FedLC: Accelerating Asynchronous Federated Learning in Edge Computing

Yang Xu, *Member, IEEE,* Zhenguo Ma, Hongli Xu, *Member, IEEE,* Suo Chen, Jianchun Liu, Yinxing Xue

Abstract—Federated Learning (FL) has been widely adopted to process the enormous data in the application scenarios like Edge Computing (EC). However, the commonly-used synchronous mechanism in FL may incur unacceptable waiting time for heterogeneous devices, leading to a great strain on the devices' constrained resources. In addition, the alternative asynchronous FL is known to suffer from the model staleness, which will lead to performance degradation of the trained model, especially on *non-i.i.d.* data. In this paper, we design a novel asynchronous FL mechanism, named FedLC, to handle the *non-i.i.d.* issue in EC by enabling the local collaboration among edge devices. Specifically, apart from uploading the local model directly to the server, each device will transmit its gradient to the other devices with different data distributions for local collaboration, which can improve the model generality. We theoretically analyze the convergence rate of FedLC and obtain the quantitative relationship between convergence bound and local collaboration. We design an efficient algorithm utilizing demand-list to determine the set of devices receiving gradients from each device. To handle the model staleness, we further assign different learning rates for various devices according to their participation frequency. The extensive experimental results demonstrate the effectiveness of our proposed mechanism.

Index Terms—*Asynchronous Federated Learning, Edge Computing, Non-i.i.d., Local Collaboration.*

1 Introduction

With the rapid proliferation of mobile devices in Internet of Things (IoT), more and more data are accumulated at the network edge (*e.g.*, gateway, switch) [1]–[3]. By preventing from centralizing the raw data, the emerging Federated Learning (FL) offers the potential to reduce the privacy leakage of devices' data in the application scenarios like Edge Computing (EC) [4], [5]. Following the commonly-used parameter server (PS) framework [6], there are usually one or multiple servers and a set of edge devices. FL solves the training tasks by the loose federation of edge devices. Concretely, each device maintains a device-specific local model based on its locally stored dataset. The global model is produced and updated on the server by aggregating the local models from edge devices.

For pursuing efficient FL in EC, we are confronted with several critical challenges. (i) *Heterogeneous devices*. The participating devices may be various types of edge devices (*e.g.*, smartphones, vehicles), which are equipped with diverse computation capacities, data size and network connections [5], [7]–[10]. As a consequence, the time required to update the local model and receive/upload models for heterogeneous devices may vary significantly. (ii) *Constrained resources*. In EC, the computation and communication resources of edge devices are always constrained [11]–[14]. Due to the periodic computation of local gradients as well as the frequent transmission between the server and devices, the FL model training will incur an enormous resource cost. (iii) *Non-i.i.d. data*. The device users may come from various communities, and pose different interests or requirements. As a consequence, the local data produced by various edge devices usually are not independent-and-identically-distributed (*i.e.*, *non-i.i.d.*) [4]. In other words, the local data on each device may fail to represent the overall distribution.

Generally, the synchronous FL is developed as the most popular mechanism for distributed model training [4], [11], [15], [16], and it is reported to be an important factor for ensuring system stability [17], since all the devices are required to upload their local models to the server at each epoch. However, regarding the heterogeneous capacities of edge devices, the synchronous FL may cause synchronization barrier, *i.e.*, the fast devices with high-performance hardware should wait for the slow devices (stragglers), which will incur unacceptable waiting time for the fast devices [9], [18]. Regarding the synchronization barrier, more time is required for the global model to reach the convergence. Concurrently, the synchronization barrier also makes the computation resources in synchronous FL system under low utilization, which leads to a great strain on devices' constrained resources. Moreover, the server in synchronous FL may become the system bottleneck due to the frequent model transmission with all participating devices [19], [20].

In order to overcome the obstacles of synchronous FL, asynchronous FL is proposed [13], [21]. To be specific, asynchronous FL can be divided into two categories: traditional asynchronous FL [21]–[23] and semi-asynchronous FL [12], [13], [24]. In the former category, the server updates the global model as long as it receives one local model from an arbitrary device. For example, AFO [21], as a traditional asynchronous federated optimization algorithm, can totally conquer the synchronization barrier, since there is no need for the server to stall for the straggler devices.

- *Y. Xu, Z. Ma, H. Xu, S. Chen, J. Liu, and Y. Xue are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, China, 230027, and also with Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou, Jiangsu, China, 215123. E-mails: xuyangcs@ustc.edu.cn; zgma@mail.ustc.edu.cn; xuhongli@ustc.edu.cn; chensuo@mail.ustc.edu.cn; jcliu17@ustc.edu.cn; yxxue@ustc.edu.cn.*

Nevertheless, the traditional mechanisms will incur significant communication resource consumption as they require frequent transmission between edge devices and the central server [24]. While for semi-asynchronous FL, the global model is updated by aggregating multiple models rather than a single model. For instance, Liu *et al.* proposed CE-AFL [13], a communication-efficient semi-asynchronous FL mechanism that controls the proportion of participating devices at each epoch, which can reduce the communication resource consumption. However, the semi-asynchronous FL is not always effective to alleviate the negative effect of synchronization barrier, since the selected participating devices may be straggler devices [9]. In addition, the asynchronous FL is known to suffer from the model staleness, which may degrade the performance of global model [25]. Moreover, the negative effect of staleness may be magnified with the presence of *non-i.i.d.* issue [24]–[26].

The above methods focus on the global collaboration of edge devices, *i.e.*, the collaboration is achieved globally on the server by aggregating the local models, without involving the direct local collaboration among edge devices. In synchronous FL, the global collaboration will cause the synchronization barrier for heterogeneous devices and raise conflicts with devices' constrained resources. The existing asynchronous global collaboration causes the performance degradation of the global model in case of the *non-i.i.d.* data. Instead, the local collaboration of edge devices can help each device to learn from the other devices with different data distributions, and the *non-i.i.d.* challenge can be well addressed. To this end, Liu *et al.* proposed an efficient FL framework, called FedMigr [27], which integrates local collaboration into FedAvg by directly migrating the local model of an edge device to another device. However, due to the heterogeneous resource on edge devices, FedMigr may lead to unacceptable waiting time, since it updates the global model in a synchronous manner. Furthermore, the migration strategy in FedMigr is generated by Deep Reinforcement Learning (DRL), which is not feasible in resource-constrained edge computing scenarios, since the training of DRL models requires large amounts of computation resource and training samples [28].

Motivated by this, to accommodate the *heterogeneous devices*, *constrained resources* and *non-i.i.d. data* simultaneously, we design a novel asynchronous FL mechanism (named FedLC) by enabling the local collaboration of edge devices. Concretely, at each epoch, apart from uploading the local model to the server, each device transmits its gradient to the other devices for local collaboration. However, the number of collaborating devices (denoted by k) is an important factor that affects the performance of the proposed mechanism. Specifically, when collaborating with more devices (large k), the model quality will be enhanced, but the resource (computation and communication) overhead is increased. In contrast, the resource overhead can be reduced if small k is adopted, but the performance of the global model may be degraded. Besides, regarding the *non-i.i.d.* data, it is also challenging to determine the set of collaborating devices for each device. Our main contributions are summarized as follows:

- We design a novel asynchronous FL mechanism, named FedLC, for achieving efficient FL in EC. We theoretically analyze the convergence rate of FedLC and obtain the convergence upper bound. The quantitative relationship between convergence bound and parameter k is also present.
- We design an efficient algorithm to determine the set of devices transmitting gradients to each device. We utilize a demand-list to transform pulling gradients from the transmitting devices into pushing gradients to the receiving devices, so as to avoid the block of pull operations.
- We conduct extensive experiments on the real-world datasets. The experimental results demonstrate that the proposed mechanism can accelerate the convergence rate of global model on the *non-i.i.d.* data.

The rest of this paper is organized as follows. Section 2 provides the preliminaries of asynchronous federated learning and the formulation of the problem. The theoretical convergence analysis is presented in Section 3. Section 4 gives the detailed description of the proposed algorithm. In Section 5, the experiments are conducted and the corresponding results are presented. The related works are summarized in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries and Problem Formulation

2.1 Asynchronous Federated Learning

In synchronous FL, the server updates the global model on receiving all the local models, which may lead to the synchronization barrier due to heterogeneous devices and make the computation resources under low utilization. However, in asynchronous FL (AFL), the global model is updated as long as one local model is received, so as to get rid of the negative effect of synchronization barrier. For sake of simplification, we assume that an AFL system consists of N edge devices and a central server in EC. Each device $i \in [N]$ holds a local dataset $\mathcal{D}_i$ that follows a specific distribution P_i, *i.e.*, $\mathcal{D}_i \sim P_i$. Based on the local dataset $\mathcal{D}_i$, device i trains a local model w_i. Concurrently, a globally shared model w is maintained and updated on the central server, so as to minimize the global loss function $F(w)$. Formally, $F(w)$ is expressed as follows [9]

$$F(w) = \sum_{i=1}^{N} \frac{d_i}{D} F_i(w), \tag{1}$$

where $F_i(w)$ denotes the local loss function of device i, d_i represents the size of local dataset $\mathcal{D}_i$, and $D = \sum_{i=1}^{N} d_i$ is the total number of training samples.

For a training task, FL aims to find an optimal model parameter w^* that satisfies

$$w^* \triangleq \arg\min_{w} F(w). \tag{2}$$

Generally, gradient descent based methods, such as [9], [13], [24], are widely used to solve Eq. (2). For each device i, after receiving the global model w^t from the central server at the t-th global epoch, the local model $w_i^{t_i}$ at local epoch t_i is set as w^t. Herein, we define a local epoch as a gradient descent step on the local loss function at each device, while a global

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3307610

TABLE 1: Table of main notations.

Notations	Semantics
N	number of participating devices
D	total number of training samples
η	learning rate
w	global model
w^*	optimal model
t	global epoch index
F	global loss function
w_i	local model of device i
$\mathcal{D}_i$	local dataset of device i
d_i	number of samples in dataset $\mathcal{D}_i$
F_i	local loss function of device i
t_i	local epoch index of device i
∇F_i	gradient of the loss function on device i
$\mathcal{R}_i$	in-neighbor set of device i
$\mathcal{V}_i$	out-neighbor set of device i
B_i	communication capacity of device i
C_i	computation capacity of device i

epoch is referred to as an aggregation step of local models on the server. It is worth noting that the global epoch index t equals the sum of all the local indices in AFL, *i.e.*, $t = \sum_{i=1}^{N} t_i$. Subsequently, the gradient descent method is used to update the local model $w_i^{t_i}$ at local epoch t_i as follows

$$w_i^{t_i+1} = w_i^{t_i} - \eta \nabla F_i(w_i^{t_i}), \tag{3}$$

where η is the learning rate, and $\nabla F_i(w_i^{t_i})$ represents the gradient of local loss function $F_i(\cdot)$. Then the updated local model $w_i^{t_i+1}$ of device i at local epoch t_i+1 is transmitted to the central server for aggregation.

Suppose the central server receives the local model $w_i^{t_i}$ from device i at local epoch t_i. The global model is updated as follows [29]

$$\begin{aligned} w^{t+1} &= w^t - \frac{d_i}{D}(w_i^{t_i} - w_i^{t_i+1}) \\ &= w^t - \frac{d_i}{D}(w_i^{t_i} - (w_i^{t_i} - \eta \nabla F_i(w_i^{t_i}))) \\ &= w^t - \eta \frac{d_i}{D} \nabla F_i(w_i^{t_i}), \end{aligned} \tag{4}$$

where w^t denotes the model at global epoch t. Afterwards, the updated global model w^{t+1} is then sent back to device i to perform local model updating. Some important notations are listed in Table 1.

2.2 Training Process of FedLC

Unlike the traditional AFL methods focusing merely on the global collaboration, FedLC enables the local collaboration among edge devices, so as to accelerate the convergence rate. The framework of FedLC is illustrated in Fig. 1, which mainly consists of four steps, *i.e.*, *model distribution*, *local updating*, *local collaboration* and *global aggregation*.

Model distribution. In this step, the server distributes the latest global model and the corresponding column of demand-list to the device that contributes to the *global aggregation*.

