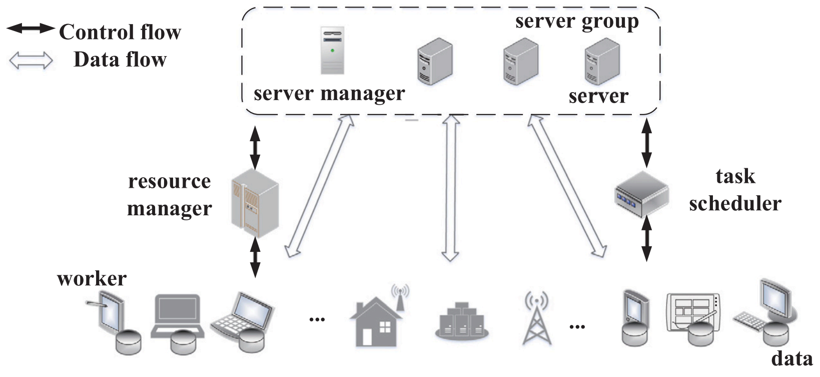

Fig. 1. Illustration of a typical parameter server architecture.

usually resource-constrained, such as computing capacity and memory size.

- **Data Imbalance:** Due to device mobility, *e.g.*, vehicular networks, each edge node will process data from different and varied numbers of devices, which leads to data imbalance among edge nodes. For example, the authors [11] have shown that the amount of data on different edge nodes may vary from 10GB to 1TB in a period of one day.
- **Edge Uncertainty:** Since edge nodes are usually deployed outdoor, some nodes may fail to work occasionally because of a system crash, dead battery or network disconnection [12].

There are two main schemes for federated learning in edge computing. The first way is the synchronous scheme [13–15]. Specifically, on receiving the trained (local) models from *all* specified edge nodes (or workers), the parameter server will aggregate these local models and send the updated global model to all the edge nodes in each epoch. This scheme can guarantee the equivalence between the distributed algorithm and the stand-alone algorithm [16]. However, it brings two main disadvantages. (1) The training time of each epoch mainly depends on the maximum training time among all edge nodes. Due to data imbalance, nodes' heterogeneity (*e.g.*, CPU capacity, memory size), and various network connections (4G, 5G or WiFi), the training time on different edge nodes may be varied significantly, called *straggler* [17], which leads to a longer even unacceptable model training time. (2) The updated local models of all edge nodes will be frequently forwarded to the parameter server(s) for aggregating in the synchronous scheme, which will consume an enormous amount of network bandwidth.

The second way is asynchronous federated learning at the edge [18–20], which allows *partial* (not all) workers to forward updated models to the parameter server(s) for model aggregation in each epoch. Since it does not require the parameter server to wait for local updates from all edge nodes, this scheme can well deal with data imbalance and edge dynamics compared with the synchronous scheme. However, these works ignore the impact of limited resources on training performance, and may require more epochs or more workers involving in each epoch, which leads to massive bandwidth consumption or a longer training time.

To better cope with the above constraints and factors, we propose a communication-efficient asynchronous federated learning (CE-AFL) mechanism, in which the parameter server will aggregate the updated models from a certain fraction α of all edge nodes by their arrival order in each epoch. Note that the subset of local updates involved in the global aggregation will be varied in different epochs, which will be illustrated in Section 2.3. According to the theoretical analysis in Section 2.4, the performance of training depends on the value of α. Therefore, it is a critical challenge to determine the optimal value of α according to resource constraints. The main contributions of this paper are:

- We design a communication-efficient asynchronous federated learning (CE-AFL) mechanism for edge computing, and formally prove the convergence of CE-AFL.
- As a case study, we then propose efficient algorithms to determine the optimal value of α for two cases of CE-AFL, single learning task and multiple learning tasks, so as to achieve less training time under bandwidth constraints. We also prove the convergence of the proposed algorithm.
- Extensive experiments on the classical models and datasets show the effectiveness of our proposed mechanism and algorithms. Specifically, our CE-AFL mechanism can reduce the training time by about 69% and improve the accuracy of the trained models by 18% under resource constraints, compared with the state-of-the-art solutions.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries, proposes the federated learning mechanism, gives the convergence analysis, and formalizes the problem. Two efficient algorithms for single learning task and multiple learning tasks are proposed in Section 3. We report our simulation and experimental results in Section 4. Section 5 discusses the related works. We conclude the paper in Section 6.

2. Preliminaries

In this section, we introduce the concept of federated learning (Section 2.1) and propose the novel federated learning mechanism (Section 2.2). In Section 2.3, we illustrate CE-AFL through an example. Then, we give the convergence analysis of CE-AFL (Section 2.4), and describe the problem formulation (Section 2.5). For ease of description, some key notations are listed in Table 1.

2.1. Federated Learning (FL)

2.1.1. The goal of FL

Federated learning enables training global models over distributed datasets in resource-constrained edge computing. Ideally, the training data from diverse users/devices result in improved representation and generalization of machine learning models [6]. Each edge node, acting as a worker, trains the model locally using its private data, while the parameter server aggregates the local updates from workers and sends the global updated model to workers. To protect users' privacy, the workers do not expose their training data to the parameter server, commonly located in a cloud [4], and instead only expose the trained local model. Moreover, federated learning can handle non-I.I.D. (non-independently identically distribution) training data which are massively distributed in the edge computing [21].

For convenience, given a training sample q_i, its loss function is denoted as f_i. Then, the loss function for $\mathcal{N}$ training samples is expressed as $f(w) = \frac{1}{\mathcal{N}}\sum_{i=1}^{\mathcal{N}} f_i(w)$, where w denotes the parameter

Table 1
Key notations.

Symbol	Semantics
V	A set of edge nodes
Γ_i	The local dataset on the edge node v_i
D	The number of iterations in a local update
T	The total number of training epochs until the training terminates
$\mathbb{T}$	The vector's transposition
$\mathcal{K}$	The number of resource categories
η	The learning rate
n	The total number of edge nodes
g_k	The consumption of resource k for local updates on an edge node
b_k	The consumption of resource k for communication between a server and a worker
B_k	The total budget for each category of resource k
L	A set of learning tasks
α_j	A certain fraction of local updates that will be involved in the global aggregation of learning task j on the server
Φ	The set of α_j, with $j \in \{1,2,\ldots,L\}$
$\hat{w}^D$	The model parameter after D local updates
β_i^t	Whether the local update of edge node v_i is involved or not in the epoch t
$F(w^T)$	The global loss function after T epochs
$F(w^*)$	The optimal value of loss function $F(w)$

vector of model. As a result, the objective of federated learning can be expressed as $\min_{w \in R^m} f(w)$, where R^m denotes the m-dimension real-number space. We note that the problem structure can cover both the simple models, *e.g.*, linear or logistic regressions [22], support vector machines (SVM) [23], and more complicated models, *e.g.*, conditional random fields or neural networks [10]. For $\mathcal{N}$ input–output pairs $\{x_i, y_i\}_{i=1}^{\mathcal{N}}$, $x_i \in R^m$ and $y_i \in R$ or $y_i \in \{-1,1\}$, some typical examples of f_i include

- Linear regression: $f_i(w) = \frac{1}{2}(x_i^{\mathbb{T}} w - y_i)^2, y_i \in R$
- Logistic regression: $f_i(w) = -\log(1 + \exp(-y_i x_i^{\mathbb{T}} w))$, $y_i \in \{-1,1\}$
- Support vector machines: $f_i(w) = \max\{0, 1 - y_i x_i^{\mathbb{T}} w\}$, $y_i \in \{-1,1\}$

where $\mathbb{T}$ denotes the vector's transposition. More complicated non-convex problems arise in the context of neural networks, in which the network makes prediction through a non-convex function of the feature vector x_i, rather than via the linear-in-the-features mapping $x_i^{\mathbb{T}} w$. In fact, the resulting loss function can still be written as $f_i(w)$ [24].

2.1.2. Optimization algorithms for FL

There are two fundamental algorithms to optimize the loss function of FL.

Gradient Descent (GD) [25] is the classical first-order method. The basic idea of GD is to minimize the first-order Taylor expansion of the objective function in the current state, so as to approximately optimize the objective function itself. Specifically, for a loss function f, the following problem can be solved at the current state w:

$$\min_w f(w) \approx \min_w f(w_t) + \nabla f(w_t)^T (w - w_t) \tag{1}$$

The right part of Eq. (1) is linear with respect to the independent variable w, and minimizes $\nabla f(w_t)^T(w)$, which is opposite to the direction of the gradient $\nabla f(w_t)$. Therefore, the updating rule of gradient descent is as follows

$$w_{t+1} = w_t - \eta \nabla f(w_t) \tag{2}$$

where η is the step size, also known as the learning rate.

Stochastic Gradient Descent (SGD) [26] is used to sample the training data randomly, and its updating formula is

$$w_{t+1} = w_t - \eta_t \nabla f_{i_t}\left(w_t\right) \tag{3}$$

where i_t is the data label of random sampling in the tth iteration. The gradient obtained from random sampling data with playback is an unbiased estimation of the gradient by all data, *i.e.*, $\mathbb{E}_{i_t} \nabla f_{i_t}(w_t) = \nabla f(w_t)$. What is more, since only one sample is randomly selected in each training epoch, the computation load will be greatly reduced. Comparing two algorithms, this paper adopts SGD, which is a natural alternative to GD and can greatly improve learning efficiency with less computing time for sampling data.

Algorithm 1 Communication-Efficient Asynchronous FL (CE-AFL)

```
1:  for each epoch $t \in \{1,2,...,T\}$ do
2:    Processing at the Parameter Server
3:    if the resource constraints are satisfied then
4:      while No. of received local updates $< \alpha \cdot n$ do
5:        Waiting for local updates from workers
6:      Aggregating the local models by their arrival order
7:      Compute the global loss $F(w)$ according to Eq. (5)
8:      Distribute the updated global model $w$ to the workers
9:      Update the budgets $B_k$ for each category of resource $k$
10:   Processing at the Edge Node $v_i$
11:   while Has sent local update to server and No global model is
      received do
12:     Waiting for the updated global model
13:   for each local update $d \in \{1,2,...,D\}$ do
14:     Update the local model: $w_{t+1} = w_t - \eta_t \nabla f_{i_t}\left(w_t\right)$
15:   Push local update to the server
16: Return the final model $w$ and loss function $F(w)$
```

2.2. Communication-efficient asynchronous FL

Assume that there is a set of edge nodes $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, with $|\mathcal{V}| = n > 2$, in the edge computing system. Each edge node trains the model over a local dataset Γ_i with its size $|\Gamma_i|$, $i \in \{1,2,\ldots,n\}$. For each node v_i, the loss function on the local dataset Γ_i is

$$F_i(w) = \frac{1}{|\Gamma_i|} \sum_{q_j \in \Gamma_i} f_j(w) \tag{4}$$

In this section, we propose the communication-efficient asynchronous federated learning (CE-AFL) mechanism, which is formally described in Algorithm 1. Similar to the synchronous and asynchronous schemes, CE-AFL also consists of training epochs. Let variable T denotes the total number of training epochs until the training terminates. One global update (model aggregation) will be performed in each epoch. Assume that there are $D(\geq 1)$ local updates (iterations) between two consecutive global updates. Let $\alpha \in \{\frac{1}{n}, \frac{2}{n}, \ldots, 1\}$ be a fraction of local model updates from all edge nodes for global model aggregation on the parameter server in each epoch $t \in \{1,\ldots,T\}$. On receiving local updated models from *arbitrary* $\alpha \cdot n$ workers in an epoch, the parameter server will perform the model aggregation by their arrival order and derive the updated global model (Line 3–7). Then, the parameter server distributes the global model to the workers have already sent local updates to the server (Line 8) and updates the resource budgets (Line 9). On the edge node side (Line 11–15), on receiving the global model, the worker will perform several updates for the local model, and then push the updated local model to the parameter server for model aggregation. After T epochs, the global loss function $F(w^T)$ of the training model is

$$F(w^T) = \frac{\sum_{i=1}^n \sum_{q_j \in \Gamma_i} \beta_i^T f_j(w^T)}{\sum_{i=1}^n \beta_i^T |\Gamma_i|} = \frac{\sum_{i=1}^n \beta_i^T |\Gamma_i| F_i(w^T)}{\sum_{i=1}^n \beta_i^T |\Gamma_i|} \tag{5}$$

where β_i^t is a binary variable to indicate whether the local update of edge node v_i is involved or not in the epoch t. Thus, it follows $\sum_{i=1}^n \beta_i^t = \alpha \cdot n, \forall t \in \{1,\ldots,T\}$. The whole training process will continue until the resource constraints are violated or the global convergence is reached. That is, $F(w^T) - F(w^*) < \varepsilon$, where ε is an arbitrary small positive value, and w^* is the optimal solution for the loss function $F(w)$.

