# Adaptive Control of Local Updating and Model Compression for Efficient Federated Learning

Yang Xu, *Member, IEEE,* Yunming Liao, *Hongli Xu, *Member, IEEE,* Zhenguo Ma, Lun Wang, Jianchun Liu

**Abstract**—Data generated at the network edge can be processed locally by leveraging the paradigm of Edge Computing (EC). Aided by EC, Federated Learning (FL) has been becoming a practical and popular approach for distributed machine learning over locally distributed data. However, FL faces three critical challenges, *i.e.*, resource constraint, system heterogeneity and context dynamics in EC. To address these challenges, we present a training-efficient FL method, termed *FedLamp*, by optimizing both the **L**ocal upd**a**ting frequency and **m**odel com**p**ression ratio in the resource-constrained EC systems. We theoretically analyze the model convergence rate and obtain a convergence upper bound related to the local updating frequency and model compression ratio. Upon the convergence bound, we propose a control algorithm, that adaptively determines diverse and appropriate local updating frequencies and model compression ratios for different edge nodes, so as to reduce the waiting time and enhance the training efficiency. We evaluate the performance of *FedLamp* through extensive simulation and testbed experiments. Evaluation results show that *FedLamp* can reduce the traffic consumption by 63% and the completion time by about 52% for achieving the similar test accuracy, compared to the baselines.

**Index Terms**—*Edge Computing, Federated Learning, Local Updating, Model Compression.*

## 1 Introduction

With the development of mobile computing technology and Internet of Things (IoT), there have been nearly 7 billion connected IoT devices and 3 billion smartphones by 2020 all over the world [1]. The data source of the network has undergone a complete transformation from cloud data center to widely used terminal devices, on which the emerging technique of Federated Learning (FL) is launched to perform distributed machine learning over the distributed data [2]–[6]. Many interesting applications in different fields, *e.g.*, keyword spotting [7], medical imaging [8], finance risk prediction [9], *etc.*, have been boosted by FL.

In addition, due to the growing storage and computing capabilities on terminal devices, it is increasingly attractive to store data locally and push more applications that require high computing power to the network edge, which is called Edge Computing (EC) [10]–[12]. Aided by EC, FL has been widely deployed for various crowdsensing tasks and on-device intelligent applications, *e.g.*, human gesture recognition [13] and augmented reality (AR) [14], *etc.* In EC, the basic process of FL includes two main steps, *i.e.*, local updating on edge nodes and global aggregation on the parameter server (PS). Firstly, each edge node performs gradient descent to update its local model and minimize the loss function on its own dataset, and then sends the updated model to the PS. Subsequently, the PS aggregates these models from different edge nodes and sends the aggregated model back to the nodes for the next training round. Since the edge nodes expose not their raw training data but the trained models to the PS, FL can efficiently protect users' privacy [2], [15].

However, to implement highly efficient FL in EC, we should take into account the following challenges in practical applications. 1) *Resource Constraint.* In EC, the communication and computing resources of edge nodes are always limited [16]. In contrast, the edge nodes will frequently perform local updating and send/receive the models, causing an enormous resource overhead (*e.g.*, network bandwidth) [17]. 2) *System Heterogeneity.* The EC system involves heterogeneous edge nodes with varying capabilities [1], [18]–[20]. For example, there can be a tenfold difference in computing capabilities (*i.e.*, CPU frequency and battery life) or communication capabilities (*i.e.*, bandwidth, throughput) for the edge nodes as illustrated in Section 5. 3) *Context Dynamics.* Owing to the intrinsic and/or external conditions (*i.e.*, context), such as ambient temperatures, CPU states, network connections to the server and so on, the per-round training time (*i.e.*, computing time) and model aggregation time (*i.e.*, communication time) at each edge node may vary significantly [21], [22].

There are many previous works focusing on resource constraints in FL. A natural solution is to perform model training for multiple local iterations (one local iteration is referred to as local updating based on a sample/mini batch) before global aggregation, which contributes to reducing the overall communication rounds for global aggregation and

- *Y. Xu, Y. Liao, H. Xu, Z. Ma and L. Wang 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; liaoyun@mail.ustc.edu.cn; xuhongli@ustc.edu.cn; zgma@mail.ustc.edu.cn; wanglun0@mail.ustc.edu.cn.*
- *J. Liu is with the School of Data Science, 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: jsen617@mail.ustc.edu.cn.*
- *H. Xu is the corresponding author.*