Local updating. After retrieving the global model from the central server, the *local updating* is triggered on each

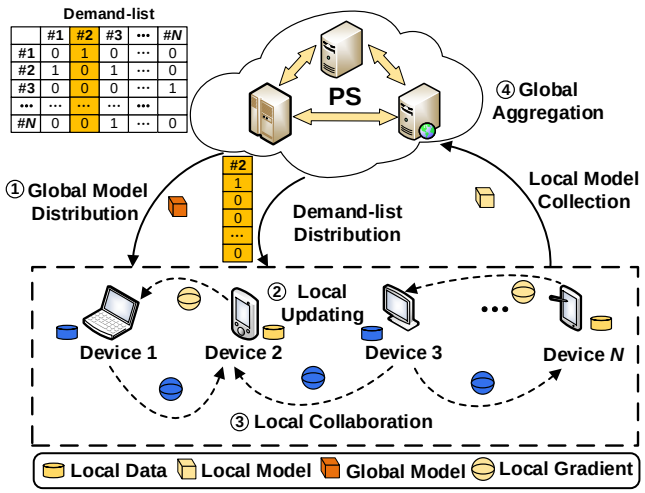


Fig. 1: Illustration of training process of FedLC.

device. On account of the *local collaboration* among devices, apart from the global model, each device may receive multiple local gradients from other devices. For sake of illustration, let $\mathcal{R}_i^{t_i}$ denote the in-neighbor set of device i, *i.e.*, the set of devices transmitting local gradients to device i at local epoch t_i. Similarly, we denote $\mathcal{V}_i^{t_i}$ as the out-neighbor set of device i at local epoch t_i, *i.e.*, the set of devices receiving local gradients from device i at local epoch t_i. Once the global model is received, device i updates the local model $w_i^{t_i}$ at local epoch t_i according to Eq. (3). Unlike the traditional AFL method, FedLC defines the aggregated gradient $\nabla F_i(w_i^{t_i})$ as follows

$$\nabla F_i(w_i^{t_i}) = \frac{1}{|\mathcal{R}_i^{t_i}| + 1} \left(\nabla F_i(w^t) + \sum_{j \in \mathcal{R}_i^{t_i}} \nabla F_j(w^t) \right), \tag{5}$$

where $|\cdot|$ denotes the number of elements in a set. By Eq. (5), the proposed FedLC mechanism is equivalent to the traditional AFL method if $\mathcal{R}_i^{t_i}$ is empty. We note that only one local iteration is illustrated in Eq. (3) for sake of analysis. By changing the definition of $w_i^{t_i+1}$ in Eq. (3), FedLC can still be applied to the cases where multiple local iterations are performed at each epoch.

Local collaboration. Different from the traditional AFL that focuses on global collaboration, we aim to address the statistical heterogeneity by enabling the *local collaboration* among edge devices in FedLC. Concretely, after obtaining the local gradient in *local updating* step, each device directly transmits the local gradient to other devices according to the received demand-list, without relying on the central server. For example, by Fig. 1, device #2 receives the 2nd column of demand-list from the server. According to the demand-list, the second element in the first row of the list is 1, indicating that device #1 requires the gradient of device #2 to correct the optimization direction of its local model. As a consequence, based on the demand-list, device #2 transmits its local gradient to device #1 for collaboration. By *local collaboration* among edge devices, the *non-i.i.d.* data can be well addressed, and the convergence rate of AFL can be improved.

Global aggregation. Similar to the traditional AFL method, the proposed FedLC mechanism requires that the central server updates the global model as long as one local model is collected, so as to remove synchronization barrier caused by system heterogeneity. The *global aggregation* is performed according to Eq. (4). Besides, according to the received parameters, the server will update the corresponding row in demand-list. Then the updated global model and the demand-list are distributed to the edge device that contributes to this *global aggregation*.

2.3 Problem Formulation

To address the statistical heterogeneity challenge, we focus on AFL with local collaboration among edge devices. In EC, the communication and computation resources of edge devices available for model training are always constrained. For device i, we denote B_i and C_i as its communication and computation capacity, respectively. In this paper, we seek to minimize the global loss function $F(w)$ under the resource constraints. The problem can be formulated as follows

$$\min F(w^T)$$
$$\textbf{s.t.}\begin{cases}(|\mathcal{V}_i^{t_i}|+1)*b \le B_i & \forall i,t_i \quad (6a)\\ (|\mathcal{R}_i^{t_i}|+1)*c \le C_i & \forall i,t_i \quad (6b)\\ \left\|w_i^{t_i}-w_j^{t_j}\right\|^2 \ge \delta & \forall i,t_i,j\in\mathcal{V}_i^t,t_j \quad (6c)\end{cases}$$

where T is the total number of global epochs, and $\|\cdot\|$ represents the $\mathcal{L}_2$ norm. Eq. (6a) represents that the resource consumption for transferring the local model and gradient to the central server and other devices should be less than B_i, where b denotes the the bandwidth consumption for transmitting one local model (or gradient). Eq. (6b) denotes that the required computation capacity for aggregating the local gradients is no more than C_i. According to [30], [31], the collaboration among devices with different data distributions helps to improve the performance on *non-i.i.d.* data. Hence, Eq. (6c) essentially promotes the collaboration among devices with different data distributions. The goal is to learn an optimal global model with local device collaboration under the resource constraints, so as to minimize the global loss function $F(w)$.

According to [29], we modify the problem formulation and present a variant of Eq. (6). By [30], the collaboration among devices with different data distributions shows a superior performance over the collaboration among devices with similar distribution. To be intuitive, the benefit of inter-cluster collaboration is to learn from the other devices with different data distributions, which can be expressed by restricting the gap between the local gradient ∇F_i and the global gradient ∇F. In this way, the constraint $\left\|w_i^{t_i}-w_j^{t_j}\right\|^2 \ge \delta$ in Eq. (6) can be replaced by [29]

$$\left\langle \nabla F(w^t), \mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle \ge \epsilon \left\|\nabla F(w^t)\right\|^2, \quad (7)$$

where $\langle\cdot\rangle$ denotes the inner product of two vectors, $\mathbb{E}$ represents the expectation, and ϵ is a predefined factor. Therefore, based on Eq. (7), a variant of Eq. (6) can be formulated as follows

$$\min F(w^T)$$
$$\textbf{s.t.}\begin{cases}(|\mathcal{V}_i^{t_i}|+1)*b \le B_i & \forall i,t_i \quad (8a)\\ (|\mathcal{R}_i^{t_i}|+1)*c \le C_i & \forall i,t_i \quad (8b)\\ \dfrac{\left\langle \nabla F(w^t), \mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle}{\left\|\nabla F(w^t)\right\|^2} \ge \epsilon & \forall i,t_i,t \quad (8c)\end{cases}$$

3 Convergence Analysis

3.1 Assumptions

In order to facilitate the convergence analysis, we make the following assumptions on the loss functions and gradients.

Assumption 1. *F_i is β-smooth with $\beta > 0$, i.e., it always holds $F_i(w_2) - F_i(w_1) \le \langle \nabla F_i(w_1), w_2 - w_1\rangle + \frac{\beta}{2}\left\|w_2 - w_1\right\|^2$ for any two model parameters w_1 and w_2.*

Assumption 2. *F_i is strongly convex with a constant $\mu > 0$, i.e., it always holds $F_i(w_2) - F_i(w_1) \ge \langle \nabla F_i(w_1), w_2 - w_1\rangle + \frac{\mu}{2}\left\|w_2 - w_1\right\|^2$ for any two model parameters w_1 and w_2.*

It is worth noting that Assumption 2 always holds for the models (*e.g.*, linear regression) with convex loss functions. Moreover, though the convergence analysis is derived based on Assumption 2, the experimental results in Section 5 demonstrate that the proposed mechanism can also cope with the non-convex models (*e.g.*, convolutional neural network).

Assumption 3. *∇F_i is V_i-dissimilar, i.e., it always holds $\mathbb{E}(\left\|\nabla F_i(w)\right\|^2) \le V_i^2\left\|\nabla F(w)\right\|^2$ for any model parameter w.*

3.2 Convergence Upper Bound

Based on the assumptions, we first present several lemmas to promote the analysis.

Lemma 1. *Since F_i is β-smooth (by Assumption 1) and μ-strongly convex (by Assumption 2), based on Eq.* (1)*, the global loss function F is also β-smooth and μ-strongly convex.*

For the sake of saving space, we only present the proof for the strong convexity of F, and the proof for smoothness can be derived in the similar way.

Proof. By Eq. (1), $\forall w_1, w_2$ it always holds

$$F(\frac{w_1+w_2}{2}) = \sum_{i=1}^{N}\frac{d_i}{D}F_i(\frac{w_1+w_2}{2}). \quad (9)$$

According to the definition of strong convexity in [32], Eq. (9) can be rewritten as

$$\begin{aligned}F(\frac{w_1+w_2}{2}) &= \sum_{i=1}^{N}\frac{d_i}{D}F_i(\frac{w_1+w_2}{2})\\ &\le \sum_{i=1}^{N}\frac{d_i}{D}\left(\frac{1}{2}F_i(w_1)+\frac{1}{2}F_i(w_2)-\mu\left\|w_1-w_2\right\|^2\right)\\ &\le \frac{1}{2}\sum_{i=1}^{N}\frac{d_i}{D}F_i(w_1)+\frac{1}{2}\sum_{i=1}^{N}\frac{d_i}{D}F_i(w_2)\\ &\quad -\mu\sum_{i=1}^{N}\frac{d_i}{D}\left\|w_1-w_2\right\|^2\\ &= \frac{1}{2}F(w_1)+\frac{1}{2}F(w_2)-\mu\left\|w_1-w_2\right\|^2. \quad (10)\end{aligned}$$

From [32], the global loss function F satisfies the convexity condition and is μ-strongly convex. □

Therefore, according to Assumptions 1-2 and Lemma 1, we can infer that the global loss function is β-smooth and μ-strongly convex.

Lemma 2. *According to the definition of $\nabla F_i(w_i^{t_i})$ in Eq. (5) and Assumption 3, it always holds that*

$$\mathbb{E}\left(\left\|\nabla F_i(w_i^{t_i})\right\|^2\right) \leq \xi_i \left\|\nabla F(w^t)\right\|^2, \tag{11}$$

where $\xi_i = (V_i^2 + \sum_{j\in\mathcal{R}_i^{t_i}} V_j^2)/(|\mathcal{R}_i^{t_i}|+1)$.

Since Lemma 2 can be derived simply by extending the Cauchy-Schwarz inequality [33], we omit the proof for Lemma 2 here.

Lemma 3. *By Lemma 1 and Assumption 2, for any $t \in [T]$, we can obtain*

$$2\mu(F(w^t) - F(w^*)) \leq \left\|\nabla F(w^t)\right\|^2. \tag{12}$$

Lemma 3 is a simple extension of strong convexity, and is supported by some literatures such as [34], [35]. Hence, we only give the rough proof of Lemma 3.

Proof. According to Lemma 1, for any two model parameters w_1 and w_2, we can get

$$F(w_2) - F(w_1) \geq \langle \nabla F(w_1), w_2 - w_1\rangle + \frac{\mu}{2}\left\|w_2 - w_1\right\|^2. \tag{13}$$

By replacing w_1 and w_2 with w^t and w, it is obvious that

$$F(w) - F(w^t) \geq \left\langle \nabla F(w^t), w - w^t\right\rangle + \frac{\mu}{2}\left\|w - w^t\right\|^2, \tag{14}$$

which is equivalent to

$$F(w) \geq F(w^t) + \left\langle \nabla F(w^t), w - w^t\right\rangle + \frac{\mu}{2}\left\|w - w^t\right\|^2. \tag{15}$$

Let $H(w)$ denote the right side of Eq. (15). We have

$$H(w) = F(w^t) + \left\langle \nabla F(w^t), w - w^t\right\rangle + \frac{\mu}{2}\left\|w - w^t\right\|^2. \tag{16}$$

According to (16), we can infer that $H(w)$ is a quadratic function of w. As a consequence, $H(w)$ is minimized when $\nabla H(w) = \nabla F(w^t) + \mu(w - w^t) = \overrightarrow{0}$, where $\overrightarrow{0}$ represents the zero vector. By letting $w = w^t - \frac{\nabla F(w^t)}{\mu}$, we can get the minimum value of $H(w)$ (denoted as $H_{\min}$), *i.e.*,

$$H_{\min} = F(w^t) - \frac{\left\|\nabla F(w^t)\right\|^2}{2\mu}. \tag{17}$$

Based on Eq. (16), it always holds that $F(w^*) \geq H(w^*)$. From Eq. (17), we can further obtain

$$F(w^*) \geq H(w^*) \geq H_{\min} = F(w^t) - \frac{\left\|\nabla F(w^t)\right\|^2}{2\mu}, \tag{18}$$

which is equivalent to

$$2\mu(F(w^t) - F(w^*)) \leq \left\|\nabla F(w^t)\right\|^2. \tag{19}$$

Hence, Lemma 3 is proved. □

Based on the above results, we give the convergence upper bound of the global model as follows.