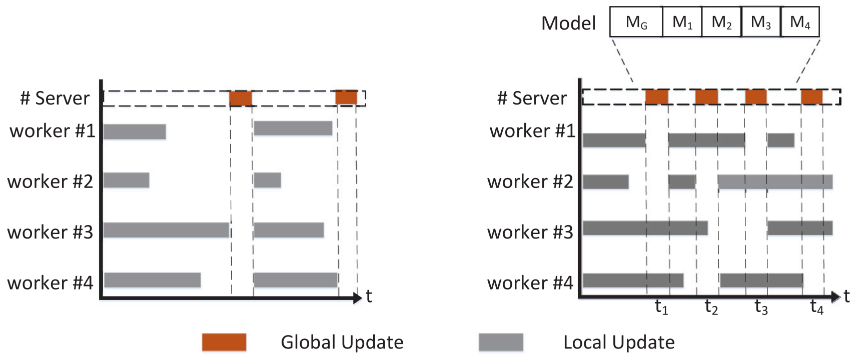

Fig. 2. Illustration of synchronous FL and CE-AFL. Different colors (light gray and dark orange) denote the local and global updates, respectively. Given a time length, there are 2 and 4 global updates for the synchronous scheme (left plot) and CE-AFL (right plot). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.3. Illustration of CE-AFL

We give an example in Fig. 2 for better illustration of CE-AFL. Assume that there are one parameter server and ten workers in the edge computing system. Due to some reasons (*e.g.*, unavailability), the existing works [13–15] usually selects partial workers for model training tasks. Thus, four workers (#1-#4) are selected to participate in the model training task here. This figure shows the local (light gray) and global (dark orange) updates of both the synchronous scheme and CE-AFL with a fixed-length time period. For the synchronous scheme in the left plot, only after the parameter server receives all local updates from four workers, it will perform model aggregation to derive the updated global model. When the workers receive the global model, they will continue to train with local data. By the left plot, there are only two global model updates in the synchronous scheme.

In CE-AFL, let $\alpha = \frac{1}{2}$. In other words, on receiving local updates from *arbitrary* two workers, the parameter server will perform the global update, as shown in the right plot of Fig. 2. In practice, the data and resources (*e.g.*, computing capacity and bandwidth budget) on the workers are always time-varying. Thus, the subset of local updates involved in the global update will be varied in different epochs. For example, the parameter server aggregates the local updates from workers #1 and #2 in the first epoch, and from workers #2 and #4 in the second epoch. Note that the local updated model will be aggregated in the next global update if the server receives the update during the current aggregation. Given a fixed time period, there are four global updates in CE-AFL (right plot), and only two global updates in the synchronous scheme (left plot). Thus, CE-AFL will perform more global updates and converge faster than the synchronous scheme with the same time budget constraint.

Note that our proposed CE-AFL mechanism may suffer from another problem, *delayed update*. For example, when worker #3 sends its local updated model to the server for the global model aggregation at the first time, the server has aggregated the local updated models from workers #1, #2, and #4 at time points t_1 and t_2. We adopt the *delay compensation* mechanism [27] to alleviate this problem. In Fig. 2, we use M_G to denote the current global model and $M_i, \forall i \in \{1, \ldots, 4\}$, to denote the latest local updated model from worker i. These models will be recorded on the server to perform delay compensation for outdated models. For example, considering a time point t between t_2 and t_3, worker #1 has sent the local updated model to the server only once, while the server has performed two global model aggregations. Then, the *staleness* of worker #1 is the gap between the number of global updates and the number of local model updates, *e.g.*, here $2 - 1 = 1$. After the server receives the local model from worker #1 twice, the model M_1 will be updated with decay coefficient ς, with $0 < \varsigma < 1$, *i.e.*, $M_1 = \varsigma^x \cdot M_1 + (1-\varsigma^x) \cdot M_G$, where x denotes the staleness of worker #1. By this way, the impact of the outdated models can be alleviated. Note that how to determine the coefficient ς is not the focus of this paper, and we set the value of ς according to [27] in the evaluation.

Besides, we give an example to show how our proposed solution solves the problem of edge uncertainty. As shown in Fig. 3, the parameter server cannot receive local updates from worker #4 because of system crash or network disconnection. Thus, there is no global model aggregation in the synchronous scheme (left plot). The parameter server perform global update after receiving local updates from worker #1 and #2 ($\alpha = \frac{1}{2}$) in the first epoch, and worker #2 and #3 in the second epoch. Even though we have not received the local model update from the worker #4, CE-AFL (right plot) still has three global updates. Thus, our proposed solution can effectively solve the edge uncertainty problem.

2.4. Convergence analysis

To analyze the feasibility of the proposed mechanism for model training, we prove that CE-AFL can achieve a constant convergence bound. We first make the following three assumptions [20].

Assumption 1 (*Smoothness*)**.** Let $L > 0$. The loss function f is L-smooth w.r.t. the model parameter if for $\forall w_1, w_2$,

$$f(w_2) - f(w_1) \le \langle \nabla f(w_1), w_2 - w_1 \rangle + \frac{L}{2} \|w_2 - w_1\|^2 \tag{6}$$

Assumption 2 (*Strong Convexity*)**.** Let $\mu \ge 0$. The loss function f is μ-strongly convex if $\forall w_1, w_2$,

$$f(w_2) - f(w_1) \ge \langle \nabla f(w_1), w_2 - w_1 \rangle + \frac{\mu}{2} \|w_2 - w_1\|^2 \tag{7}$$

Note that if $\mu = 0$, f is convex, *i.e.*

$$f(w_2) - f(w_1) \ge \langle \nabla f(w_1), w_2 - w_1 \rangle \tag{8}$$

This assumption can be satisfied for models with convex function, *e.g.*, linear regression and SVM.

Assumption 3 (*Bounded Gradient Variance*)**.** The variance of stochastic gradients at each edge node is bounded: $\mathbb{E}\|\nabla f(w;q) - \nabla F(w)\|^2 \le \mathcal{P}, \forall w \in \mathcal{R}^m, q \in \Gamma_i, i \in \{1, \ldots, n\}$, where $\mathcal{P}$ is a positive number.

Assumption 4 (*Existence of Global Optimization*)**.** Assume that there exists at least one solution, denoted as w^*, that can achieve the global minimum of the loss function $f(w)$.

CE-AFL will perform T epochs until the global convergence is achieved. In each epoch, D local updates will be performed on an edge node. We first derive the convergence bound of each local update (Theorem 5). Then, we prove the convergence of model training at each epoch t and the convergence bound after T epochs (Theorem 6).

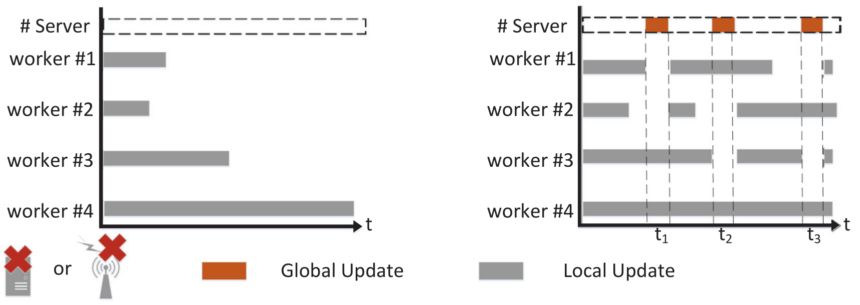

Fig. 3. Illustration of how CE-AFL handles edge uncertainty. *Left plot:* synchronous scheme; *right plot:* CE-AFL.

Theorem 5. *Assume that the global loss function F is L-smooth and μ-strongly convex, and each worker executes D local updates before reporting the updated model to the parameter server. When all the following three conditions are satisfied:*

- $\eta < \frac{1}{L}$
- $\eta\mu > 1 - \sqrt[D]{\frac{1}{4n}}$
- $F(w^0) - F(w^*) > \frac{D\eta\mathcal{P}}{4(1-\eta\mu)^D}$

we have

$$\begin{cases} 2\alpha n(1-\eta\mu)^D \in (0,1) \\ \mathbb{E}[F(\hat{w}^D) - F(w^*)] \le (1-\eta\mu)^D[F(w^0)-F(w^*)] + \frac{D\eta\mathcal{P}}{2} \end{cases}$$

where $\hat{w}^D$ denotes the model parameter after D local updates and w^0 is the initial model parameter.