Theorem 1. *Suppose that Assumptions 1-3 always hold. Let $\xi = \max_{i\in[N]} \xi_i$ and $\eta = \frac{\epsilon D}{d_i \beta \xi}$. After T global epochs, the convergence bound of the global model is given by*

$$\mathbb{E}(F(w^T) - F(w^*)) \leq (1 - \frac{\mu\epsilon^2}{\beta\xi})^T \mathbb{E}(F(w^0) - F(w^*)). \tag{20}$$

According to Theorem 1, we find that the convergence bound is closely related to the *non-i.i.d.* degree via some important factors (*i.e.*, ϵ and ξ). Concretely, when the *non-i.i.d.* degree decreases, the value of ξ becomes lower, and the value of ϵ is increased [29]. Hence, the value of $(1 - \frac{\mu\epsilon^2}{\beta\xi})$ is decreased, leading to a tight convergence bound, and vice versa. Notably, according to [29], when all the local datasets follow the same distribution, the values of ϵ and ξ are all equal to 1, and the convergence bound becomes $(1 - \frac{\mu}{\beta})^T \mathbb{E}(F(w^0) - F(w^*))$.

Proof. According to Lemma 1 and Assumption 1, the β-smoothness condition guarantees

$$F(w_2) - F(w_1) \leq \langle \nabla F(w_1), w_2 - w_1\rangle + \frac{\beta}{2}\left\|w_2 - w_1\right\|^2. \tag{21}$$

Let $w_1 = w^t$ and $w_2 = w^{t+1}$. Eq. (21) can be rewritten as

$$\begin{aligned} &F(w^{t+1}) - F(w^t) \\ &\leq \left\langle \nabla F(w^t), w^{t+1} - w^t\right\rangle + \frac{\beta}{2}\left\|w^{t+1} - w^t\right\|^2. \end{aligned} \tag{22}$$

Then, by Eq. (4), we can further get

$$\begin{aligned} &F(w^{t+1}) - F(w^t) \\ &\leq \left\langle \nabla F(w^t), w^{t+1} - w^t\right\rangle + \frac{\beta}{2}\left\|w^{t+1} - w^t\right\|^2 \\ &= \left\langle \nabla F(w^t), -\eta\frac{d_i}{D}\nabla F_i(w_i^{t_i})\right\rangle + \frac{\beta}{2}\|\eta\frac{d_i}{D}\nabla F_i(w_i^{t_i})\|^2. \end{aligned} \tag{23}$$

By taking the expectation of $F(w^{t+1})$, we can obtain

$$\begin{aligned} &\mathbb{E}(F(w^{t+1})) - F(w^t) \\ &\leq \left\langle \nabla F(w^t), -\eta\frac{d_i}{D}\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle \\ &\quad + \frac{\beta\eta^2 d_i^2}{2D^2}\mathbb{E}\left(\left\|\nabla F_i(w_i^{t_i})\right\|^2\right). \end{aligned} \tag{24}$$

According to Eq. (8c), we can infer that .

$$\left\langle \nabla F(w^t), -\eta\frac{d_i}{D}\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle \leq -\epsilon\eta\frac{d_i}{D}\left\|\nabla F(w^t)\right\|^2. \tag{25}$$

Besides, by Lemma 2, it always holds

$$\frac{\beta\eta^2 d_i^2}{2D^2}\mathbb{E}\left(\left\|\nabla F_i(w_i^{t_i})\right\|^2\right) \leq \frac{\beta\eta^2 d_i^2}{2D^2}\xi_i\left\|\nabla F(w^t)\right\|^2. \tag{26}$$

Hence, by substituting Eqs. (25) and (26) in to Eq. (24), we have

$$\begin{aligned} &\mathbb{E}(F(w^{t+1})) - F(w^t) \\ &\leq \left(-\epsilon\eta\frac{d_i}{D} + \frac{\beta\eta^2 d_i^2}{2D^2}\xi_i\right)\left\|\nabla F(w^t)\right\|^2, \end{aligned} \tag{27}$$

which is equivalent to

$$\mathbb{E}(F(w^{t+1})) - F(w^t) \leq \left(-\epsilon\eta\frac{d_i}{D} + \frac{\beta\eta^2 d_i^2}{2D^2}\xi_i\right)\left\|\nabla F(w^t)\right\|^2 \leq -2\mu\eta\left(\epsilon\frac{d_i}{D} - \frac{\beta\eta d_i^2}{2D^2}\xi_i\right)(F(w^t) - F(w^*)), \quad (28)$$

where the last inequality is derived from Lemma 3. Upon rearrangement, Eq. (28) is equivalent to

$$\mathbb{E}(F(w^{t+1})) - F(w^*) \leq (1 - 2\mu\eta(\epsilon\frac{d_i}{D} - \frac{\beta\eta d_i^2}{2D^2}\xi_i))(F(w^t) - F(w^*)). \quad (29)$$

By taking the expectation of both sides in Eq. (29), we can obtain

$$\mathbb{E}(F(w^{t+1}) - F(w^*)) \leq (1 - 2\mu\eta(\epsilon\frac{d_i}{D} - \frac{\beta\eta d_i^2}{2D^2}\xi_i))\mathbb{E}(F(w^t) - F(w^*)). \quad (30)$$

We can see that $(1 - 2\mu\eta(\epsilon\frac{d_i}{D} - \frac{\beta\eta d_i^2}{2D^2}\xi_i))$ is increased with ξ_i, and $\mathbb{E}(F(w^t) - F(w^*)) \geq 0$. Hence, according to the definition of ξ, Eq. (30) can be modified as

$$\mathbb{E}(F(w^{t+1}) - F(w^*)) \leq (1 - 2\mu\eta(\epsilon\frac{d_i}{D} - \frac{\beta\eta d_i^2}{2D^2}\xi))\mathbb{E}(F(w^t) - F(w^*)). \quad (31)$$

Moreover, for Eq. (31), by letting $t + 1 = T$, we can further get

$$\mathbb{E}(F(w^T) - F(w^*)) \leq (1 - 2\mu\eta(\epsilon\frac{d_i}{D} - \frac{\beta\eta d_i^2}{2D^2}\xi))^T\mathbb{E}(F(w^0) - F(w^*)). \quad (32)$$

We can easily infer that $(1-2\mu\eta(\epsilon\frac{d_i}{D}-\frac{\beta\eta d_i^2}{2D^2}\xi))^T$ is minimized when $\eta = \frac{\epsilon D}{d_i\beta\xi}$. Hence, Theorem 1 is proved by setting $\eta = \frac{\epsilon D}{d_i\beta\xi}$ in Eq. (32). □

4 Algorithm Design

For a certain loss function $F(w)$, its minimum value $F(w^*)$ can be regarded as a constant. In this way, optimizing $F(w^T)$ in Eq. (8) can be shifted to minimizing the upper bound of $\mathbb{E}(F(w^T) - F(w^*))$, which is related to some parameters (*e.g.*, ϵ, ξ and β) by Eq. (20). As a consequence, we first present a method to estimate these parameters. Then we present the proposed algorithm for the server and each device to solve the problem. Finally, we extend the proposed algorithm for adapting to the real EC environment.

4.1 Approximate Estimation of Parameters ϵ, ξ and β

According to Eq. (20), the upper bound of $\mathbb{E}(F(w^T) - F(w^*))$ is related to the parameters μ, ϵ, β and ξ. In this paper, we consider a practical scenario where these parameters are unknown in advance and may vary as training progresses. For simplification, we ignore $\frac{\mu}{\beta}$ and minimize $(1 - \frac{\mu\epsilon^2}{\beta\xi})$ by maximizing $\frac{\epsilon^2}{\xi}$, since the value of $\frac{\mu}{\beta}$ is not affected by the local collaboration of edge devices [11].

According to Eq. (8c), we can obtain

$$\left\langle\nabla F(w^t), \mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle \geq \epsilon\left\|\nabla F(w^t)\right\|^2. \quad (33)$$

By taking square of both sides, Eq. (33) can be rewritten as

$$\left\langle\nabla F(w^t), \mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle^2 \geq \epsilon^2\left\|\nabla F(w^t)\right\|^4. \quad (34)$$

Moreover, it is obvious that

$$\begin{aligned}&\left\langle\nabla F(w^t), \mathbb{E}(\nabla F_i(w_i^{t_i}))\right\rangle \\ &= \left\|\nabla F(w^t)\right\|\left\|\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\|\cos(\nabla F(w^t), \mathbb{E}(\nabla F_i(w_i^{t_i}))) \\ &\leq \left\|\nabla F(w^t)\right\|\left\|\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\|. \end{aligned} \quad (35)$$

By combining Eqs. (34) and (35), we can obtain

$$\left\|\nabla F(w^t)\right\|^2\left\|\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\|^2 \geq \epsilon^2\left\|\nabla F(w^t)\right\|^4, \quad (36)$$

which is equivalent to

$$\left\|\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\|^2 \geq \epsilon^2\left\|\nabla F(w^t)\right\|^2. \quad (37)$$

Hence, the factor ϵ satisfies

$$\left\|\mathbb{E}(\nabla F_i(w_i^{t_i}))\right\| \geq \epsilon\left\|\nabla F(w^t)\right\|. \quad (38)$$

For ease of notations, we let l_i denote the norm of gradient for device i, *i.e.*,

$$l_i = \left\|\nabla F_i(w)\right\|. \quad (39)$$

By Eqs. (7) and (38), ϵ can be approximately formulated as

$$\epsilon \triangleq \frac{N \cdot (l_i + \sum_{j\in\mathcal{R}_i^{t_i}} l_j)}{k \cdot (l_1 + l_2 + \cdots + l_N)}, \quad (40)$$

where N denotes the number of edge devices, $\mathcal{R}_i^{t_i}$ represents the in-neighbor set of device i at local epoch t_i, and k is defined as $k = |\mathcal{R}_i^{t_i}| + 1$.

In terms of ξ, according to Assumption 3, V_i^2 can be approximately formulated as

$$V_i^2 \triangleq \frac{N \cdot l_i^2}{l_1^2 + l_2^2 + \cdots + l_N^2}. \quad (41)$$

Then, by the definition in Lemma 3, we can estimate ξ as

$$\xi \triangleq \frac{N \cdot (l_i^2 + \sum_{j\in\mathcal{R}_i^{t_i}} l_j^2)}{k \cdot (l_1^2 + l_2^2 + \cdots + l_N^2)}. \quad (42)$$

Regarding the restriction on learning rate in Theorem 1, apart from the parameters ϵ and ξ, β is also necessary to calculate η. As a consequence, for estimating η in real time, we should dynamically calculate β as follows [11],

$$\beta = \frac{1}{N}\sum_{i\in[N]}\beta_i = \frac{1}{N}\sum_{i\in[N]}\frac{|F_i(w_i^{t_i}) - F_i(w_i^{t_i-1})|}{\left\|w_i^{t_i} - w_i^{t_i-1}\right\|}. \quad (43)$$

4.2 Description of FedLC

It is worth noting that the convergence upper bound is related to the devices from in-neighbor set $\mathcal{R}$. However, pulling gradients from other devices may lead to block and incur non-negligible waiting time, since all the devices perform the local training in an asynchronous manner. In this way, we propose to maintain a demand-list $\mathcal{M}$ on the server. By Fig. 1, $\mathcal{M}$ is a matrix with size of $N \times N$, where device $i \in [N]$ demands the gradient of device $j \in [N]$ when $\mathcal{M}_{i,j} = 1$. Hence, the i-th row of $\mathcal{M}$ denotes the in-neighbor set of device i, and the j-th column of $\mathcal{M}$ represents the out-neighbor set of device j. By distributing the j-th column of

Algorithm 1 Procedure at the server

Require: Total number of epochs T
Ensure: w^T

1: Initialize the global model w^0 as a random vector;
2: Initialize the global epoch t as $t \leftarrow 0$;
3: Initialize the demand-list $\mathcal{M}$ as a matrix with full zero;
4: Initialize β_i and l_i for all device $i \in [N]$;
5: Initialize the learning rate η and D;
6: **for** $i = 1$ to N **do**
7: Send w^0, η and $\mathcal{M}_{:,i}$ to device i;
8: **end for**
9: **while** $t < T$ **do**
10: Receive $w_i^{t_i+1}$ and d_i from device i;
11: Update the global model w^{t+1} according to Eq. (4);
12: **if** $t_i \geq 1$ **then**
13: Receive β_i and l_i from device i and replace the old one;
14: **if** All the β_i and l_i ($i \in [N]$) are received **then**
15: Set the row related to device i in demand-list as $\mathcal{M}_{i,:} = \text{LocalCol}(\mathcal{M})$;
16: Estimate β according to Eq. (43);
17: Update the learning rate η as $\eta = \frac{\epsilon D}{d_i \beta \xi}$;
18: **end if**
19: **end if**
20: Send w^{t+1}, η and $\mathcal{M}_{:,i}$ to device i;
21: Start the next epoch and set $t \leftarrow t + 1$;
22: **end while**
23: Send $STOP$ flag and the global model W^T to all devices;

24: **function** LocalCol($\mathcal{M}$)
25: Initialize $S \leftarrow [0, 0, \cdots, 0]$, $k \leftarrow 1$;
26: Calculate $\overline{\epsilon} \leftarrow \frac{N \cdot l_i}{l_1 + l_2 + \cdots + l_N}$ and $\overline{\xi} \leftarrow \frac{N \cdot l_i^2}{l_1^2 + l_2^2 + \cdots + l_N^2}$;
27: **for** $j = 1$ to N **do**
28: **if** $j \neq i$, $k < \frac{C_i}{c}$ and $\sum_{h \in [N]} \mathcal{M}_{h,j} < \frac{B_j}{b}$ **then**
29: Calculate $\hat{\epsilon}$ and $\hat{\xi}$ according to Eqs. (40) and (42);
30: **if** $\frac{\hat{\epsilon}^2}{\hat{\xi}} > \frac{\overline{\epsilon}^2}{\overline{\xi}}$ **then**
31: Select device j and set $S_j \leftarrow 1$;
32: Set $k \leftarrow k + 1$;
33: **end if**
34: **end if**
35: **end for**
36: **return** S
37: **end function**

Algorithm 2 Procedure at device i

Ensure: w^T

1: Initialize the resource budgets B_i, b, C_i and c;
2: Initialize the local epoch t_i as $t_i \leftarrow 0$;
3: Initialize the receive buffer R_i as an empty set;
4: Initialize d_i;
5: **while** The *STOP* flag is not received **do**
6: Receive the gradients from other devices and store them in $\mathcal{R}_i$;
7: Receive w^t, η, and $\mathcal{M}_{:,i}$ from the server;
8: Calculate the local gradient $\nabla F_i(w^t)$;
9: Compute the aggregated gradient $\nabla F_i(w_i^{t_i})$ according to Eq. (5) and clear the receive buffer R_i;
10: Update the local model $w_i^{t_i+1}$ according to Eq. (3);
11: **for** $j = 1$ to N **do**
12: **if** $\mathcal{M}_{j,i} = 1$ **then**
13: Send $\nabla F_i(w^t)$ to the device j;
14: **end if**
15: **end for**
16: Send $w_i^{t_i+1}$ and d_i to the server;
17: **if** $t_i \geq 1$ **then**
18: Calculate the norm $l_i \leftarrow \left\|\nabla F_i(w^t)\right\|$;
19: Estimate β_i as $\beta_i \leftarrow \frac{|F_i(w_i^{t_i}) - F_i(w_i^{t_i-1})|}{\left\|w_i^{t_i} - w_i^{t_i-1}\right\|}$;
20: Send β_i and l_i to the server;
21: **end if**
22: Set $t_i \leftarrow t_i + 1$;
23: **end while**
24: Receive w^T from the server;

$\mathcal{M}$ to device j, we can transform the in-neighbor set $\mathcal{R}$ into the out-neighbor set $\mathcal{V}$, and the block can be eliminated.

On the basis of the above results, we present the algorithm for the server and each device in Algorithms 1 and 2, respectively. First of all, we initialize some critical parameters for the server (Lines 1-5 of Algorithm 1) and each device (Lines 1-4 of Algorithm 2). On the server side, the global model w^0 is initialized as a random vector. The demand-list $\mathcal{M}$ is created as a matrix with full zero, indicating that all the devices train the model locally and collaborate with each other via the global aggregation at the start. In addition, the other parameters, such as the learning rate η and total number of samples D, are also initialized on the server. On the device side, Algorithm 2 begins with the initialization of the resource budgets (*i.e.*, B_i, b, C_i and c). Besides, each device locally maintains a receive buffer that stores the gradients received from the other devices. The receive buffer R_i is initialized as an empty set on device i. The other important parameters (*e.g.*, local epoch index t_i and number of samples d_i on device i) are specified in advance on each device.

After the initialization, the server sends the initialized global model w^0, learning rate η and $\mathcal{M}_{:,i}$ to device i and starts the model training (Lines 6-8 in Algorithm 1), where $\mathcal{M}_{:,i}$ represents the i-th column of $\mathcal{M}$ (*i.e.*, the devices that request the gradient of device i). To go along with this, each device i receives the gradients from other devices and stores them in the receive buffer R_i (Line 6 of Algorithm 2). Simultaneously, device i receives w^t, η and $\mathcal{M}_{:,i}$ from the server (Line 7 of Algorithm 2). Afterwards, as shown by Line 8 in Algorithm 2, the local gradient $\nabla F_i(w^t)$ is computed by device i. On the basis of $\nabla F_i(w^t)$ and the received gradients in receive buffer R_i, device i aggregates the local gradients according to Eq. (5). The receive buffer R_i will be cleared after the aggregation (Line 9 of Algorithm 2). The aggregated gradient $\nabla F_i(w_i^{t_i})$ and the learning rate η are then used to update the local model $w_i^{t_i+1}$ according to Eq. (3) (Line 10 of Algorithm 2). The local collaboration step is performed on device i by sending its local gradient $\nabla F_i(w^t)$ to device j if $\mathcal{M}_{j,i} = 1$ (Lines 11-15 of Algorithm 2). At the same time, the updated local model w^{t_i+1} is transmitted to

the server. Besides, when the local epoch index t_i is larger than 0, device i will estimate the local gradient norm l_i and the parameter β_i. The estimated l_i and β_i are sent to the server (Lines 17-21 of Algorithm 2), so as to calculate the learning rate and update the demand-list. Then device i set $t_i = t_i + 1$ and starts the next epoch, as shown by Line 22 of Algorithm 2.

On the server side, the global aggregation step is triggered by receiving the local model $w_i^{t_i+1}$ from device i (Line 10 of Algorithm 1). As depicted by Line 11 of Algorithm 1, the server updates the global model w^{t+1} using the received local model $w_i^{t_i+1}$ according to Eq. (4). Similar to Algorithm 2 on each device, the server will receive the estimated β_i and l_i if the received local epoch index t_i is larger than 0 (Line 13 of Algorithm 1). The received β_i and l_i will replace the previously stored values. Furthermore, when all the β_i and l_i ($i \in [N]$) are received, the server updates the demand-list $\mathcal{M}_{i,:}$ associated with device i (*i.e.*, the i-th row of $\mathcal{M}$) using LocalCol (Line 15 of Algorithm 1). The demand-list for device i (denoted by S) is first initialized as a vector filling with zero. Besides, the number of collaborating devices k is specified as $k \leftarrow 1$ in advance. Then we calculate $\overline{\epsilon} \leftarrow \frac{N \cdot l_i}{l_1 + l_2 + \cdots + l_N}$ and $\overline{\xi} \leftarrow \frac{N \cdot l_i^2}{l_1^2 + l_2^2 + \cdots + l_N^2}$ using the received l_i (Line 26 of Algorithm 1). For each device $j \neq i$, if the computation constraint of device i and the bandwidth of target device j are satisfied, we estimate $\hat{\epsilon}$ and $\hat{\xi}$ according to Eqs. (40) and (42) (Line 29 of Algorithm 1). We note that the gradient of device j is helpful to device i if $\frac{\hat{\epsilon}^2}{\hat{\xi}} > \frac{\overline{\epsilon}^2}{\overline{\xi}}$, since it produces lower convergence upper bound and accelerates the convergence of the model. Hence, we claim that device i requires the gradient of device j and set $S_j = 1$ when $\frac{\hat{\epsilon}^2}{\hat{\xi}} > \frac{\overline{\epsilon}^2}{\overline{\xi}}$ (Line 31 of Algorithm 1). When all devices are checked, LocalCol generates the demand-list for device i and returns it to the server (Line 36 of Algorithm 1). Then the server replaces the i-th row of $\mathcal{M}$ with S. Subsequently, the server estimates β according to Eq. (43) and updates the learning rate as $\eta = \frac{\epsilon D}{d_i \beta \xi}$, where ϵ and ξ can be calculated using the updated demand-list based on Eqs. (40) and (42). At the end of global epoch t, the server sends the updated global model w^{t+1}, learning rate η and demand-list $\mathcal{M}_{:,i}$ (i-th column of $\mathcal{M}$) to device i (Line 20 of Algorithm 1). Then the server starts the next epoch. The above process is conducted iteratively until all the required T epochs are completed. Then, the server sends the *STOP* flag and the resulting global model w^T to all devices, and the proposed algorithm terminates.

4.3 Staleness-Compensated FedLC

In EC, the computation and bandwidth capacities of edge devices are generally heterogeneous, and thus the arrival frequency of different devices may also be diverse. In other words, the version of some local models may be stale in AFL. The convergence rate of the global model is degraded when using the stale local models for global aggregation [9]. Hence, inspired by [24], we propose staleness-compensated FedLC (SC-FedLC), which assigns different local learning rates for each device. The local learning rate η_i for each device i is determined by the relative frequency (denoted as r_i) it participates in the global aggregation. The participation frequency r_i for device i is defined as the local epoch index t_i divided by the global epoch index t, *i.e.*, $r_i = \frac{t_i}{t}$. Recall that the global epoch index t is defined as $t = \sum_{i \in [N]} t_i$. As a consequence, it is obvious that $\sum_{i \in [N]} r_i = 1$. Afterwards, based on the participation frequency r_i, the local learning rate η_i for device i can be calculated as

$$\eta_i = \frac{\eta}{N \cdot r_i}. \tag{44}$$

We note that the weighted average learning rate of all devices is equal to η: $\sum_{i \in [N]} r_i \eta_i = N \cdot \frac{\eta}{N} = \eta$. We can see that the learning rate of device i is inversely proportional to its participation frequency, which has been verified to be effective by [24].

4.4 Discussion

In EC, the edge devices participating in FL may be various kinds of mobile devices. In addition, edge devices always connect to base station (BS) via unstable wireless links. Hence, considering the device mobility in dynamic EC environment, the communication timeout issue may occur due to the BS handover or unstable network condition. We note that the communication timeout may occur in both global model distribution and local model collection. In both two cases, edge devices cannot retrieve the global model from the server. In addition, by Line 28 in Algorithm 1, when the network condition becomes poor, most of the devices are prohibited to collaborate due to the increased communication time. The lack of collaboration may cause the deviation of global model in AFL, and the performance of global model is degraded. To this end, we propose to refer to the timeout retransmission mechanism in TCP protocol, and introduce a similar method that retransmits the local models if no reply is received from the server within the retransmission timeout (RTO), so as to handle the poor network conditions. Furthermore, the network conditions always vary with time in EC. As as consequence, the value of RTO (denoted as $\mathcal{T}_{RTO}$) should be adjusted based on the real-time network condition. According to [36], $\mathcal{T}_{RTO}$ can be adjusted as follows:

$$\mathcal{T}_{RTO} = \widehat{\mathcal{T}}_{RTT} + \varphi * \sigma(\mathcal{T}_{RTT}), \tag{45}$$

where φ is a factor indicating the network condition and is set as 4 in general [36], $\widehat{\mathcal{T}}_{RTT}$ denotes the smoothed round-trip time and can be obtained by incorporating the history round-trip time $\mathcal{T}_{RTT}$, and $\sigma(\mathcal{T}_{RTT})$ represents the round-trip time variation.