Proof. We first consider the convergence analysis of the D local updates. For each local update $d \in \{1,\dots,D\}$, by the assumptions of smoothness and strong convexity, we have

$$\begin{aligned}
&\mathbb{E}[F(\hat{w}^d) - F(w^*)] \\
&\le F(\hat{w}^{d-1}) - F(w^*) - \eta\mathbb{E}[\langle \nabla F(\hat{w}^{d-1}), \nabla f(\hat{w}^{d-1};q_d)\rangle] \\
&\quad + \frac{L\eta^2}{2}\mathbb{E}[\|\nabla f(\hat{w}^{d-1};q_d)\|^2] \\
&\le F(\hat{w}^{d-1}) - F(w^*) - \frac{\eta}{2}\|\nabla F(\hat{w}^{d-1})\|^2 \\
&\quad + \frac{\eta}{2}\mathbb{E}[\|\nabla F(\hat{w}^{d-1}) - \nabla f(\hat{w}^{d-1};q_d)\|^2] \\
&\le F(\hat{w}^{d-1}) - F(w^*) - \frac{\eta}{2}\|\nabla F(\hat{w}^{d-1})\|^2 \\
&\quad + \frac{\eta\mathcal{P}}{2} \qquad (\rhd\ \mathbb{E}\|\nabla f(w;q) - \nabla F(w)\|^2 \le \mathcal{P}) \\
&\le F(\hat{w}^{d-1}) - F(w^*) - \eta\mu[F(\hat{w}^{d-1}) - F(w^*)] \\
&\quad + \frac{\eta\mathcal{P}}{2} \qquad (\rhd\ F(w) \le F(w^*) + \frac{1}{2\mu}\|\nabla F(w)\|^2, \forall x) \\
&\le (1-\eta\mu)[F(\hat{w}^{d-1}) - F(w^*)] + \frac{\eta\mathcal{P}}{2}
\end{aligned} \tag{9}$$

We have derived the convergence result of each local update $d \in \{1,\dots,D\}$. By telescoping and taking the total expectation, after D local updates, we have

$$\begin{aligned}
&\mathbb{E}[F(\hat{w}^D) - F(w^*)] \\
&\le (1-\eta\mu)[F(\hat{w}^{D-1}) - F(w^*)] + \frac{\eta\mathcal{P}}{2} \\
&\le (1-\eta\mu)[(1-\eta\mu)[F(\hat{w}^{D-2}) - F(w^*)] + \frac{\eta\mathcal{P}}{2}] + \frac{\eta\mathcal{P}}{2} \\
&\dots\dots \qquad \text{(Telescoping by Eq.(9))} \\
&\le (1-\eta\mu)^D[F(w^0) - F(w^*)] + \frac{\eta\mathcal{P}}{2}\sum_{d=1}^{D}(1-\eta\mu)^{d-1} \\
&\le (1-\eta\mu)^D[F(w^0) - F(w^*)] + \frac{\eta\mathcal{P}}{2}\frac{1-(1-\eta\mu)^D}{1-(1-\eta\mu)} \\
&\le (1-\eta\mu)^D[F(w^0) - F(w^*)] + \frac{\eta\mathcal{P}}{2}\frac{D\eta\mu}{1-(1-\eta\mu)} \\
&\quad (\rhd\ \eta\mu \le 1, 1-(1-\eta\mu)^D \le D\eta\mu) \\
&\le (1-\eta\mu)^D[F(w^0) - F(w^*)] + \frac{D\eta\mathcal{P}}{2}
\end{aligned} \tag{10}$$

On the parameter server side, it will perform global model aggregation after receiving $\alpha \cdot n$ updated models at the epoch t. It follows $w^t = \frac{1}{\alpha n}\sum_{i=1}^{\alpha n}\hat{w}_i^D$, where $\hat{w}_i^D$ denotes the local updated model in the worker node v_i after D local updates.

Theorem 6. *After T epochs, the convergence bound of CE-AFL is*

$$\mathbb{E}[F(w^T) - F(w^*)] \le \tau[F(w^0) - F(w^*)] + (1-\tau)\frac{D\eta\mathcal{P}}{4\varphi}$$

where $\varphi = (1-\eta\mu)^D$, and $\tau = (2\alpha n\varphi)^T$.

Proof. According to the definition of w^t, for each global update at epoch $t \in \{1,\dots,T\}$, we have

$$\begin{aligned}
&\mathbb{E}[F(w^t) - F(w^*)] \\
&\le \frac{1}{\alpha n}\sum_{i=1}^{\alpha n}[F(\hat{w}_i^D) - F(w^*)] \\
&\le \frac{1}{\alpha n}\sum_{i=1}^{\alpha n}[(1-\eta\mu)^D(F(w_i^0) - F(w^*)) + \frac{D\eta\mathcal{P}}{2}] \\
&\le \alpha n[(1-\eta\mu)^D(F(w^0) - F(w^*)) + \frac{D\eta\mathcal{P}}{2}] \\
&\le \alpha n[(1-\eta\mu)^D(F(w^0) - F(w^{t-1}) + F(w^{t-1}) + F(w^*)) + \frac{D\eta\mathcal{P}}{2}] \\
&\le \alpha n(1-\eta\mu)^D(F(w^0) - F(w^{t-1})) \\
&\quad + \alpha n(1-\eta\mu)^D(F(w^{t-1}) - F(w^*)) + \frac{\alpha nD\eta\mathcal{P}}{2} \\
&\le \alpha n(1-\eta\mu)^D(F(w^{t-1}) - F(w^*)) \\
&\quad + \alpha n(1-\eta\mu)^D(F(w^{t-1}) - F(w^*)) + \alpha\frac{nD\eta\mathcal{P}}{2} \\
&\le (2\alpha n(1-\eta\mu)^D)[F(w^{t-1}) - F(w^*)] + \alpha\frac{nD\eta\mathcal{P}}{2}
\end{aligned} \tag{11}$$

Then, the convergence bound after T training epochs can be derived as follows

$$\begin{aligned}
&\mathbb{E}[F(w^T) - F(w^*)] \\
&\le (2\alpha n(1-\eta\mu)^D)[F(w^{T-1}) - F(w^*)] + \alpha\frac{nD\eta\mathcal{P}}{2} \\
&\dots\dots \qquad \text{(Telescoping by Eq.(11))} \\
&\le (2\alpha n(1-\eta\mu)^D)^T[F(w^0) - F(w^*)] + \frac{(1-(2\alpha n(1-\eta\mu)^D)^T)\alpha nD\eta\mathcal{P}}{2(1-(2\alpha n(1-\eta\mu)^D))} \\
&(\rhd\ \eta\mu > 1 - \sqrt[D]{1/4n}, 2\alpha n(1-\eta\mu)^D < \frac{1}{2}) \\
&\le (2\alpha n(1-\eta\mu)^D)^T[F(w^0) - F(w^*)] + \frac{(1-(2\alpha n(1-\eta\mu)^D)^T)\alpha nD\eta\mathcal{P}}{4\alpha n(1-\eta\mu)^D} \\
&\le (2\alpha n(1-\eta\mu)^D)^T[F(w^0) - F(w^*)] + \frac{D\eta\mathcal{P}}{4(1-\eta\mu)^D}(1-(2\alpha n(1-\eta\mu)^D)^T)
\end{aligned} \tag{12}$$

For simplicity, let $\varphi = (1-\eta\mu)^D$. We write the above equation as follows:

$$\mathbb{E}[F(w^T) - F(w^*)] \\ \leq \tau[F(w^0) - F(w^*)] + (1-\tau)\frac{D\eta\mathcal{P}}{4\varphi} \tag{13}$$

where $\tau = (2\alpha n\varphi)^T$ □

We note that the convergence bound (*i.e.*, the optimality gap), $F(w^T) - F(w^*)$, is related to the value of α and the number of epochs T by Eq. (13). Furthermore, the optimality gap can be reduced when both α and T become larger, which requires massive resource cost. Thus, it is a challenge to determine the optimal values of α and T for the optimization of training performance (*e.g.*, training time) under resource constraints.

2.5. Problem formulation

We define the asynchronous federated learning with resource constraints (AFL-RC) problem. Specifically, we will determine the values of α and T for a learning task so as to minimize the training time with resource constraints. It needs to consume several categories of resources (*e.g.*, network bandwidth, CPU, *etc.*.) during model training, including the local updates on the workers, the model exchanging between workers and parameter servers, and the global updates on the parameter servers. Here we mainly consider the static scenario, in which the edge nodes are fixed, such as base station or surveillance camera, *etc.*. We will discuss the problem of dynamic scenario in Section 6. Assume that there are totally $\mathcal{K}$ different categories of resources. Let g_k denotes the consumption of resource $k \in \{1, 2, \ldots, \mathcal{K}\}$ for local updates on an edge node. Meanwhile, b_k denotes the consumption of resource k for the model exchanging once between an edge node and a parameter server. Once the global model has been updated, the server distributes the updated model to all workers. Thus, for each resource category k, the total resource consumption of local updates and global updates at all nodes after T epochs is $T \cdot n \cdot g_k$ and $T \cdot (\alpha+1) \cdot n \cdot b_k$, respectively. Let B_k denotes the total budget for each resource category k. Accordingly, we formulate the AFL-RC problem as follows: $\min_{T\in\{1,2,3,\ldots\}} F(w^T)$

$$s.t.\begin{cases} T \cdot n \cdot [g_k + 2\alpha \cdot b_k] \leq B_k, \quad \forall k \\ \alpha \in \{\frac{1}{n}, \frac{2}{n}, \ldots, 1\} \end{cases} \tag{14}$$

The first set of inequalities expresses the constraints of each resource category $k \in \{1, \ldots, \mathcal{K}\}$ during totally T training epochs. The second set of equations denotes the bound of variable α. The objective of the AFL-RC problem is to minimize the global loss function $F(w^T)$.

By the first set of constraints in Eq. (14), we find that two parameters α and T are dependent. For simplicity, we consider how to determine the optimal value of α for two cases in edge computing. One is the case in which there is a single learning task. The other is the more general case in which there are multiple learning tasks. Then, the value of parameter T can be accordingly determined.

3. A case study of bandwidth optimization

In this section, we design efficient algorithms to determine the proper value of α so as to achieve convergence with less training time under the bandwidth constraint. Here, we mainly consider the bandwidth resource as the most important resource consumption, which is always the communication bottleneck in edge computing [8]. We first consider the scenario in which there is only single training task, and can accurately achieve the optimal value of α for parameter aggregation (Section 3.1). Then, we consider the scenario of multiple tasks for model training (Section 3.2). In order to determine the value α of each task, we propose an algorithm called SQP-PA to solve the problem, and prove the global convergence of the algorithm.

3.1. Algorithm for a single learning task

We consider a simple case of only single learning task in edge computing. The objective of Eq. (14) is equivalent to minimizing $F(w^T) - F(w^*)$. Furthermore, we use the upper bound in Eq. (13) as an approximation of $F(w^T) - F(w^*)$, yielding the new objective as follows:
$\min_{\alpha\in\{\frac{1}{n},\ldots,1\}} (2\alpha n\varphi)^T[F(w^0) - F(w^*)] + [1-(2\alpha n\varphi)^T]\frac{D\eta\mathcal{P}}{4\varphi}$
By Theorem 5, since $2\alpha n\varphi < 1$, it is easy to see that the objective function decreases with the increasing value of T. According to the first inequality in Eq. (14), the optimal solution of T is $\lfloor \min_k \frac{B_k}{n\cdot[g_k+(\alpha+1)\cdot b_k]}\rfloor$. To simplify the analysis, we ignore the rounding operations and substitute $T \approx \min_k \frac{B_k}{n\cdot[g_k+(\alpha+1)\cdot b_k]} = \max_k \frac{n\cdot[g_k+(\alpha+1)\cdot b_k]}{B_k}$ into the objective function. Under this situation, the local update does not require any bandwidth cost, that is, $g_k = 0, \forall k \in \{1, 2, \ldots, K\}$. Let B denotes the bandwidth constraint for a learning task. According to the above approximation, we reformulate the AFL-RC problem as

$$\min_{\alpha\in\{\frac{1}{n},\ldots,1\}} H(\alpha)$$

$$s.t.\begin{cases} 2\alpha \cdot n \cdot b \cdot T \leq B, \\ \alpha \in \{\frac{1}{n}, \frac{2}{n}, \ldots, 1\} \end{cases} \tag{15}$$

where

$$H(\alpha) = (2\alpha n\varphi)^{\frac{(1+\alpha)nb}{B}}[F(w^0) - F(w^*)] + [1-(2\alpha n\varphi)^{\frac{(1+\alpha)nb}{B}}]\frac{D\eta\mathcal{P}}{4\varphi} \tag{16}$$

We can derive the optimal $\alpha^* = \text{argmin}_{\alpha\in\{\frac{1}{n},\frac{2}{n},\ldots,1\}} H(\alpha)$.