Additionally, we aim to show the communication efficiency of FedLC by comparing its communication overhead with that of FedAvg. Concretely, for FedAvg, the server collects all the local models and distributes the global model to all devices at each epoch, leading to the communication overhead of $\mathcal{O}(N \cdot b)$, where b denotes the communication overhead of transmitting one model. While for FedLC, the communication overhead is $\mathcal{O}(k \cdot b)$, where k denotes the number of collaborating devices. Furthermore, when the local model is independently dropped with a probability of q for each device, the communication overhead of FedAvg becomes $\mathcal{O}(N \cdot b \cdot (1 + \frac{q}{1-q^2}))$[1]. Similarly, the pro-

1. $\lim_{n \to \infty} q + 2 \cdot q^2 + \cdots + n \cdot q^n = \frac{q}{1-q^2}$

posed mechanism incurs the communication overhead of $\mathcal{O}(k \cdot b \cdot (1+\frac{q}{1-q^2}))$. We note that the number of collaborating devices k is usually smaller than the total number of devices N. Hence, at each epoch, the communication overhead of FedLC is lower than that of FedAvg.

Furthermore, we also compare the time complexity of the proposed mechanism and typical FedAvg at each epoch. For sake of analysis, let $\mathcal{T}_f$, $\mathcal{T}_b$ and $\mathcal{T}_p$ denote the time complexity of forward pass, backward pass and parameter update, respectively. For typical FedAvg, the server updates the global model after collecting all the local models at each epoch, and the total time complexity is expressed as $\mathcal{O}(N \cdot (\mathcal{T}_f + \mathcal{T}_b + \mathcal{T}_p) + N \cdot \mathcal{T}_p)$ [9]. Concretely, $N \cdot (\mathcal{T}_f + \mathcal{T}_b + \mathcal{T}_p)$ is caused by the local update of N devices, and $N \cdot \mathcal{T}_p$ is derived from the global aggregation on the server. We can observe that the time complexity of FedAvg increases with N (*i.e.*, the number of devices). In comparison, FedLC updates the global model at each epoch as soon as one local model from an arbitrary device is received, and the total time complexity becomes $\mathcal{O}((\mathcal{T}_f + \mathcal{T}_b + 3 \cdot \mathcal{T}_p) + \mathcal{T}_p)$. Concretely, $\mathcal{T}_f + \mathcal{T}_b + 3 \cdot \mathcal{T}_p$ denotes the time complexity of local update and estimation of some parameters (*i.e.*, l_i and β_i), and $\mathcal{T}_p$ is caused by aggregating one local model. When the number of participating devices is increased, the time complexity of FedLC remains stable, which shows the efficiency of the proposed mechanism.

As for the data privacy, we note that each edge device transmits the local model (or gradient) rather than the raw data to the server (or the other edge devices), protecting private data from being eavesdropped by hidden adversaries [4], [37]. Besides, some privacy-preserving techniques (e.g., differential privacy [38]) can be combined to further enhance the privacy preserving of transmitted models or gradients.

5 Experimentation and Evaluation

5.1 Experimental Setup

Concretely, the heterogeneous resource on edge devices may lead to unacceptable waiting time for synchronous FL and model staleness for asynchronous FL, and the completion time for model training is increased. In addition, the *non-i.i.d.* challenge will cause the deviation of the global model, and the training performance is degraded. In this way, we compare the performance of FedLC and baselines on different *non-i.i.d.* degrees and various computing capacity levels.

We build an FL experimental environment on an AMAX deep learning workstation (CPU: Intel 671 (R) E5-2620v4, GPU: NVIDIA TITAN RTX), upon which the baselines are implemented with PyTorch [39]. In the simulation environment, we create 21 virtual machines (VMs), one of which is specified as the edge server and the remaining 20 VMs act as the edge devices. To represent the computation heterogeneity of edge devices, we separately measure and record the computation time of four different types of commercial devices (*i.e.*, Raspberry Pi, NVIDIA Jetson TX2, XAVIER NX, and AGX XAVIER) in advance. Subsequently, we set the computation capacity of each VM as the average of one type of the commercial devices. For simulating the communication heterogeneity, we measure the communication bandwidth (denoted as B) between the server and edge device. Then, by taking B as the standard, the communication bandwidth between devices i and j is set as $\gamma_{i,j} \cdot B$, where $\gamma_{i,j}$ is a factor ranging randomly from 0.1 to 10. In practice, the devices i and j with $\gamma_{i,j} > 1$ are encouraged to collaborate for reducing the communication cost, which intuitively reflects the network connectivity of devices.

TABLE 2: Table of main performance modes (GHz) for Jetson devices.

Device	Mode	Denver2	ARMA57	ARMv8.2	GPU
TX2	1	2.0×2	2.0×4	-	1.3
	2	1.4×2	1.4×4	-	1.12
	3	-	1.2×4	-	0.85
NX	1	-	-	1.4×6	1.1
	2	-	-	1.2×4	0.8
AGX	1	-	-	1.2×4	0.67
	2	-	-	1.2×8	0.9

We also perform the testbed experiments on an AMAX deep learning workstation and a varying number (*i.e.*, 15, 20, 30, 40 and 50) of NVIDIA Jetson TX2, XAVIER NX, and AGX XAVIER developer Kits, where the workstation acts as the server and the Jetson Kits serve as the edge devices. Each TX2 device has one GPU and one CPU cluster (*i.e.*, a 2-core Denver2 and a 4-core ARM CortexA57). Each NX device is equipped with a 6-core NVIDIA Carmel ARMv8.2 CPU and a 384-core NVIDIA Volta GPU. The AGX device carries a 512-core NVIDIA Volta GPU and an 8-core NVIDIA Carmel ARMv8.2 CPU. The experimental network is established via a commercial router, where the server is connected by the Ethernet and Jetson devices are accessed via wireless link. We empower the Jetson devices with different performance modes[2] to represent the computation heterogeneity, as shown in Table 2. Concretely, the computation heterogeneity can be divided into 3 levels: *level-1*, *level-2*, and *level-3*, and the maximum performance gap under 3 levels is measured as 5×, 10×, and 30×, respectively.

Besides, we simulate the communication heterogeneity of edge devices by varying their communication distance to the router. Specifically, we put the Jetson devices at three different locations that are 2m, 10m and 30m away from the router, and their average communication bandwidth is around 10MB/s, 3MB/s and 1MB/s, respectively. It is worth noting that the simulation and testbed experiments are mostly performed on 20 edge devices (*i.e.*, VMs in simulation and Jetson devices in tested), unless otherwise stated.

5.1.1 Datasets and Models

We conduct extensive experiments on three real-world datasets: (i) CIFAR-10[3], (ii) EMNIST [40], and (iii) ImageNet [41]. Specifically, CIFAR-10 contains a training set with 50,000 samples and a test set with 10,000 samples. Each sample in CIFAR-10 is a 32×32×3 RGB image from 10 categories. EMNIST is an extended version of the popular MNIST dataset, which contains 731,668 training samples

2. The NVIDIA Jetson developer Kits can be powered by different performance modes. The CPU and GPU frequencies can be various under different modes. More details are available at https://jetsonhacks.com/2017/03/25/nvpmodel-nvidia-jetson-tx2-development-kit/

3. http://www.cs.toronto.edu/kriz/cifar.html

TABLE 3: Table of main parameters used in experiments.

	CIFAR-10	EMNIST	IMAGE-100
models	LeNet-5	AlexNet	VGG-16
batch size	64	64	32
learning rate	0.003	0.0001	0.001
training epochs	300	500	500

and 82,587 testing samples from 62 categories (10 digits, 52 characters with lowercase and uppercase). ImageNet is a widely used dataset for visual recognition which consists of 1,281,167 training images, 50,000 validation images and 100,000 test images from 1,000 categories, and each sample in ImageNet is a 224×224×3 image. Considering the constrained resource on edge devices, we create IMAGE-100, a subset of ImageNet that consists of 100 out of 1,000 categories, and each image is downsized with the shape of 144×144×3.

In both simulation and testbed experiments, all the edge devices are divided into 5 clusters. For simulating the *non-i.i.d.* data, we split each dataset into 5 partitions, which are allocated to 5 clusters. Concretely, each cluster generates the partition by selecting 20% of all the labels as its main categories, and the main categories of all clusters are disjoint. In addition, we introduce a factor p that represents the percentage of samples from main categories, so as to define different *non-i.i.d.* degrees. For instance, when $p = 0.8$, 80% of the samples on each cluster belong to the main categories, while the remaining samples belong to the other categories. In this paper, four *non-i.i.d.* degrees are adopted: (i) *degree-1*, (ii) *degree-2*, (iii) *degree-3*, and (iv) *degree-4*, and p is set as 1.0, 0.8, 0.4 and 0.2, respectively. We note that *degree-4* (*i.e.*, $p = 0.2$) is a special case, where the datasets of all the clusters follow the *i.i.d.* distribution. Then, these partitions are further divided to serve as the local datasets of edge devices in each cluster.

Corresponding to the above three datasets, three models with different structures are adopted for performance evaluation: (i) LeNet-5 [42] on CIFAR-10, (ii) AlexNet [43] on EMNIST, and (iii) VGG-16 [44] on IMAGE-100. Firstly, for the simple task (*i.e.*, CIFAR-10), the classical model LeNet-5 which has two 3×3 convolutional layers, two dense layers and a softmax output layer is implemented. Secondly, for classifying the images in EMNIST which contains more samples and categories, the 8-layer AlexNet model with more complex structures is adopted. Thirdly, a famous model VGG-16 that consists of 13 convolutional layers with the kernel of 3×3, two dense layers and a softmax output layer is utilized for image classification of IMAGE-100. To cope with the downsized images and the constrained resource, the VGG-16 model is also reshaped to an appropriate size.

5.1.2 Parameter Settings

In both simulation and testbed experiments, the batch size and learning rate are set as 64 and 0.003, 64 and 0.0001, as well as 32 and 0.001 for CIFAR-10, EMNIST and IMAGE-100, respectively. In addition, for three datasets, the total number of training epochs is specified as 300, 500 and 500 [45]. We note that the required time for completing one epoch varies significantly for synchronous FL and asynchronous FL. To this end, we compare the test accuracy given the training time and the completion time to achieve the target accuracy, as depicted in Section 5.1.4.

5.1.3 Baselines

In order to evaluate the effectiveness of the proposed FedLC and SC-FedLC, we introduce the other three methods as baselines: (i) FedAvg [4], (ii) CE-AFL [13], and (iii) AFO [21]. Concretely, the first baseline FedAvg is the typical parameter server based synchronous FL mechanism, where the server updates the global model by averaging the local models from all edge devices. The second baseline CE-AFL is a semi-asynchronous FL mechanism, where the server updates the global model after receiving the local models from $\alpha \cdot N$ edge devices. According to the results in [13], α is set as 0.5 for reducing the training time, *i.e.*, half of the edge devices participate in the model update on the server at each epoch. The third baseline AFO is an asynchronous FL algorithm with provable convergence rate, in which the server updates the global model on receiving only one local model from an arbitrary edge device. We implement the adaptive weight to handle the model staleness for AFO, *i.e.* $w^{t+1} = (1 - x_t)w^t + x_t w_i^{t_i}$, where $x_t \in [0, 1]$ represents the aggregation weight at global epoch t. Moreover, the weight x_t is adjusted as $x_t = x \times \frac{1}{s+1}$, where x is set as 0.6 and s denotes the model staleness (*i.e.*, the difference of global epoch and local epoch).