Theorem 7. *When there are sufficient bandwidth resource, i.e., $B \to \infty$, the convergence bound can always be achieved during the model training.*

Proof. If $B \to \infty$, it follows $\frac{(1+\alpha)nb}{B} \to 0$. Accordingly, we can derive that $(2\alpha n\varphi)^{\frac{(1+\alpha)nb}{B}} \approx 1$. Combining with Eq. (16), $H(\alpha) \approx F(w^0) - F(w^*)$, which is a constant value. Thus, the convergence can always be reached no matter the value of α. □

In order to determine the optimal value of α for function H with bandwidth resource constraint, we consider the monotonicity of function $e^{G(\alpha)} = (2\alpha n\varphi)^{\frac{(1+\alpha)nb}{B}} = e^{\frac{(1+\alpha)nb}{B}\ln(2\alpha n\varphi)}$, where $G(\alpha) = \frac{(1+\alpha)nb}{B}\ln(2\alpha n\varphi)$. Since the exponential function e^x is monotonically increasing, we can easily derive the optimal value by solving the objective function G. We construct an auxiliary function $h(\alpha) = \frac{nb}{B}[\ln(2\alpha n\varphi) + \frac{1}{\alpha} + 1]$ according to $G'(\alpha)$. Then $h'(\alpha) = \frac{nb}{B}(\frac{1}{\alpha} - \frac{1}{\alpha^2}) < 0$ with $\alpha \in [\frac{1}{n}, 1]$. Thus $h(\alpha)$ is monotonically decreasing with α. In the following, assume that the two conditions in Theorem 5 are satisfied. We consider three cases of function $h(\alpha)$.

Theorem 8. *Assume that $h(\alpha) > 0$, with $\alpha \in [\frac{1}{n}, 1]$. If $\varphi > \frac{e^{-2}}{2n}$, we have $\alpha^* = \frac{1}{n}$.*

Proof. To complete the proof, we first relax α to a continuous interval, *i.e.*, $\alpha \in [\frac{1}{n}, 1]$. Then, we obtain the value of $\alpha \in \{\frac{1}{n}, \frac{2}{n}, \ldots, 1\}$ by using the randomized rounding method [28].

Due to $\varphi > \frac{e^{-2}}{2n}$, we have

$$h(1) = \frac{nb}{B}[\ln(2n\varphi) + 2] > 0 \tag{17}$$

Besides, as $h(\alpha)$ is decreasing with $\alpha \in [\frac{1}{n}, 1]$, it follows

$$h(\alpha) = \frac{nb}{B}[\ln(2\alpha n\varphi) + \frac{1}{\alpha} + 1] > 0 \Longrightarrow G'(\alpha) > 0 \tag{18}$$

Thus, the minimum value of function G is $G(\frac{1}{n})$. We can conclude that $\alpha^* = \frac{1}{n}$, if $\varphi > \frac{e^{-2}}{2n}$. □

Theorem 9. *Assume that $h(\alpha) < 0$, with $\alpha \in [\frac{1}{n}, 1]$. If $\varphi < \frac{e^{-(n+1)}}{2}$, we have $\alpha^* = 1$.*

Algorithm 2 SQP based Proportion Assignment (SQP-PA)

```
1:  Step 1. Initialization.
2:  Let $\Phi^0 \in \Theta^m$, $E_0 \in \Theta^{m\times m}$, $B' = B$.
3:  Initialize a symmetric positive definite matrix $U$.
4:  Step 2. Computing the Search Direction.
5:      Compute $(\lambda_r, \mathcal{D}^r)$ by Eq. (22)
6:      If $(\lambda_r, \mathcal{D}^r) = (0,0)$ then stop.
7:      Compute $\overline{\mathcal{M}}_{j_r}, \overline{\mathcal{M}}_{j,j_r}, U$ by Eq. (23).
8:      If the matrix $U$ is of full rank, then obtain $\widetilde{s}^r$ by solving $U^{\mathbb{T}} s = -\|\mathcal{D}^r\|^{\pi} q - \overline{\mathcal{M}}_{j_r}(\Phi^r + \mathcal{D}^r)$, where $q = (1, ..., 1)^{\mathbb{T}}$.
9:      If $\|\widetilde{s}^r\| > \|\mathcal{D}^r\|$ or the matrix $U$ is not of full rank,
10:         then $\widetilde{\mathcal{D}}^r = 0$; else, $\widetilde{\mathcal{D}}^r = \widetilde{s}^r$.
11: Step 3. Start Nonmonotone Line Search.
12: Step 4. Performing Updates.
13:     Compute a new symmetric definite positive matrix $E_{r+1}$.
14:     $\Phi^{r+1} = \Phi^r + \delta_r \mathcal{D}^r + \delta_r^2 \widetilde{\mathcal{D}}^r, r = r+1$.
15:     Update resource budget $B'$
16:     If $B' \le 0$, then stop; else go back to Step 2.
```

Since the proof is similar to that of Theorem 8, we omit the detailed proof here.

Theorem 10. *Assume that* $h(\alpha) = 0, \exists \alpha \in (\frac{1}{n}, 1)$. *If* $\frac{e^{-(n+1)}}{2} \le \varphi \le \frac{e^{-2}}{2n}$, *we have* $\alpha^* \approx \sqrt[3]{-\frac{\mathcal{E}}{2} + \mathcal{Q}} + \sqrt[3]{-\frac{\mathcal{E}}{2} - \mathcal{Q}}$, *where* $\mathcal{Q} = \sqrt{(\frac{\mathcal{E}}{2})^2 + (\frac{\mathcal{F}}{3})^3}$, $\mathcal{E} = \frac{-27n\varphi-23}{54n^3\varphi^3}$, $\mathcal{F} = -\frac{13}{12n^2\varphi^2}$.

Proof. Since $\alpha^* \in (\frac{1}{n}, 1)$, we make Taylor approximate expansion [29] of $\ln(2\alpha n\varphi)$ at $\alpha = \frac{1}{2n\varphi}$, and only focus on the former two items of expansion, *i.e.*, the first and second order. Combining the two expanded items, we have

$$\begin{aligned} h(\alpha) &= \frac{nb}{B}[\ln(2\alpha n\varphi) + \frac{1}{\alpha} + 1] \\ &\approx \frac{nb}{B}[-2n^2\varphi^2\alpha^2 + 4n\varphi\alpha + \frac{1}{\alpha} - \frac{1}{2}] = \widetilde{h}(\alpha) = 0 \end{aligned} \tag{19}$$

Then, we derive the value of α' so that $\widetilde{h}(\alpha') = 0$. If $\frac{1}{n} \le \alpha \le \alpha', \widetilde{h}(\alpha) \le 0$ and $\alpha' \le \alpha \le 1, \widetilde{h}(\alpha) \ge 0, \exists \alpha$. Thus, we derive that $G(\alpha^*) = G(\alpha')$ and $\alpha^* = \alpha' \approx \sqrt[3]{-\frac{\mathcal{E}}{2} + \mathcal{Q}} + \sqrt[3]{-\frac{\mathcal{E}}{2} - \mathcal{Q}}$, where $\mathcal{Q} = \sqrt{(\frac{\mathcal{E}}{2})^2 + (\frac{\mathcal{F}}{3})^3}$, $\mathcal{E} = \frac{-27n\varphi-23}{54n^3\varphi^3}$, and $\mathcal{F} = -\frac{13}{12n^2\varphi^2}$. □

3.2. Algorithm for multiple learning tasks

In many practical scenarios, there are usually multiple simultaneous learning tasks, *e.g.*, machine translation, face recognition and speech recognition, on distributed edge nodes [30–32]. In this paper, we focus our attention on multiple independent learning tasks, while the case of multiple dependent learning tasks will be regarded as our future work.

Without loss of generality, we just consider the bandwidth resource constraint in the problem definition. When multiple learning tasks $L = \{l_1, \dots, l_m\}$ are trained in edge computing, we will determine the optimal value of parameter α_j for each task l_j so as to minimize the maximum loss function of all learning tasks under resource constraints. Intuitively, we expect to minimize the loss function of each task l_j after T epochs. That is $\min F(w_j^T)$. The objective function $\min F(w_j^T)$ for the task l_j is equivalent to minimizing the gap between the loss value $F(w_j^T)$ at T epochs and the optimal loss value $F(w_j^*)$, *i.e.*, $F(w_j^T) - F(w_j^*)$. Furthermore, we use the upper bound in Eq. (13), *i.e.*, $\mathbb{E}[F(w_j^T) - F(w_j^*)] \le (2\alpha_j n\varphi)^{T_j}[F_j(w^0) - F_j(w^*)] + (1 - (2\alpha_j n\varphi)^{T_j})\frac{D_j\eta\mathcal{P}}{4\varphi}$, as an approximation of $F(w_j^T) - F(w_j^*)$. Thus, the new objective for the learning task l_j is $\mathbf{F}(j) = (2\alpha_j n\varphi)^{T_j}[F_j(w^0) - F_j(w^*)] + (1 - (2\alpha_j n\varphi)^{T_j})\frac{D_j\eta\mathcal{P}}{4\varphi}$. Note that we should minimize the maximum loss function of all learning tasks under resource constraints, *i.e.*, $\min_{\alpha_{j'} \in \{\frac{1}{n}, \dots, 1\}} \mathbf{F}(j')$, where $j' = \mathrm{argmax}_{j \in \{1,2,\dots,m\}}(\mathbf{F}(j))$. Accordingly, the problem can be described as follows:

$$\min_{\alpha_{j'} \in \{\frac{1}{n}, \dots, 1\}} \mathbf{F}(j')$$
$$s.t. \begin{cases} \sum_{j \in \{1,\dots,m\}} 2\alpha_j \cdot n \cdot b^j \cdot T_j \le B \\ \alpha_j \in \{\frac{1}{n}, \dots, 1\} & \forall j \end{cases} \tag{20}$$

b^j denotes the bandwidth cost for forwarding a global update of learning task l_j to an edge node, and B is the total bandwidth budget. Let Φ be the set of $\alpha_j, \forall j \in I = \{1, \dots, m\}$. For simplicity, we use $\mathcal{M}_j(\Phi)$ to denote the objective function for task l_j, which can be described as $\min_{\Phi} \theta(\Phi) = \max_{j \in I} \mathcal{M}_j(\Phi)$. Besides, we use $\mathcal{A}(\Phi)$ to denote the left part of the first set of inequalities in Eq. (20). Let $\mathcal{F}_t$ denote the left part of the second set of equations in epoch t. Eq. (20) can be transformed into a smooth constrained optimization problem as follows

$$\min \lambda$$
$$s.t. \begin{cases} \mathcal{M}_j(\Phi) \le \lambda, & \forall j, \Phi \\ \mathcal{A}(\Phi) \le B, & \forall \Phi \\ \mathcal{F}_t = \Phi \cdot n, & \forall \Phi, t \end{cases} \tag{21}$$

Due to the non-differentiability of the objective function $\theta(\Phi)$, it is difficult to solve such an optimization problem in Eq. (21) by the classical gradient methods directly. In this paper, we propose a sequence quadratic program based proportion assignment (SQP-PA) algorithm. For ease of description, we define $I(\Phi) = \{j \ |\mathcal{M}_j(\Phi) = \theta(\Phi)\}$ and $j_r = \min\{j : j \in I(\Phi^r)\}$, where Φ^r denotes the proportion assignment in the rth search round. We also use Θ^m to denote $(\alpha_1, \dots, \alpha_m)^{\mathbb{T}}$.