In order to cope with the poor network conditions, we introduce an extra baseline, named FedLC-T, which combines FedLC with timeout retransmission mechanism. To simulate the poor network conditions, during the communication process at each epoch, the local models on edge devices are assumed to be dropped with a probability of 0.1. For performance evaluation, at *level-1* in testbed experiments, we compare the test accuracy of FedLC and FedLC-T under the simulated network conditions.

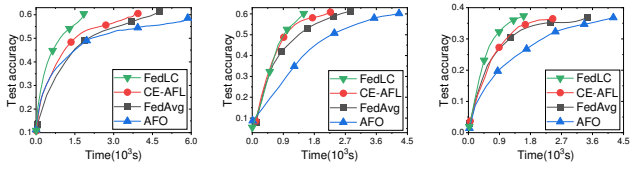
5.1.4 Metrics

In this paper, the following metrics are utilized to evaluate the performance of the above methods: (i) test accuracy, (ii) completion time, and (iii) traffic consumption. The test accuracy is one of the most common performance metrics, which measures the proportion between the number of correct data after model inference and the total number of data. Specifically, during the training process, at each epoch, we evaluate the test accuracy of three models trained with different methods on the test datasets. Training time is an important metric for evaluating the training speed of a method on a training task. Hence, we record the completion time of each method when it achieves the target accuracy for performance evaluation. In addition, in order to illustrate that the proposed mechanisms will not incur extra communication overhead, we also record the accumulated network traffic consumption for different methods to achieve the same test accuracy.

5.2 Results of Simulation Experiments

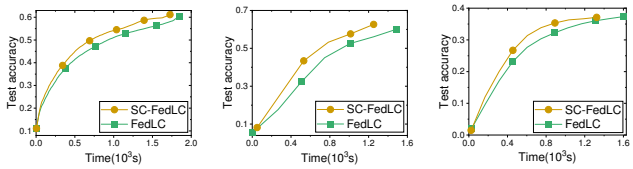
5.2.1 Convergence Performance

We conduct the first set of simulation experiments to compare the convergence performance of FedLC with those



(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 2: Test accuracy curves of the models trained with different methods in simulation.

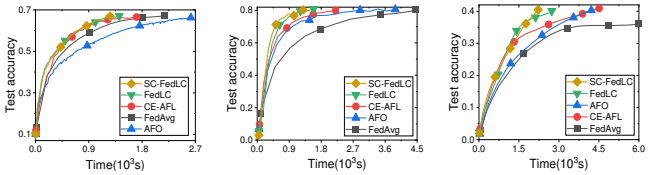


(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 3: Test accuracy curves of the models trained with FedLC and SC-FedLC in simulation.

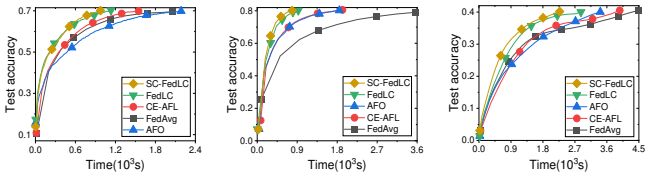
of the existing methods (*i.e.*, FedAvg, CE-AFL, and AFO). Concretely, the test accuracies of the models trained with different methods under *degree-1* (*i.e.*, the *non-i.i.d.* factor $p = 1$) are shown in Fig. 2, where the horizontal axis represents the training time. By Fig. 2, AFO (*i.e.*, the asynchronous baseline) always performs the slowest convergence performance among four methods on three datasets. For example, in Fig. 2(c), when the training time equals 1,000s, the test accuracy of AFO is 0.21, which is far from the convergence. In comparison, the test accuracies of FedAvg, CE-AFL and FedLC are 0.28, 0.29 and 0.34, which are almost converged. Among three existing baselines, CE-AFL always shows a superior performance to FedAvg (the synchronous baseline) and AFO (the asynchronous baseline), since the partial device participation in CE-AFL is more resistant to the device heterogeneity (compared with FedAvg) and can catch up with the track of whole data distribution (compared with AFO). Moreover, though the test accuracies increase for both FedLC and three baselines as the training progresses, the proposed FedLC mechanism achieves a faster convergence performance. To obtain the test accuracies of 0.6, 0.6 and 0.4 on three datasets, the completion time of FedLC is 1,843s, 1,491s, and 1,603s, while the minimum completion time among three baselines is 3,654s, 2,266s, and 2,445s. Generally, to achieve the convergence on three datasets, FedLC reduces the completion time by at least 49.6%, 34.2% and 34.4% compared with the existing methods, which demonstrates the effectiveness of the proposed mechanism.

From the above results, we can see that FedLC is able to reduce the completion time for model convergence. That is because the asynchronous mechanism can completely remove the waiting time between devices, and the time for finishing one epoch is decreased. In addition, the local collaboration between devices can help to track the global data distribution, and the total number of epochs for convergence is reduced.



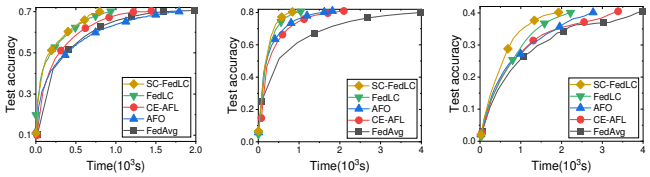
(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 4: Test accuracy curves of the models trained with different methods on three datasets under *degree-2*.



(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 5: Test accuracy curves of the models trained with different methods on three datasets under *degree-3*.



(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 6: Test accuracy curves of the models trained with different methods on three datasets under *degree-4*.

5.2.2 *Effect of Staleness Compensation*

The second set of simulation experiments is performed on three datasets under *degree-1* to evaluate the effect of staleness compensation. The test accuracies of the models trained with FedLC and SC-FedLC are illustrated in Fig. 3, where the horizontal axis denotes the training time. By Fig. 3, we can see that the test accuracies of both FedLC and SC-FedLC increase with the training time. However, SC-FedLC achieves a faster convergence rate than the FedLC mechanism. For example, in Fig. 3(c), when the training time equals 891s, SC-FedLC obtains the test accuracy of 0.35 on IMAGE-100, which is close to the convergence. In comparison, the test accuracy of FedLC is 0.32 on IMAGE-100. Generally, in Fig. 3, to obtain the test accuracies of 0.61, 0.62 and 0.36 on three datasets, the completion time of SC-FedLC is 1,709s, 1,252s and 1,323s, while FedLC consumes the training time of 1,843s, 1,491s and 1,603s. In other words, compared with FedLC, SC-FedLC reduces the completion time by 7%, 16% and 17% on CIFAR-10, EMNIST and IMAGE-100, respectively.

From the above results, SC-FedLC outperforms FedLC and reduces the completion time for convergence by using the staleness compensation mechanism. That is because SC-FedLC assigns larger learning rates for the stale local models, and the negative effect of staleness on convergence can be alleviated.

5.2.3 Impact of Non-i.i.d. Degrees

The third set of experiments is conducted to testify the impact of different *non-i.i.d.* degrees. Specifically, we compare the test accuracies of different methods on three datasets in simulation, under *degrees 2-4*. The test accuracy curves of the models trained with different methods are depicted in Figs. 4 (*degree-2*), 5 (*degree-3*), and 6 (*degree-4*), where the horizontal axis is the training time. According to Figs. 2-6, we find that the resulting test accuracies on three datasets increase when decreasing the factor p. For examples, under *degree-1* ($p = 1.0$) in Figs. 2 and 3, the optimal test accuracies are 0.61, 0.63 and 0.37 on CIFAR-10, EMNIST and IMAGE-100, respectively. While in Fig. 6, under *degree-4* ($p = 0.2$), the optimal test accuracies of different methods reach 0.7, 0.81 and 0.4 on three datasets. Besides, by Figs. 4-6, with the degradation of factor p, it always requires less time for all the methods to achieve the convergence. For instance, in Fig. 4(c), it requires 5,981s for FedAvg to obtain the test accuracy of 0.36. By Fig. 6(c), the completion time for FedAvg to reach the test accuracy of 0.40 is 3,972s. Moreover, according to Figs. 2 and 4-6, AFO shows a increased performance with the degradation of factor p. For example, in Fig. 2(b), the completion time of AFO is higher than those of other four methods. However, in Fig. 6(b), AFO consumes less training time than FedAvg and CE-AFL to achieve the convergence, indicating that AFO may outperform the synchronous mechanisms under the *i.i.d.* setting.

According to Figs. 4-6, though the test accuracies of all methods increase in simulation experiments, the proposed algorithms (*i.e.*, FedLC and SC-FedLC) still achieve faster convergence than the existing methods. For example, in Fig. 5(b), when the training time equals 681s, the test accuracies of FedLC and SC-FedLC reach 0.77 and 0.79, which are close to the convergence. In comparison, the other methods (*i.e.*, FedAvg, CE-AFL and AFO) obtain the test accuracies of 0.62, 0.70 and 0.70 on EMNIST, which are far from the convergence. Besides, compared with the existing methods, it requires less training time for the FedLC and SC-FedLC to achieve the convergence. For instance, in Fig. 5(b), on EMNIST, it requires 923s and 786s for FedLC and SC-FedLC to obtain the test accuracy of 0.8. In contrast, to reach the test accuracy of 0.8 on EMNIST, the completion time of FedAvg, CE-AFL and AFO is 3,654s, 1,920s and 1,850s, which is much higher than the completion time of FedLC and SC-FedLC. Generally, according to Figs. 4-6, compared with the existing methods, SC-FedLC reduces the completion time by 35.9%, 51% and 37.2% on average over three datasets.

5.3 Results of Testbed Experiments

5.3.1 Impact of Heterogeneity Levels

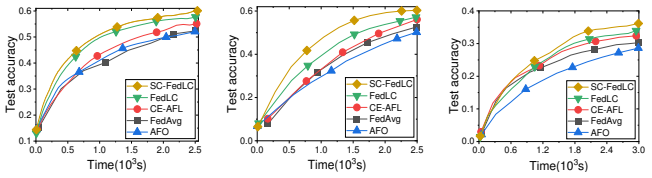
In order to evaluate the impact of different heterogeneity levels, we conduct the fourth set of experiments by empowering the participating Jetson devices with various performance modes. Specifically, the test accuracies of the models trained with different methods are plotted in Figs. 7 (*level-1*), 8 (*level-2*) and 9 (*level-3*), where the horizontal axis denotes the training time. According to Figs. 7-9, with the increased heterogeneity levels, it always requires more time for each method to reach the convergence, especially for FedAvg and CE-AFL. That is because the increased heterogeneity level will enlarge the performance gap between edge devices. While FedAvg and CE-AFL require that multiple devices participate in the global update, the waiting time among devices is magnified, and the epoch time is increased. Hence, the total completion time is also multiplied. Besides, by Figs. 7-9, we can see that AFO shows better performance than FedAvg and CE-AFL at high heterogeneity levels. For example, in Fig. 7(c), given the same training time of 3,000s, the resulting test accuracy of AFO is 0.28, which is much smaller than those of FedAvg (0.30) and CE-AFL (0.32). However, by Fig. 9, at *level-3*, the test accuracy of AFO is much higher than those of FedAvg and CE-AFL on three datasets, since AFO requires the participation of only one device and is more resistant to heterogeneity than the synchronous mechanisms.

Moreover, though the test accuracies of all methods increase with the training time in testbed experiments, FedLC and SC-FedLC still show faster convergence rate than the existing methods. For instance, by Fig. 9(b), when the training time equals 2,349s, the test accuracies of FedLC and SC-FedLC reach 0.52 and 0.57, which are almost converged. In contrast, the test accuracies of other three methods are 0.26, 0.32 and 0.41, which are far from the convergence. Besides, by Figs. 7-9, SC-FedLC always outperforms FedLC on three datasets at different levels, which further demonstrates the effectiveness of staleness compensation. Given the same training time, FedLC and SC-FedLC obtain higher test accuracies than existing methods on three datasets. Generally, compared with the existing methods, SC-FedLC improves the test accuracy by 6%, 6% and 5% on average over CIFAR-10, EMNIST and IMAGE-100, respectively.

From the above results, given the same training time, the proposed mechanisms (*i.e.*, FedLC and SC-FedLC) obtain higher test accuracies than those of existing methods, which definitely illustrates the effectiveness of local collaboration and staleness compensation.

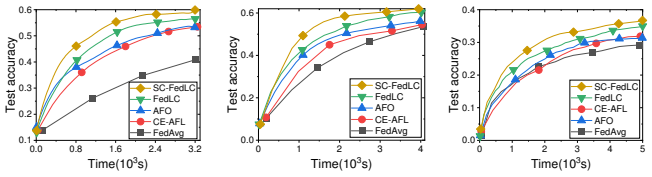
5.3.2 Varying the Number of Jetson Devices

In testbed experiments, we also conduct the scalability experiments by varying the number of participating Jetson devices. At the aforementioned three heterogeneity levels, we compare the performance of different methods on CIFAR-10 with various quantities of Jetson devices (from 15 to 50). Note that all the methods are performed with the same training time at each heterogeneity level. According to Figs. 7(a)-9(a), the training time budgets are specified as 2,500s, 3,200s and 4,500s for *level-1*, *level-2* and *level-3*, respectively. The test accuracies of the models trained with varying number of devices at different heterogeneity levels are depicted in Fig. 10, where the horizontal axis denotes the number of devices. By Fig. 10, we find that the test accuracies of all methods decrease with the number of Jetson devices. For the asynchronous mechanisms (AFO, FedLC and SC-FedLC), increasing the number of devices leads to the decline of quantity of samples on each device. While the asynchronous mechanisms require one participating device at each epoch, which may cause the deviation of global model. In terms of the semi-asynchronous (CE-AFL) and synchronous (FedAvg) mechanisms that require multiple participating devices at each epoch, increasing the number of edge devices will increase the possibility of block. Hence,



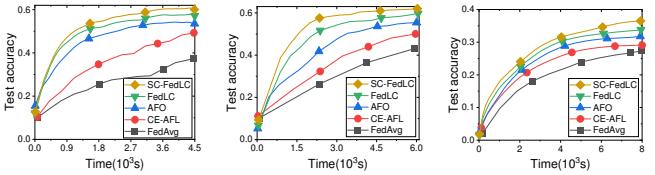
(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 7: Test accuracy curves of the models trained with different methods at *level-1*.



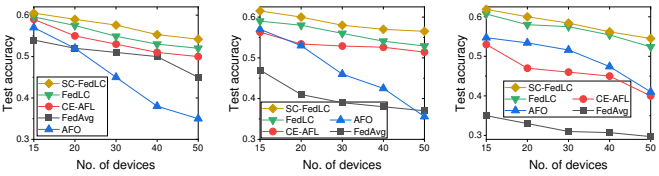
(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 8: Test accuracy curves of the models trained with different methods at *level-2*.



(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 9: Test accuracy curves of the models trained with different methods at *level-3*.

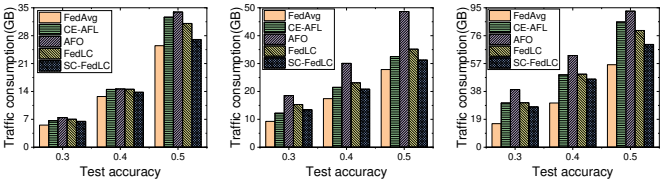


(a) *level-1* (b) *level-2* (c) *level-3*

Fig. 10: Test accuracy curves of the models trained with varying number of devices at different heterogeneity levels.

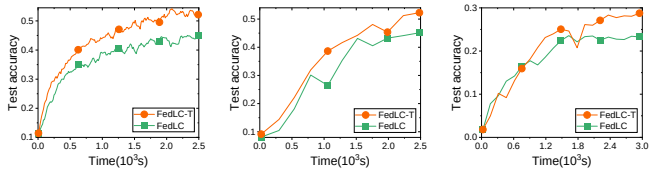
the training time for each epoch is increased, and the total completion time is also multiplied. Moreover, AFO shows a superior performance than FedAvg and CE-AFL at level-3, which is consistent with the result in Fig. 9.

According to Fig. 10, as the heterogeneity level increases, FedLC and SC-FedLC achieve higher gain on test accuracy compared with the existing methods. For instance, at *level-1*, the optimal test accuracies of SC-FedLC and CE-AFL are 0.60 and 0.59, and their gap is 0.01. While under *level-3*, SC-FedLC and CE-AFL obtain the test accuracies of 0.62 and 0.53, and their gap becomes 0.09. Moreover, no matter how many participating devices, FedLC and SC-FedLC achieve higher test accuracies than the existing methods at different heterogeneity levels. To be precise, at three heterogeneity levels, SC-FedLC improves the test accuracy over the existing methods by at most 19%, 21% and 25%, which fully proves the effectiveness of the proposed mechanisms.



(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 11: Traffic consumption of five methods when achieving the same test accuracy in testbed.



(a) CIFAR-10 (b) EMNIST (c) IMAGE-100

Fig. 12: Test accuracy curves of the models trained with FedLC and FedLC-T on three datasets.

5.3.3 *Comparison of Traffic Consumption*

To testify the communication efficiency of the proposed mechanisms, we conduct the sixth set of experiments by comparing the accumulated network traffic when different methods achieve the same test accuracy. The results are presented in Fig. 11, where the horizontal axis represents the target test accuracy. By Fig. 11, though the traditional FedAvg method shows a degraded performance in completion time, it outperforms the other four mechanisms (*i.e.*, CE-AFL, AFO, FedLC and SC-FedLC) in reducing the network traffic consumption. For instance, in Fig. 11(b), to obtain the test accuracy of 0.3 on EMNIST, the consumed traffic is 9.5GB, 12.5GB, 18.9GB, 15.7GB and 13.8GB for FedAvg, CE-AFL, AFO, FedLC and SC-FedLC, respectively. Though all the devices are required to participate in the training at each epoch, the total number of training epochs for FedAvg to reach the convergence can be decreased, and the communication overhead is naturally reduced. In comparison, AFO always consumes more network traffic for achieving the target test accuracy, since it requires much more communication epochs to reach the convergence. Besides, by Figs. 7-9 and 11, the semi-asynchronous mechanism (*i.e.*, CE-AFL) can always achieve a trade-off between completion time and traffic consumption. Similarly, the proposed mechanisms (*i.e.*, FedLC, SC-FedLC) consume the similar (or even less) amount of network traffic as that of CE-AFL for achieving the target test accuracies on three datasets. For example, by Fig. 11(c), it requires 87.3GB, 81.4GB and 71.7GB for CE-AFL, FedLC and SC-FedLC to obtain the test accuracy of 0.3 on IMAGE-100, which shows the communication efficiency of the proposed mechanisms.

From the above results, we find that the proposed mechanisms can improve the test accuracy while reducing the completion time, without dramatically increasing the communication overhead, which definitely illustrates the effectiveness of the proposed mechanisms.

5.3.4 Effect of Timeout Retransmission

We aim to evaluate the effect of timeout restransmission mechanism by comparing the performance of FedLC and FedLC-T under poor network conditions. The training time is specified as 2,500s, 2,500s and 3,000s for CIFAR-10, EMNIST and IMAGE-100, respectively. The test accuracy of FedLC and FedLC-T is compared, and the results are illustrated in Fig. 12. According to Fig. 12, we find that the test accuracy increases for both FedLC and FedLC-T. In addition, the test accuracy curves of FedLC and FedLC-T are fluctuating on three models. That is, FedLC and FedLC-T are performed under poor network condition where communication timeout occurs, leading to deviation of the global model, and the convergence curves become unstable. However, FedLC-T always achieves higher test accuracy than FedLC given the same training time. For example, by Fig. 12(a), when the training time equals 1,700s, FedLC obtains the test accuracy of 0.51 on CIFAR-10, which is 10% higher than the test accuracy of FedLC. With timeout retransmission, FedLC-T is expected to obtain more knowledge from the retransmitted models than FedLC, since FedLC will directly ignore the timeout models and just progress to further training. Generally, FedLC-T obtains the test accuracy of 0.53, 0.52 and 0.29 on three datasets, which is approximately 7% higher than that of FedLC, indicating the effectiveness of timeout retransmission mechanism.

6 Related Works

In recent years, Federated Learning (FL) has been widely adopted in both academia and industry fields [12], [13]. The existing FL approaches are mostly based on the parameter server (PS) framework [6]. In what follows, we mainly introduce three categories of PS-based FL, *i.e.*, synchronous FL [4], [16], [46], traditional asynchronous FL [21], [47], semi-asynchronous FL [12], [13], [24]. The above methods focus on the global collaboration among edge devices. Additionally, the intermediate collaboration is proposed as a variant of the global collaboration. Afterwards, we review the previous works of FL with intermediate and direct local collaboration [27], [48], [49].

6.1 Synchronous FL

The Federated Learning paradigm was first proposed in [4], which coordinates multiple edge devices to learn a globally shared model based on their local datasets. Concretely, McMahan *et al.* developed the FedAvg algorithm by combining the local stochastic gradient descent (SGD) on each client with a server that performs synchronous model averaging [4]. The FedAvg can decouple the model training from the need for direct access to raw data, and the data privacy can be preserved. However, Wang *et al.* noticed that the SGD-based FedAvg may incur high communication overhead due to the high-frequency transmission between edge devices and the server. In this way, Wang *et al.* proposed a synchronous FL solution given the resource budgets, which can achieve a trade-off between communication efficiency and model precision by adjusting the global aggregation frequency [16]. Similarly, to reduce the communication overhead on the server, Wang *et al.* proposed to construct a special cluster topology [15], and the communication overhead is reduced by merely aggregating the models from the cluster headers. Recently, InFEDge also achieved hierarchical FL in end-edge-cloud systems [50], which can significantly reduce the communication overhead and improve the resource utilization. However, the cluster is constructed according to the heterogeneous resources on edge devices, which may cause performance degradation on *non-i.i.d.* data. Moreover, Xu *et al.* also proposed to optimize the global aggregation frequency under the resource constraints in edge computing systems. Different from [16], Xu *et al.* proposed FedLamp, a synchronous FL solution that jointly optimizes global aggregation frequency and compression ratio [46]. For theoretical guarantee, Xu *et al.* analyzed the relationship between convergence bound and both aggregation frequency and compression ratio. The experimental results demonstrate that FedLamp can significantly reduce the traffic consumption and completion time compared with the existing methods.

The classical synchronous FL approaches can train a shared model by coordinating the edge devices without centralizing the local datasets, so as to prevent the user privacy leakage. However, the synchronous mechanism for FL may incur unbearable waiting time when edge devices possess heterogeneous resources (*i.e.*, synchronization barrier), since it requires the full participation of devices.

6.2 Traditional Asynchronous FL

The traditional AFL mechanism updates the global model after a local model from arbitrary device is received, and the negative effect of synchronization barrier can be completely addressed. However, due to the heterogeneous resources on edge devices, their participation frequency is always different, which may incur a large staleness on the stragglers. According to [25], a large staleness will cause the performance degradation of global model. Recently, several works were proposed to handle the model staleness. For example, Zheng *et al.* designed a novel mechanism, called DC-ASGD, to tackle the problem the model staleness. Concretely, in [47], Zheng *et al.* sought to approximate the optimization behavior of SGD with asynchronous SGD, by leveraging the Taylor expansion [51] of the gradient function and efficient approximation of Hessian Matrix [52]. From another perspective, Xie *et al.* proposed AFO [21], an asynchronous FL mechanism that adjusts the aggregation weight of each device according to the staleness, so as to address the model staleness. The convergence bound of AFO was presented in [21], and the experimental results show the effectiveness of AFO. However, in EC, the local data of edge devices usually are *non-i.i.d.* [4]. It has been verified that the *non-i.i.d.* local data may lead to the performance degradation (lower convergence rate and model accuracy), especially for asynchronous FL with staleness [24], [25]. As a consequence, it requires extra mechanisms for asynchronous FL to tackle the *non-i.i.d.* challenge. In addition, the traditional asynchronous mechanisms will incur significant communication resource consumption as they require frequent transmission between edge devices and the central server [24].