The complete SQP-PA algorithm is formally described in Algorithm 2. At the beginning, SQP-PA initializes some variables (Line 2–3). The initial value of Φ is composed of m stochastic values of α. Similar to [33], we set $\gamma \in (0, \frac{1}{2})$, and $\pi \in (2,3)$. Let $\mathcal{D}^r$ denote the search direction in the round r. We first compute the search direction (Line 5–10). The vector $(\lambda_r, \mathcal{D}^r)$ is computed by solving the following quadratic problem at Φ^r

$$\min \lambda + \frac{1}{2}\mathcal{D}^{\mathbb{T}} E_r \mathcal{D} \tag{22}$$

subject to $\mathcal{M}_j(\Phi^r) - \theta(\Phi^r) + \nabla\mathcal{M}_j(\Phi^r)^{\mathbb{T}}\mathcal{D} \le \lambda, \forall j$ and $\mathcal{A}(\Phi^r) \le B', \mathcal{F}_t = \Phi^r \cdot n, \forall t$, where $E_r \in \Theta^{m\times m}$ is the two-dimension matrix in the round r. We compute the auxiliary variable $\overline{\mathcal{M}}_{j,j_r}, \overline{\mathcal{M}}_{j_r}$ and the matrix U as follows

$$\begin{cases} \overline{\mathcal{M}}_{j,j_r}(\Phi^r) = \mathcal{M}_j(\Phi^r) - \mathcal{M}_{j_r}(\Phi^r), j \in J_r \setminus \{j_r\} \\ \overline{\mathcal{M}}_{j_r}(\Phi^r) = \{\overline{\mathcal{M}}_{j,j_r}(\Phi^r), j \in J_r \setminus \{j_r\}\} \\ U = \nabla\overline{\mathcal{M}}_{j_r}(\Phi^r) = (\nabla\mathcal{M}_j(\Phi^r) - \nabla\mathcal{M}_{j_r}(\Phi^r)) \end{cases} \tag{23}$$

where $J_r = \{j \ |\mathcal{M}_j(\Phi^r) + \nabla\mathcal{M}_j(\Phi^r)^{\mathbb{T}}\mathcal{D}^r - \theta(\Phi^r) - \lambda_r = 0\}$. Then, we start nonmonotone line search (Line 11) and compute the step size δ_r which is the first number of sequence $\{1, \frac{1}{2}, \frac{1}{4}...\}$ satisfying

$$\mathcal{M}(\Phi^r + \delta\mathcal{D}^r + \delta^2\widetilde{\mathcal{D}}^r) \le \mathcal{M}(\Phi^r) - \gamma\delta(\mathcal{D}^r)^{\mathbb{T}} E_r \mathcal{D}^r \tag{24}$$

Finally, the parameters and resource budget will be updated (Line 12–16).

3.2.1. The globally convergence of SQP-PA

Before the convergence analysis of SQP-PA, we make the following three assumptions [33].

Assumption 11. $\forall j \in I, \mathcal{M}_j(\Phi)$ is continuously differentiable.

Assumption 12. $\forall \Phi \in \Theta^m, (\nabla\mathcal{M}_j(\Phi), -1)^{\mathbb{T}}$ is linearly independent.

Assumption 13. There exist two constants $0 < \epsilon \le \rho$ such that $\epsilon\|\mathcal{D}\|^2 \le \mathcal{D}^{\mathbb{T}} E_r \mathcal{D} \le \rho\|\mathcal{D}\|^2, \forall \mathcal{D} \in R^m, \forall r$.

Lemma 14. *Let $(\lambda_r, \mathcal{D}^r)$ be the solution of Eq. (22). If $\mathcal{D}^r = 0$, then Φ^r is a K-T point which meets Kuhn–Tucker condition [34] of Eq. (21); Otherwise, we have*

$$\nabla \mathcal{M}_j(\Phi^r)^{\mathbb{T}}\mathcal{D}^r \le \lambda_r \le -\frac{1}{2}(\mathcal{D}^r)^{\mathbb{T}} E_r \mathcal{D}^r < 0, j \in I(\Phi^r).$$

Lemma 15. *If $\mathcal{D}^r \neq 0$, there exists r such that the line search yields a step size $\delta_r = (\frac{1}{2})^j$.*

Proof. According to Eq. (22), we have

$$\lambda_r + \frac{1}{2}(\mathcal{D}^r)^{\mathbb{T}} E_r \mathcal{D}^r \le 0 \Longrightarrow \lambda_r \le -\frac{1}{2}(\mathcal{D}^r)^{\mathbb{T}} E_r \mathcal{D}^r < 0$$

$\forall j \in I$, we define

$$\begin{aligned}\epsilon_r &= \mathcal{M}_j(\Phi^r + \delta\mathcal{D}^r + \delta^2\widetilde{\mathcal{D}}^r) - \theta(\Phi^r) + \gamma\delta(\mathcal{D}^r)^{\mathbb{T}} E_r \mathcal{D}^r \\ &\le (\gamma - \frac{1}{2})\delta(\mathcal{D}^r)^{\mathbb{T}} E_r \mathcal{D}^r + o(\delta)\end{aligned} \tag{25}$$

Thus, there exists some $\bar{\delta}_j > 0$ such that $\epsilon_r < 0$. Let $\bar{\delta} = \min\{\bar{\delta}_j, j \in I\}$, and the condition in Eq. (24) is satisfied for all $\delta \in [0, \bar{\delta}]$. In other words, the line search is always completed. □

Theorem 16. *SQP-PA either stops at a K-T point Φ^r of Eq. (21) in a finite number of rounds, or generates an infinite sequence Φ^r in which any accumulation point is a K-T point of Eq. (21).*

Proof. Assume that the SQP-PA algorithm generates an infinite sequence $\{\Phi^r\}$, and there exists a subsequence Ω such that the values of some variables tend to be optimal, *i.e.*, $\Phi^r \to \Phi^*, E_r \to E_*, \mathcal{D}^r \to \mathcal{D}^*, \lambda_r \to \lambda_*, \exists r \in \Omega$, where $\Phi^*, E_*, \mathcal{D}^*, \theta_*$ denote their optimal values, respectively. Then, we need to prove that $\mathcal{D}^* = 0$, *i.e.*, $\mathcal{D}^r \to 0, \exists r \in \Omega$. According to Eq. (24) and Lemma 14, we can derive that $\{\theta(\Phi^r)\}$ is monotonically decreasing. Hence, considering $\{\Phi^r\}_{r\in\Omega} \to \Phi^*$ and the continuity of $\theta(\Phi)$, we have

$$\lim_{r\to\infty} \theta(\Phi^r) = \theta(\Phi^*) \Longrightarrow \lim_{r\to\infty}(\theta\left(\Phi^{r+1}\right) - \theta\left(\Phi^r\right)) = 0 \tag{26}$$

Suppose by contradiction that $\mathcal{D}^* \neq 0$. Then $\lambda_* < 0$. According to the conclusion about the successful line search, the step size $\delta_r \ge 0$ on Ω. Combining Lemma 15, we have

$$\begin{aligned}\lim_{r\to\infty}(\theta\left(\Phi^{r+1}\right) - \theta\left(\Phi^r\right)) &\le -\gamma\delta_r\left(\mathcal{D}^r\right)^{\mathbb{T}} E_r \mathcal{D}^r \\ &\le -\frac{1}{2}\gamma\delta_*\left(\mathcal{D}^*\right)^{\mathbb{T}} E_* \mathcal{D}^* < 0\end{aligned} \tag{27}$$

It is obvious that Eqs. (26) and (27) contradict. It follows that $\mathcal{D}^* = 0$, and $\mathcal{D}^r \to 0, \exists r \in \Omega$. Thus, QP-PA can converge globally and achieve the optimal result. □

4. Performance evaluation

This section first introduces the two benchmarks and several metrics for performance comparison (Section 4.1). We then describe some settings of evaluation, and give the extensive evaluation results in Section 4.2. Finally, we implement our algorithm on the small-scale edge computing platform, and give the testing results in Section. 4.3.

4.1. Benchmarks and performance metrics

We choose two typical algorithms as benchmarks for performance comparison. The first one, called ADP-FL [35], belongs to the synchronous FL scheme. In an epoch, the server can adaptively determine the number of local updates via linear search to minimize the loss function under a given resource budget. The second one, called AFO [20], is an asynchronous federated optimization algorithm with provable convergence, in which the parameter server will perform the global update on receiving only one local update from an arbitrary worker. Thus, we choose these two algorithms, the synchronous scheme ($\alpha = 1$) and the asynchronous scheme ($\alpha = \frac{1}{n}$), as benchmarks in our work.

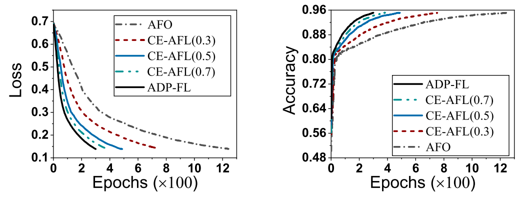

Fig. 4. Loss and Accuracy with Logistic-R over MNIST.

In order to evaluate the performance of training models, we adopt four performance metrics. (1) *Loss function* is the quantification difference of probability distributions between model output and observation results. The loss value reflects the quality of model learning and whether convergence has been achieved or not as described in Section 2.2. (2) As one of the most common performance metrics for classification, *accuracy* is measured by the proportion between the amount of the right data by the model and that of all data. (3) We adopt the *training time* to estimate the training speed of a learning task. (4) When there are multiple learning tasks in the network, we measure the *maximum loss* and *minimum accuracy* of all tasks to evaluate the training performance.

4.2. Simulation evaluation

4.2.1. Evaluation settings

The simulations are conducted on an AMAX deep learning workstation[1] (CPU: Intel(R) E5-2620v4, GPU:NVIDIA GeForce RTX 2080Ti), where we build an FL simulation environment and implement all models which are listed with PySyft [36], a Python library for privacy-preserving deep learning including FL, under the PyTorch framework.

(I) Models and Datasets: We evaluate the training process with three different models over five different datasets, which represent a large variety of both small and large models and datasets. We adopt three models, linear regression (referred to as Linear-R in short), logistic regression (Logistic-R), and deep convolutional neural networks (CNN[2]), respectively.

Linear-R is trained over the energy dataset [37], which contains 14,803 training data and 4,932 testing data from a wireless sensor network. This model can predict the energy consumption of appliances, including some environmental parameters (*e.g.*, temperature and humidity) and one sub-metered electrical energy consumption (*e.g.*, lights).

Logistic-R is performed over the original MNIST dataset [38] (referred to as MNIST), which contains gray-scale images of 70,000 handwritten digits (60,000 for training and 10,000 for testing). This model outputs a binary label that corresponds to whether the digit is even or odd.

CNN is trained over three different datasets. The first one is the fashion MNIST dataset [39] (referred to as FMNIST), which has 70,000 images of fashion items (60,000 for training and 10,000 for testing)

[1] https://www.amax.com/products/gpu-platforms/.