6.3 Semi-Asynchronous FL

To reduce the communication resource consumption in FL, the existing works have focused their attention on the semi-asynchronous FL, *i.e.*, the server updates the global model when a subset of all local models is received. Specifically, Liu *et al.* formulated the problem of asynchronous federated learning with resource constraints (AFL-RC) to minimize the loss function for model training. Then, a communication-efficient semi-asynchronous FL mechanism, CE-AFL, was designed to solve AFL-RC in [13], which controls the portion of participating devices with a factor $\alpha \in \{\frac{1}{N}, \frac{2}{N}, \cdots, 1\}$. Intuitively, CE-AFL looses the restriction of synchronization, and can alleviate the negative effect of synchronization barrier. However, CE-AFL adopts the constant and fixed α during training, which may cause the performance degradation considering the training dynamics. As an extension of CE-AFL, Liu *et al.* defined the problem of adaptive asynchronous federated learning with resource constraints (AAFL-RC) in EC. Afterwards, Liu *et al.* proposed to solve AAFL-RC by DAFL that adaptively adjusts the value of α based on the deep reinforcement learning (DRL) technique. Though DAFL can keep pace with the dynamic training process and reduce the completion time, the training of DRL model requires large quantities of data and is extremely time-consuming. Similarly, Ma *et al.* proposed a semi-synchronous FL mechanism, FedSA, which determines the number of participating devices [24]. The extensive experiments on real-world datasets illustrate that FedSA can significantly reduce the completion time compared with the baselines. The semi-asynchronous FL mechanism can alleviate the negative effect of synchronization barrier by loosing the restriction of synchronization. However, the above methods that select participating devices are not always effective, since the selected devices may be straggler devices [9].

6.4 FL with Intermediate and Direct Collaboration

In terms of the intermediate collaboration, edge devices collaborate with each other via model swapping [48] or cross aggregation [49] on the server, without relying on the global model aggregation. Concretely, FedSwap supports the model swapping between any two of all local models at the server, so as to improve the performance of global model. However, the swapping of local models is performed in a random manner, without considering the data distributions on devices, which will suffer from the performance degradation on *non-i.i.d.* data. Similarly, FedCross maintains multiple middleware models on the server and selects a collaborative model for each middleware model to aggregate [49]. Unlike FedSwap, FedCross selects the collaborative models according to the cosine similarity among local models, which can tackle the *non-i.i.d.* issue to some degree. Moreover, Sun *et al.* proposed a novel FL framework, named semi-decentralized FEEL (SD-FEEL), which incorporates the inter-cluster aggregation between edge servers [53], so as to address the *non-i.i.d.* issue. However, the inter-cluster communication between different edge servers is much more expensive than the communication among edge devices [27], since the former typically requires longer transmission distances and involves more complex network protocols. The above methods achieve the collaboration of edge devices by regarding the server as a relay station, which is different from the direct local collaboration in our FedLC.

Regarding the direct collaboration of devices, based on the DRL technique, FedMigr [27] proposes to migrate the local model from an edge device to another device by observing the environmental information (*e.g.*, data distribution variations), which can cope with the *non-i.i.d.* data. Nevertheless, regarding that the global model is synchronously updated, FedMigr may lead to non-negligible waiting time on heterogeneous edge devices. Additionally, training DRL models is computing-intensive and requires large amounts of training samples, which makes it difficult for FedMigr to work in resource-constrained EC scenarios. As a consequence, FedMigr is not incorporated as baseline method in this paper. In addition, Hu *et al.* proposed to split the model into several segments, and incorporated the local collaboration by transmitting the model segments between edge devices using Gossip learning [54]. Moreover, Hegedűs *et al.* made a thorough comparison of the Gossip learning and the centralized FL, and illustrated that the Gossip learning always outperforms the centralized solutions on *i.i.d.* data [55]. However, the Gossip learning may suffer from the performance degradation on *non-i.i.d.* data, since it is difficult to achieve the global consistency when lacking the global knowledge [56]. Similarly, Meng *et al.* proposed to perform topology construction among edge devices based on DRL, where the reward function is calculated by including the training loss, network connectivity and resource constraints [14]. To address the statistical heterogeneity, Liao *et al.* jointly optimized the local updating frequency and network topology in decentralized networks [57]. Regarding the *non-i.i.d.* data, the personalized FL was proposed by adopting Euclidean distance [58] and cosine similarity [45] as metrics for collaboration selection. It is worth noting that the proposed mechanism aims to accelerate the convergence rate of AFL in centralized networks. Hence, the above decentralized methods considering network topology are not adopted as baselines.

Different from the above works, we seek to address the critical challenges for FL in EC by combining asynchronous FL with local collaboration. By asynchronous FL, the negative effect of synchronization barrier can be completely eliminated. Meanwhile, with the local collaboration of edge devices, the challenge of *non-i.i.d.* data can be well addressed, and a satisfying convergence rate can be guaranteed.

7 Conclusion

In this paper, we focused on accelerating the asynchronous federated learning in EC by enabling the local collaboration among edge devices. We studied the impact of parameter k (number of collaborating devices) on training performance and obtained the relationship between the convergence bound and k. We proposed a demand-list based algorithm FedLC to transform the in-neighbor set to the out-neighbor set, and the waiting between edge devices can be avoided. To cope with the model staleness, we proposed staleness-compensated FedLC (*i.e.*, SC-FedLC), which assigns different learning rates to edge devices according to their partic-

ipation frequency. We conducted extensive simulation and testbed experiments on real-world datasets. The experimental results demonstrated the effectiveness of our proposed mechanism.

REFERENCES

[1] B. Chander and G. Kumaravelan, "Internet of things: Foundation," in *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*. Springer, pp. 3–33, 2020.

[2] X. Zheng, L. Tian, B. Hui, and X. Liu, "Distributed and privacy preserving graph data collection in internet of thing systems," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9301–9309, 2021.

[3] Z. Cai, X. Zheng, J. Wang, and Z. He, "Private data trading towards range counting queries in internet of things," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

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

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[6] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pp. 583–598, 2014.

[7] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[8] C. Van Berkel, "Multi-core for mobile phones," in *2009 Design, Automation & Test in Europe Conference & Exhibition*. IEEE, pp. 1260–1265, 2009.

[9] Z. Ma, Y. Xu, H. Xu, Z. Meng, L. Huang, and Y. Xue, "Adaptive batch size for federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 1, pp. 37–53, 2023.

[10] X. Wang, X. Ren, C. Qiu, Z. Xiong, H. Yao, and V. C. M. Leung, "Integrating edge intelligence and blockchain: What, why, and how," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 4, pp. 2193–2229, 2022.

[11] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[12] J. Liu, H. Xu, L. Wang, Y. Xu, C. Qian, J. Huang, and H. Huang, "Adaptive asynchronous federated learning in resource-constrained edge computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 2, pp. 674–690, 2023.

[13] J. Liu, H. Xu, Y. Xu, Z. Ma, Z. Wang, C. Qian, and H. Huang, "Communication-efficient asynchronous federated learning in resource-constrained edge computing," *Computer Networks*, vol. 199, p. 108429, 2021.

[14] Z. Meng, H. Xu, M. Chen, Y. Xu, Y. Zhao, and C. Qiao, "Learning-driven decentralized machine learning in resource-constrained wireless edge computing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, pp. 1–10, 2021.

[15] Z. Wang, H. Xu, J. Liu, Y. Xu, H. Huang, and Y. Zhao, "Accelerating federated learning with cluster construction and hierarchical aggregation," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2022.

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

[17] R. Carli, A. Chiuso, L. Schenato, and S. Zampieri, "A pi consensus controller for networked clocks synchronization," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10 289–10 294, 2008.

[18] Z. Wang, H. Xu, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, pp. 1–10, 2021.

[19] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 2021–2031, 2020.

[20] J. Hamer, M. Mohri, and A. T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *International Conference on Machine Learning*. PMLR, pp. 3973–3983, 2020.

[21] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.

[22] T. Chen, X. Jin, Y. Sun, and W. Yin, "Vafl: a method of vertical asynchronous federated learning," *arXiv preprint arXiv:2007.06081*, 2020.

[23] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous sgd," in *International Conference on Machine Learning*. PMLR, pp. 10 495–10 503, 2020.

[24] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3654–3672, 2021.

[25] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "A delayed proximal gradient method with linear convergence rate," in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, pp. 1–6, 2014.

[26] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[27] J. Liu, Y. Xu, H. Xu, Y. Liao, Z. Wang, and H. Huang, "Enhancing federated learning with intelligent model migration in heterogeneous edge computing," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE,, pp. 1586–1597 2022.

[28] J. Ho and C.-M. Wang, "Explainable and adaptable augmentation in knowledge attention network for multi-agent deep reinforcement learning systems," in *2020 IEEE Third International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. IEEE, pp. 157–161, 2020.

[29] Y. Chen, Y. Ning, M. Slawski, and H. Rangwala, "Asynchronous online federated learning for edge devices with non-iid data," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 15–24, 2020.

[30] H. Jamali-Rad, M. Abdizadeh, and A. Singh, "Federated learning with taskonomy for non-iid data," *IEEE transactions on neural networks and learning systems*, 2022.

[31] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–9, 2020.

[32] J.-P. Vial, "Strong convexity of sets and functions," *Journal of Mathematical Economics*, vol. 9, no. 1-2, pp. 187–205, 1982.

[33] R. Bhatia and C. Davis, "A cauchy-schwarz inequality for operators with applications," *Linear algebra and its applications*, vol. 223, pp. 119–129, 1995.

[34] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer Science & Business Media, vol. 87, 2003.

[35] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.

[36] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing tcp's retransmission timer," Tech. Rep., 2011.

[37] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," *arXiv preprint arXiv:2102.07078*, 2021.

[38] J. Zhang, Y. Zhao, J. Wang, and B. Chen, "Fedmec: improving efficiency of differentially private federated learning via mobile edge computing," *Mobile Networks and Applications*, vol. 25, no. 6, pp. 2421–2433, 2020.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[40] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "Emnist: Extending mnist to handwritten letters," in *2017 international joint conference on neural networks (IJCNN)*. IEEE, pp. 2921–2926, 2017.

[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet

large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[42] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, vol. 20, no. 5, p. 14, 2015.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[45] Z. Ma, Y. Xu, H. Xu, J. Liu, and Y. Xue, "Like attracts like: Personalized federated learning in decentralized edge computing," *IEEE Transactions on Mobile Computing*, 2022.

[46] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Transactions on Mobile Computing*, 2022.

[47] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z.-M. Ma, and T.-Y. Liu, "Asynchronous stochastic gradient descent with delay compensation," in *International Conference on Machine Learning*. PMLR, pp. 4120–4129, 2017.

[48] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-iid data for aiot service platform," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9266–9277, 2020.

[49] M. Hu, P. Zhou, Z. Yue, Z. Ling, Y. Huang, Y. Liu, and M. Chen, "Fedcross: Towards accurate federated learning via multi-model cross aggregation," *arXiv preprint arXiv:2210.08285*, 2022.

[50] X. Wang, Y. Zhao, C. Qiu, Z. Liu, J. Nie, and V. C. Leung, "Infedge: A blockchain-based incentive mechanism in hierarchical federated learning for end-edge-cloud communications," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3325–3342, 2022.

[51] G. B. Folland, "Higher-order derivatives and taylor's formula in several variables," *Preprint*, pp. 1–4, 2005.

[52] W. C. Thacker, "The role of the hessian matrix in fitting models to measurements," *Journal of Geophysical Research: Oceans*, vol. 94, no. C5, pp. 6177–6196, 1989.

[53] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning with data and device heterogeneity," *IEEE Transactions on Network and Service Management*, 2023.

[54] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *arXiv preprint arXiv:1908.07782*, 2019.

[55] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Distributed Applications and Interoperable Systems: 19th IFIP WG 6.1 International Conference, DAIS 2019, Held as Part of the 14th International Federated Conference on Distributed Computing Techniques, DisCoTec 2019, Kongens Lyngby, Denmark, June 17–21, 2019, Proceedings 19*. Springer, pp. 74–90, 2019.

[56] W. Wu, L. He, W. Lin, and C. Maple, "Fedprof: Selective federated learning based on distributional representation profiling," *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[57] Y. Liao, Y. Xu, H. Xu, L. Wang, and C. Qian, "Adaptive configuration for heterogeneous participants in decentralized federated learning," *arXiv preprint arXiv:2212.02136*, 2022.

[58] V. Zantedeschi, A. Bellet, and M. Tommasi, "Fully decentralized joint learning of personalized models and collaboration graphs," in *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 864–874, 2020.