[2] The detailed CNN network architectures in our experiments. (1) 9 layers for MNIST: $5\times5\times32$ Convolutional → Local Response Normalization → 2×2 MaxPool → $5\times5\times64$ Convolutional → Local Response Normalization → 2×2 MaxPool → 1600×512 Fully connected → 512×10 Fully connected → Softmax. (2) 10 layers for CIFAR10 and CIFAR100: $5\times5\times64$ Convolutional → Local Response Normalization → 2×2 MaxPool → $5\times5\times128$ Convolutional → Local Response Normalization → 2×2 MaxPool → 3200×512 Fully connected → 512×256 Fully connected → 256×10 Fully connected → Softmax.

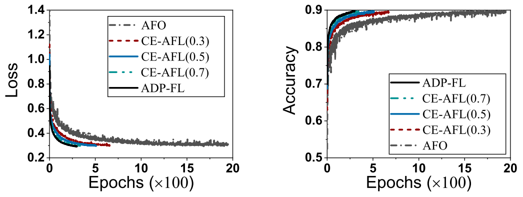

Fig. 5. Loss and accuracy with CNN over FMNIST.

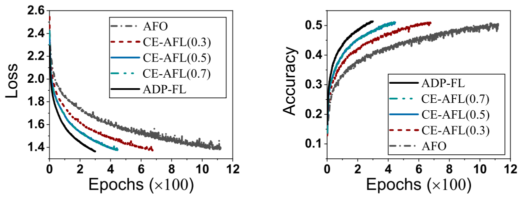

Fig. 6. Loss and accuracy with CNN over CIFAR10.

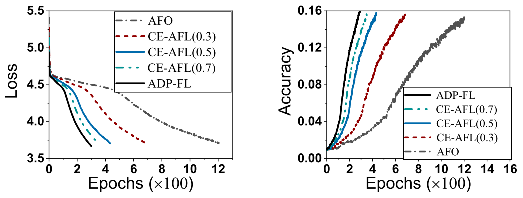

Fig. 7. Loss and accuracy with CNN over CIFAR100.

with the same format as MNIST. The second one is the CIFAR10 dateset, which includes 60,000 color images (50,000 for training and 10,000 for testing) of 10 different types. The third one is the CIFAR100 dateset, which includes 60,000 color images with 100 different types for training and testing [40].

(II) Network Resources: In the simulations, we mainly focus on the bandwidth resource cost in edge computing. Specifically, the bandwidth consumption can be measured by the size of the model parameters. We train some models under a fixed amount of resource budget (*e.g.*, network bandwidth and training time). In order to implement the resource efficient asynchronous federated learning, we set the value of parameters $\eta = 0.01, \varsigma = 0.3$ and estimate the values of parameters L, μ in real time according to [35].

The model training is quite time-consuming, even with GPU which can speed up the training process compared with CPU [41]. As suggested in [42], in order to efficiently simulate the training processing in FL of our proposed solution and benchmarks, a total of 100 edge nodes are generated in the simulation, and 10 of them are randomly activated to participate in the model training. The solution can be easily extended to the case of more edge nodes. Besides, we distribute the data among these edge nodes according to Gaussian distribution with both the fixed mean and the variance of 0.5. For ease of presentation and interpretation of results, CE-AFL performs only one local update between two global updates, *i.e.*, $D = 1$. We repeat each simulation 10 times and compute the average results.

4.2.2. *Simulation results*

(I) **Single Learning Task:** The first set of simulations evaluates the performance of the classification models (*e.g.*, Logistic-R and CNN) without resource constraints. We train each model using ADP-FL with 300 epochs. Three different values of α (*e.g.*, 0.3, 0.5 and 0.7) are adopted in CE-AFL for model training. Without confusion, we denote as CE-AFL(α). As shown in Figs. 4–7, AFO will run averagely 4 times as many epochs as ADP-FL to achieve the similar performance (*e.g.*, loss function values or classification accuracy). Besides, with the decreasing value of α, the number of training epochs by CE-AFL will increase gradually to achieve the similar performance of ADP-FL. The number of training epochs by CE-AFL is significantly fewer than that of AFO. For example, with CNN training over the FMNIST dataset, the number of required epochs by CE-AFL ($\alpha = 0.3$) and AFO is about 590 and 1880, respectively.

We also test the training time of these models. Since ADP-FL should wait for the local updates from all workers in each epoch, it takes a longer time than CE-AFL. In Fig. 8, the training time of CE-AFL changes with α and is less than that of AFO ($\alpha = 0.1$) and ADP-FL ($\alpha = 1$). For example, when CE-AFL adopts the optimal value of α, with CNN training over the FMNIST dataset, the minimum training time of CE-AFL is about 750 s. However, the training time of AFO and ADP-FL is about 2690 s and 2580 s. Thus, CE-AFL can reduce the training time by about 72% and 70% compared with AFO and ADP-FL, respectively.

In the second set of simulations, we run the Linear-R model over the energy dataset. For performance evaluation, we compare the MSE performance for different algorithms. As shown in the left plot of Fig. 9, ADP-FL can achieve the smallest MST among all solutions. However, from the results in the right plot of Fig. 9, we observe that the training time of CE-AFL is less than the other two benchmarks. Specifically, CE-AFL can reduce the training time by about 58% and 75% compared with AFO and ADP-FL, respectively.

(II) **Multiple Learning Tasks:** The third set of simulations observes the performance of multiple learning tasks without resource constraints in the system. We run two models over the different datasets simultaneously, including Logistic-R over MNIST, CNN over FMNIST and CIFAR10. Each model training task performs 300 epochs. Fig. 10 shows that the maximum loss and minimum accuracy of the three groups of tasks. By this figure, ADP-FL can achieve the best performance of loss and accuracy among the three solutions. When adopting the optimal Φ for each task by SQP-PA, CE-AFL achieves a little worse performance (*e.g.*, loss or accuracy) than ADP-FL, but better than AFO. For example, given 300 training epochs, the loss of CE-AFL is about 1.445 and that of ADP-FL is about 1.338. Accordingly, the accuracy of ADP-FL and CE-AFL is about 51% and 49%, respectively.

In the last set of simulations, we test the performance of multiple learning tasks with a limited training time budget. In practice, some training tasks often need to be completed within the specified time. We first consider the training time constraint for the three solutions. As shown in Fig. 11, the maximum loss becomes smaller and the minimum accuracy becomes higher by changing the training time constraint from 300 s to 3000 s for all three algorithms. The proposed CE-AFL framework can achieve less loss and higher accuracy compared with the other two benchmarks. For example, when the time budget is 1500 s, the minimum accuracy of CE-AFL is about 37%, while that of ADP-FL and AFO is only about 29% and 19%, respectively. Therefore, CE-AFL

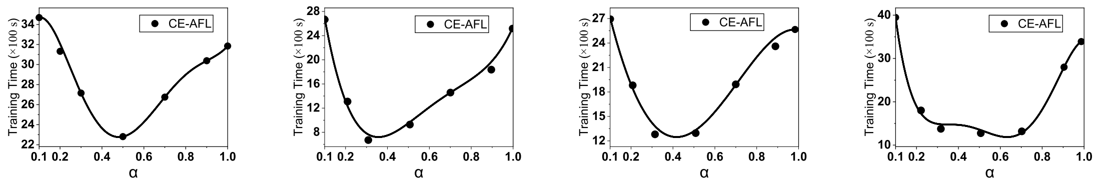

Fig. 8. Training time of different tasks, including Logistic-R over MNIST, CNN over FMNIST, CIFAR10 and CIFAR100.

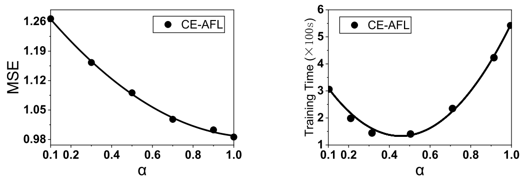

Fig. 9. MSE and training time with linear-R on the energy dataset.

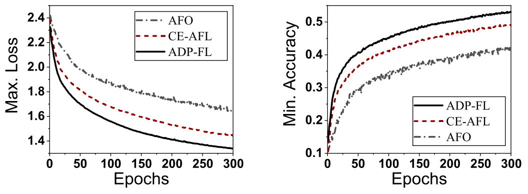

Fig. 10. Max. Loss and Min. Accuracy for multi-learning tasks.

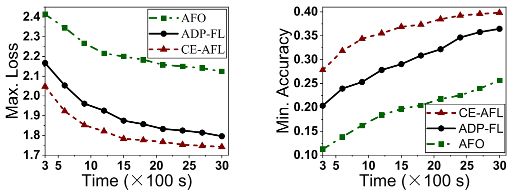

Fig. 11. Max. Loss and Min. accuracy with training time budget for multi-learning tasks.

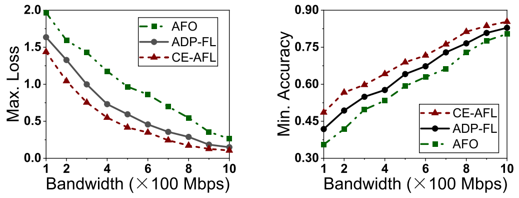

Fig. 12. Max. Loss and Min. accuracy with bandwidth budget for multi-learning tasks.

can improve the minimum accuracy by about 8% and 18% compared with ADP-FL and AFO, respectively.

We further observe the performance of multiple learning tasks with a limited bandwidth budget. The communication between the parameter server and the workers will emerge huge consumption of network bandwidth. We test the three training tasks by changing the bandwidth constraint from 100 Mbps to 1000 Mbps. As shown in Fig. 12, CE-AFL can achieve significantly higher minimum accuracy than both AFO and ADP-FL. For example, when the bandwidth constraint is 800Mbps, the minimum accuracy of the three training tasks by using CE-AFL is about 81%, while that of AFO and ADP-FL is about 72% and 76%. Thus, the proposed CE-AFL framework can improve the minimum accuracy by about 9% and 5% compared with AFO and ADP-FL, respectively. These results show that CE-AFL can significantly improve the classification accuracy compared with two benchmarks under resource constraints.

In conclusion, our proposed CE-AFL can reduce the training time by about 70% compared with the existing scheme in the single learning task. Moreover, CE-AFL can improve the maximum loss value and minimum accuracy compared with AFO and ADP-FL.

4.3. Test-bed evaluation

4.3.1. Implementation on the platform

We implement the AFO, ADP-FL and CE-AFL algorithms on a real small-scale test-bed shown in Fig. 13, which is composed of two main parts: a deep learning workstation with four NVIDIA GeForce RTX Titan GPUs and 10 Jetson TX2 development kits[3] (CPU: ARMv8 Cortex-A57, RAM: 8GB). Specifically, the workstation acts as the parameter server which is responsible for the model aggregation and verifying the global model. We adopt a Jetson TX2 developer as a worker to locally train the model and send the updates to the server for aggregation. We develop a distributed model training framework with pytorch. The workers and the parameter server are physically connected through wireless network in the same router. Besides, they are logically connected through *torch.distributed* package (*gloo* back-end). Specifically, the IP address of the server and a designated port are combined to establish a connection between the server and the worker through TCP protocol. After the connection is established, the server segments the training and testing datasets, and sends the segmentation results to each worker. After receiving the results, the worker generates the local dataset for training. All our code is publicly available at github.[4]

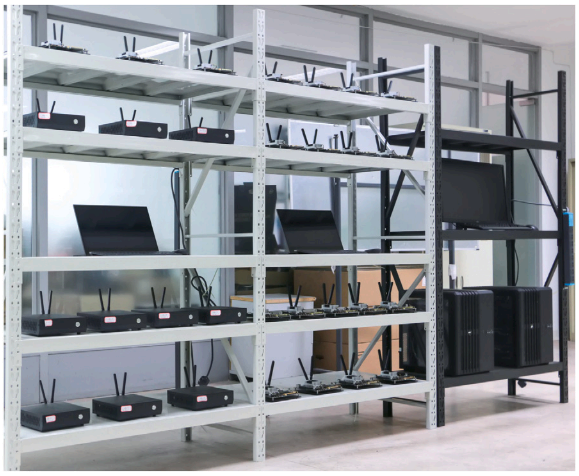

Fig. 13. The Test-bed Platform.

Two CNN models with different types and structures are implemented for the CIFAR10 and FMNIST on the test-bed, respectively. The first CNN model is used for the CIFAR10 dataset. It has two 5×5 convolution layers (64, 64 channels, each followed by 3×3 max pooling), two dense layers with 384 and 192 units and a softmax output layer with 10 units. The second CNN model which has two 5×5 convolution layers (32, 64 channels, each followed by 2×2 max pooling), a dense layer with 1024 units and a softmax output layer with 10 units (related to the 10 classes in FMNIST) is used for the FMNIST dataset.

In the test-bed, we mainly consider the impact of different data distribution, including quantity and category, among workers on the performance of model training. First, the amount of data which varies significantly with time and space on the workers is often imbalanced. Thus, we adopt three different cases of data distribution to simulate the data imbalance. (1) Case 1: We allocate the same amount of training data (*e.g.*, 6,000) among the ten workers; (2) Case 2: There is little difference in the amount of data between different workers (*e.g.*, 4000–8000); (3) Case 3: The amount of data on these workers varies greatly (*e.g.*, 1000–11,000). Second, different categories of data distribution, *i.e.* I.I.D. and non-I.I.D. data, among the workers also emerge different effects on model training. For example, in the case of I.I.D., each worker has all categories of data samples (*e.g.*, 10 classes), but in the case of non-I.I.D., each worker may have only part of categories (*e.g.*, 5 classes). We adopt four different cases to verify the effects of data

[3] https://developer.nvidia.com/embedded/jetson-tx2-developer-kit.

[4] https://github.com/lyl617/CE-AFL.

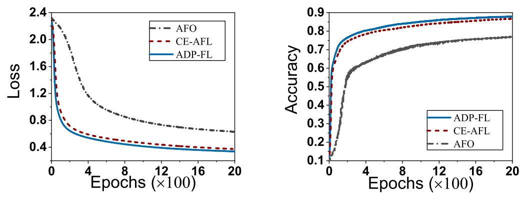

Fig. 14. Loss and accuracy with CNN over FMNIST in Test-bed.

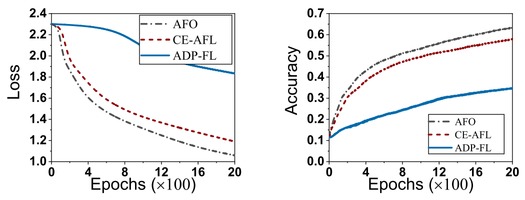

Fig. 15. Loss and accuracy with CNN over CIFAR10 in Test-bed.

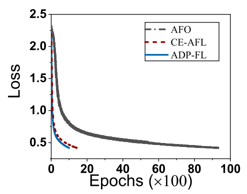

Fig. 16. Loss vs. Epochs with Balanced Data in Test-bed.

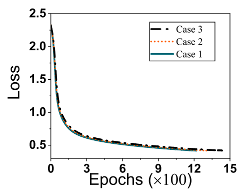

Fig. 17. Loss vs. Epochs under Case 1–3 with I.I.D. Data in Test-bed.

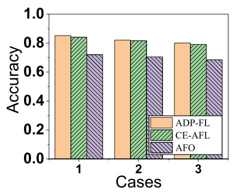

Fig. 18. Training accuracy under Cases 1–3 in Test-bed.

distribution on model training, including Case (I): Each data sample is randomly assigned to a worker, thus each worker has uniform (but not full) information, *i.e.*, I.I.D. data; Case (II): Each worker has 5 categories of data samples; Case (III): Each worker has 2 categories of data samples; Case (IV): Each worker only has 1 categories of data samples. The data samples in cases II-IV are non-I.I.D., and the degree of nonuniformity of data distribution increases gradually.

4.3.2. Testing results

In the first set of experiments, we test the balanced and uniform data with CNN training over FMNIST and CIFAR10, respectively. We run two groups of experiments 2000 training epochs. As shown in the Figs. 14–15, the training performance (*i.e.*, loss and accuracy) of CE-AFL (α = 0.7) is very close to that of ADP-FL and much better than that of AFO. For example, given 2000 epochs in CNN training over FMNIST dataset, the loss value of CE-AFL is 0.3737, while that of ADP-FL and AFO is

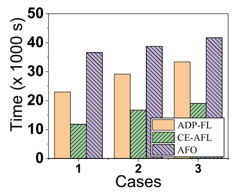

Fig. 19. Training time under Cases 1–3 in Test-bed.

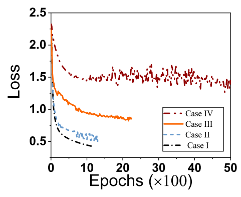

Fig. 20. Loss vs. Epochs under Case I-IV with balanced data in Test-bed.

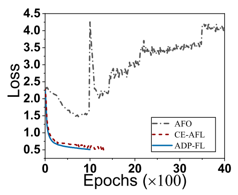

Fig. 21. Loss vs. Epochs under Case II in Test-bed.

0.3382 and 0.6296, respectively. Accordingly, the training accuracy of CE-AFL is about 86.8%, and the accuracy of ADP-FL and AFO is about 87.8% and 76.9%, respectively. Thus, our proposed mechanism can improve the training accuracy by about 10% compared with AFO.

In the second set of experiments, we observe the performance of model training (CNN over FMNIST) under three different amount of data distribution cases (cases 1–3). In each case, we run the ADP-FL algorithm with 1000 training epochs as baseline. Fig. 16 shows that more training epochs (about 1435) are needed by CE-AFL to reach the loss value of the baseline under case 1. That is because the server only aggregates the updated local model from the arbitrary one worker at a time in AFO. Moreover, AFO needs run 9328 training epochs to reach the same performance of training loss. In other words, AFO needs 9× training epochs compared with ADP-FL, while CE-AFL only needs 1.5× epochs compare with the baseline. The training loss of CE-AFL under the three different cases is shown in Fig. 17. The results show that the experiment under case 3 needs a few more training epochs than that of case 1 and case 2 to reach the same performance (*e.g.*, loss).

We also observe the training accuracy and time under the cases 1–3 with the three solutions. Fig. 18 shows the training accuracy of each case after 1000 training epochs. In CE-AFL, more updated models from the workers are involved in the model aggregation than AFO in each epoch. In each case, CE-AFL always achieves better performance of accuracy compared with AFO, and achieves similar performance with that of ADP-FL. As shown in Fig. 19, our proposed mechanism achieves the minimum training time while reaching the same training performance (loss and accuracy) of baseline compared with the other two benchmarks. For example, in case 1, the training time of CE-AFL is about 11,835 s, while that of ADP-FL and AFO is about 22,957 s and 36,587 s, respectively. In other words, CE-AFL can reduce the training time about 48.4% and 67.9% compared with ADP-FL and AFO, respectively.

The last set of experiments tests the performance of model training (CNN over FMNIST) under four different categories of data distribution cases I-IV. We first test the training performance of CE-AFL under four different cases of data categories distribution (cases I-IV). Fig. 20 shows

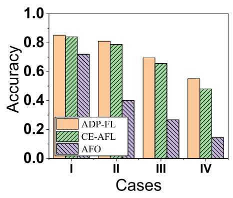

Fig. 22. Training Accuracy under Cases I-IV in Test-bed.

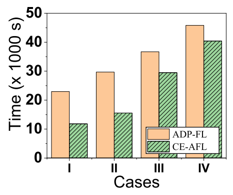

Fig. 23. Training time under Cases I-IV in Test-bed.

that the distribution of data categories will emerge the effects on the speed of model training. For example, the training loss of the experiment under case IV by running 5,000 epochs is about 1.3834, while that of case II by running 1300 epochs is about 0.5042. In other words, the training performance with non-I.I.D. data is worse than that of I.I.D. data. Then, we test the training performance with case II. As shown in Fig. 21, a little more training epochs (about 1310) are run by CE-AFL to reach the same loss value of ADP-FL (1000 epochs). However, the loss value of AFO greatly fluctuates and gradually increases during the training. Thus, the solution AFO cannot well handle non-I.I.D. training data, but our proposed CE-AFL can well handle it.

Besides, the training performances (accuracy and time) of three solutions under cases I-IV are shown in Figs. 22–23. Let the training epochs be 1000, we test the training accuracy under cases I-IV. As shown in Fig. 22, the training accuracy achieved by CE-AFL is always better than AFO in each case. For example, the accuracy of CE-AFL is about 78.8%, while the accuracy of ADP-FL and AFO is about 79.7% and 40.1%, respectively. In other words, our proposed mechanism can improve the training accuracy by about 39% compared with AFO. We then test the training time of ADP-FL and CE-AFL under cases I-IV. Our proposed solution can efficiently avoid the straggler problem caused by ADP-FL. Fig. 23 shows that the training time required by CE-AFL is always less than that of ADP-FL in each case. For example, the required training time by CE-AFL is about 15,562 s in case II, while that of ADP-FL is about 29,684 s. Thus, CE-AFL can reduce the training time by about 47.6% compared with ADP-FL under Case II.

5. Related works

Recently, federated learning (FL) has been widely mentioned and studied in both academia and industry fields.

One research area related to FL is distributed machine learning (DML) through the use of worker machines and parameter servers [7]. Bao *et al.* [43] propose an online algorithm for scheduling the arriving jobs and deciding the numbers of concurrent workers and parameter servers for each job over its course, so as to maximize the overall utility of all jobs. Ho *et al.* [44] design a parameter server system which maximized the time computational workers spend doing useful work on ML algorithms for DML, which followed a Stale Synchronous Parallel (SSP) model of computation. The authors [41] propose a parameter server based distributed computing framework for training large-scale deep neural networks. Besides, the authors introduce a new learning rate modulation strategy to counter the effect of stale gradients and proposed a new synchronization protocol that could effectively bound the staleness in gradients, improve runtime performance and achieve good model accuracy.

The above works mainly study efficient solutions of DML in datacenters. Under this scenario, shared storage is usually adopted. But in edge computing, no storage is shared among edge nodes. The worker machines will not keep persistent data storage, but fetch the data from the shared storage at the beginning of the learning process. As a result, the data samples on different workers are usually I.I.D. in datacenters.

In federated learning, the data are collected at the edge directly and stored persistently at edge nodes, thus the data distribution at different edge nodes is usually non-I.I.D. and imbalanced, which is different from DML in datacenters [24]. Smith *et al.* [12] show that multi-task learning is naturally suited to handle the statistical challenges of this setting, and propose a novel systems-aware optimization method that was robust to practical systems issues. Our method and theory considered issues of high communication cost, stragglers, and fault tolerance for distributed multi-task learning. The authors [45] proposes an asynchronous distributed machine learning framework based on the emerging serverless architecture, with which stateless functions can be executed in the cloud without the complexity of building and maintaining virtual machine infrastructures. Shi *et al.* [46] merge some short communication tasks into a single one to reduce the overall communication time and formulated an optimization problem to minimize the training iteration time. The authors [47] introduce a new and increasingly relevant setting for distributed optimization in machine learning, where the data for the optimization of training are distributed over an extremely large number of nodes. However, most of these solutions ignore the impact of limited resource constraints on training performance, which may consume massive resources on edge computing systems. [20] proposes a new asynchronous federated optimization algorithm. We prove that the proposed approach has near-linear convergence to a global optimum, for both strongly and non-strongly convex problems, as well as a restricted family of non-convex problems.

Last but not least, some works [6,15] similar to our research will be introduced. The authors [6] perform FL efficiently while actively managing workers based on their resource conditions. Specifically, the proposed solution solves a worker selection problem with resource constraints, which allows the server to aggregate as many local updates as possible and to accelerate performance improvement in ML models. Wang [15] propose an experience-driven control framework that intelligently chooses the workers to participate in each round of federated learning to counterbalance the bias introduced by non-IID data and to speed up convergence of model training. However, after selecting the subset of workers to participate in the model training, the parameter server only perform model aggregation while receiving all local updates from these workers. In other words, the *synchronous scheme* is adopted by these works for global updating on the server. Compared with our proposed asynchronous scheme, these researches cannot solve synchronization barrier problem which will lead to longer training time and worse training performance under given resource budget.

To our best knowledge, we are the first to address the problem of determining the number of received local updates from the workers to optimize the training performance of learning tasks, with a given resource budget for federated learning in edge computing systems.

6. Conclusion

In this paper, we propose the communication-efficient asynchronous federated learning (CE-AFL) mechanism for edge computing. We analyze the convergence bound of CE-AFL. The asynchronous federated learning with resource constraints (AFL-RC) problem is formulated for minimizing the loss function of model training. We then design efficient algorithms to determine the optimal values of parameters α in CE-AFL for single learning task and multiple learning tasks, respectively. The simulation and experimental results show CE-AFL can achieve significantly higher accuracy and less training time under resource constraints, compared with the existing solutions. The problem of dynamic scenario is the focus of our next study, which can be easily solved by experience-driven method.

CRediT authorship contribution statement

Jianchun Liu: Conceptualization, Methodology, Writing – original draft. **Hongli Xu:** Methodology, Writing – review & editing. **Yang Xu:** Formal analysis, Writing – review & editing. **Zhenguo Ma:** Software, Validation. **Zhiyuan Wang:** Data curation,Validation. **Chen Qian:** Writing – review & editing. **He Huang:** Writing – review & editing.

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.

Acknowledgments

This article is supported in part by the National Science Foundation of China (NSFC) under Grants 62132019, 61936015, 62102391 and U1709217.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: Esann, 2013.
[2] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, K. Huang, Towards an intelligent edge: Wireless communication meets machine learning, 2018, arXiv preprint arXiv:1809.00343.
[3] M. Satyanarayanan, The emergence of edge computing, Computer 50 (1) (2017) 30–39.
[4] H.B. McMahan, D. Ramage, 2017, http://www.googblogs.com/federated-learning-collaborative-machine-learning-without-centralized-training-data/, Google.
[5] X. Wei, Q. Li, Y. Liu, H. Yu, T. Chen, Q. Yang, Multi-agent visualization for explaining federated learning, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, AAAI Press, 2019, pp. 6572–6574.
[6] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019-2019 IEEE International Conference on Communications, ICC, IEEE, 2019, pp. 1–7.
[7] M. Li, D.G. Andersen, J.W. Park, A.J. Smola, A. Ahmed, V. Josifovski, J. Long, E.J. Shekita, B.-Y. Su, Scaling distributed machine learning with the parameter server, in: 11th {USENIX} Symposium on Operating Systems Design and Implementation, {OSDI} 14, 2014, pp. 583–598.
[8] M. Li, D.G. Andersen, A.J. Smola, K. Yu, Communication efficient distributed machine learning with the parameter server, in: Advances in Neural Information Processing Systems, 2014, pp. 19–27.
[9] N. Wang, B. Varghese, M. Matthaiou, D.S. Nikolopoulos, ENORM: A framework for edge node resource management, IEEE Trans. Serv. Comput. (2017).
[10] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[11] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, X. Shen, Internet of vehicles in big data era, IEEE/CAA J. Autom. Sin. 5 (1) (2017) 19–35.
[12] V. Smith, C.-K. Chiang, M. Sanjabi, A.S. Talwalkar, Federated multi-task learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4424–4434.
[13] 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, 2017, pp. 1273–1282.
[14] 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.
[15] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-IID data with reinforcement learning, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 1698–1707.
[16] W. Dai, A. Kumar, J. Wei, Q. Ho, G. Gibson, E.P. Xing, High-performance distributed ML at scale through parameter server consistency models, in: Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[17] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, 2018, arXiv preprint arXiv:1812.06127.
[18] Y. Chen, Y. Ning, H. Rangwala, Asynchronous online federated learning for edge devices, 2019, arXiv preprint arXiv:1911.02134.
[19] Y. Lu, X. Huang, Y. Dai, S. Maharjan, Y. Zhang, Differentially private asynchronous federated learning for mobile edge computing in urban informatics, IEEE Trans. Ind. Inf. (2019).
[20] C. Xie, S. Koyejo, I. Gupta, Asynchronous federated optimization, 2019, arXiv preprint arXiv:1903.03934.
[21] 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.
[22] R. Christensen, Log-Linear Models and Logistic Regression, Springer Science & Business Media, 2006.
[23] J.A. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.
[24] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, 2018, arXiv preprint arXiv:1806.00582.
[25] S. Ruder, An overview of gradient descent optimization algorithms, 2016, arXiv preprint arXiv:1609.04747.
[26] L. Bottou, Stochastic gradient descent tricks, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 421–436.
[27] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, T.-Y. Liu, Asynchronous stochastic gradient descent with delay compensation, in: International Conference on Machine Learning, 2017, pp. 4120–4129.
[28] P. Raghavan, C.D. Tompson, Randomized rounding: a technique for provably good algorithms and algorithmic proofs, Combinatorica 7 (4) (1987) 365–374.
[29] R. Kanwal, K. Liu, A Taylor expansion approach for solving integral equations, Internat. J. Math. Ed. Sci. Tech. 20 (3) (1989) 411–414.
[30] L. Jacob, J.-p. Vert, F.R. Bach, Clustered multi-task learning: A convex formulation, in: Advances in Neural Information Processing Systems, 2009, pp. 745–752.
[31] Y. Zhang, D.-Y. Yeung, A convex formulation for learning task relationships in multi-task learning, 2012, arXiv preprint arXiv:1203.3536.
[32] A.R. Gonçalves, F.J. Von Zuben, A. Banerjee, Multi-task sparse structure learning with gaussian copula models, J. Mach. Learn. Res. 17 (1) (2016) 1205–1234.
[33] Z. Zhu, X. Cai, J. Jian, An improved SQP algorithm for solving minimax problems, Appl. Math. Lett. 22 (4) (2009) 464–469.
[34] M.A. Hanson, On sufficiency of the Kuhn–Tucker conditions, J. Math. Anal. Appl. 80 (2) (1981) 545–550.
[35] 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 INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 63–71.
[36] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, J. Passerat-Palmbach, A generic framework for privacy preserving deep learning, 2018, arXiv preprint arXiv:1811.04017.
[37] L.M. Candanedo, V. Feldheim, D. Deramaix, Data driven prediction models of energy use of appliances in a low-energy house, Energy Build. 140 (2017) 81–97.
[38] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al., Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[39] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.
[40] A. Krizhevsky, G. Hinton, et al., Learning Multiple Layers of Features from Tiny Images, Tech. Rep., Citeseer, 2009.
[41] S. Gupta, W. Zhang, F. Wang, Model accuracy and runtime tradeoff in distributed deep learning: A systematic study, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, IEEE, 2016, pp. 171–180.
[42] J. Konečnỳ, H.B. McMahan, D. Ramage, P. Richtárik, Federated optimization: Distributed machine learning for on-device intelligence, 2016, arXiv preprint arXiv:1610.02527.
[43] Y. Bao, Y. Peng, C. Wu, Z. Li, Online job scheduling in distributed machine learning clusters, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 495–503.
[44] Q. Ho, J. Cipar, H. Cui, S. Lee, J.K. Kim, P.B. Gibbons, G.A. Gibson, G. Ganger, E.P. Xing, More effective distributed ml via a stale synchronous parallel parameter server, in: Advances in Neural Information Processing Systems, 2013, pp. 1223–1231.
[45] H. Wang, D. Niu, B. Li, Distributed machine learning with a serverless architecture, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 1288–1296.
[46] S. Shi, X. Chu, B. Li, MG-WFBP: Efficient data communication for distributed synchronous SGD algorithms, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 172–180.
[47] J. Konečnỳ, B. McMahan, D. Ramage, Federated optimization: Distributed optimization beyond the datacenter, 2015, arXiv preprint arXiv:1511.03575.

Jianchun Liu received B.S. degree in 2017 from the North China Electric Power University. He is currently a Ph.D. candidate in the School of Data Science, University of Science and Technology of China (USTC). His main research interests are software defined networks, network function virtualization, edge computing and federated learning.

Hongli Xu received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2002 and 2007, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or coauthored over 70 papers, and held about 30 patents. His main research interest is software-defined networks, cooperative communication, and vehicular ad hoc network.

Yang Xu is currently an associate researcher in the School of Computer Science and Technology at University of Science and Technology of China. He got his Ph.D. degree in computer science and technology from University of Science and Technology of China in 2019. He got his B.S. degree in Wuhan University of Technology in 2014. His research interests include Ubiquitous Computing, Deep Learning and Mobile Edge Computing.

Zhenguo Ma received the B.S. degree in software engineering from the Shandong University, China, in 2018. He is currently pursuing his Ph.D. degree in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include edge computing and federated learning.

Zhiyuan Wang received the B.S. degree from the Jilin University in 2019. He is currently studying for a master's degree in the School of Computer Science, University of Science and Technology of China (USTC). His main research interests are edge computing, deep learning and federated learning.

Chen Qian received the B.S. degree from Nanjing University in 2006, the M.Phil. degree from The Hong Kong University of Science and Technology in 2008, and the Ph.D. degree from The University of Texas at Austin in 2013, all in computer science. He is currently an Assistant Professor with the Department of Computer Engineering, University of California at Santa Cruz. His research interests include computer networking, network security, and Internet of Things. He has authored over 60 research papers in highly competitive conferences and journals. He is a member of the ACM.

He Huang Dr. He Huang is an associate professor in the School of Computer Science and Technology at Soochow University, P.R. China. He received his Ph.D. degree in Department of Computer Science and Technology from University of Science and Technology of China (USTC), in 2011. His current research interests include traffic measurement, spectrum auction, privacy preserving in auction, and algorithmic game theory. He is a Member of both IEEE and ACM.