saving the communication resource. Stich *et al.* [23] give a theoretical analysis of the convergence rate with respect to local updating frequency (*i.e.*, the number of local iterations between two successive global aggregations). Wang *et al.* [24] and Luo *et al.* [25] adaptively determine proper local updating frequency for participating nodes on the basis of both the constrained resource and dynamic context so as to realize communication-efficient FL and speed up the training process. To further reduce communication consumption, the technique of model compression [1], [26] is widely utilized by compressing the gradients or model parameters to be transmitted through sparsification [27]–[31] or quantization [32]–[38], for efficient model aggregation. For example, Nori *et al.* [39] consider both the communication frequency and communication data volume during training, and determine local updating frequency and model compression ratio for the participating nodes to further speed up training and reduce resource consumption. However, the existing works always assign identical or fixed model compression ratios and local updating frequencies for all the heterogeneous edge nodes, which can not make full use of the nodes' capacities. Although Han *et al.* [31] apply adaptive model compression to overcome the communication heterogeneity, they still cannot balance the computing time given the identical local updating frequency. For the popular synchronous FL, the synchronization barrier requires heterogeneous edge nodes to wait for others to finish local updating before model transmission and aggregation, which will incur non-negligible waiting time and deteriorate training efficiency [18], [19].

In this paper, we propose a training-efficient FL method, termed *FedLamp*, by optimizing both the **L**ocal upd**a**ting frequency and **m**odel com**p**ression ratio in the resource-constrained EC systems. Unlike the previous works, we explore to adaptively adjust the local updating frequencies and model compression ratios for different edge nodes. Generally, the models updated with more iterations of local updating will be assigned with larger compression ratios to reserve more parameters for global aggregation. However, a very large local updating frequency may lead the local models converging to the local optimal solutions rather than the global optima, which will hinder model convergence. The core mission of FedLamp is to quantify the relationship between local updating frequency and model compression ratio regarding the training performance, and jointly optimize these two variables for different edge nodes so as to better improve the training efficiency. The main contributions of this paper are summarized as follows:

- We design an efficient FL method, called FedLamp, which integrates adaptive control of local updating and model compression to better overcome the challenges of system heterogeneity and context dynamics in the resource-constrained EC systems.
- We theoretically analyze the model convergence rate and obtain a convergence upper bound related to the local updating frequency and model compression ratio. Upon the bound, we propose a control algorithm, that adaptively determines diverse and appropriate local updating frequencies and model compression ratios for different edge nodes, so as to reduce the waiting time and enhance the training efficiency.
- The performance of FedLamp is evaluated through extensive simulation and testbed experiments. The evaluation results show that FedLamp can reduce the traffic consumption by 63% and the completion time by about 52% for achieving the similar test accuracy, compared to the baselines.

The rest of this paper is organized as follows. Section 2 formalizes the optimization problem of our proposed FedLamp. Section 3 gives the convergence analysis of FedLamp. Based on the analysis, we propose an efficient algorithm to adaptively determine the local updating frequency and model compression ratio for each edge node in Section 4. Then we report our simulation and test-bed results in Section 5. Section 6 describes some related works and we conclude the paper in Section 7.

## 2 Preliminaries and Problem Formulation

### 2.1 Federated Learning in Edge Computing

In EC, there are $N$ workers (*i.e.*, edge nodes) and a parameter server (PS) constituting a edge computing cluster, where federated learning is proposed to solve the learning tasks through a loose federation of participating workers controlled by the PS (*e.g.*, edge server). The PS maintains a globally-shared model, *i.e.*, the global model, $\boldsymbol{w} \in \mathbb{R}^d$, where $\boldsymbol{w}$ is the model vector and $d$ is the dimension size. Each worker keeps its local dataset $\mathbb{D}_i$ and maintains a local model $\boldsymbol{w}_i \in \mathbb{R}^d$. The global loss function $F(\boldsymbol{w}) : \mathbb{R}^d \to \mathbb{R}$ in FL can be formulated as: [24], [40]

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w}) \triangleq \frac{1}{N}\sum_{i=1}^{N} F_i(\boldsymbol{w}_i) \tag{1}$$

where $F_i(\boldsymbol{w}_i) = \mathbb{E}_{\xi_i\sim\mathbb{D}_i}[F_i(\boldsymbol{w}_i)]$ is the local loss function corresponding to worker $i$, $\mathbb{E}_{\xi_i\sim\mathbb{D}_i}[\cdot]$ denotes expectation over a random sample $\xi_i$ chosen from the local dataset $\mathbb{D}_i$.

FL aims to find an optimal global model $\boldsymbol{w}^*$ that satisfies:

$$\boldsymbol{w}^* := \arg\min_{\boldsymbol{w}\in\mathbb{R}^d} F(\boldsymbol{w}) \tag{2}$$

Due to the inherent complexity of most machine learning tasks, trying to find a closed-form solution of Eq. (2) is always difficult [41]. Nonetheless, Eq. (2) can be solved by the gradient descent algorithms [18], [24]. For the mini-batch stochastic gradient descent (SGD), a gradient descent step over a mini batch on each worker is regarded as a local iteration (or a local update). After performing one or multiple local iterations, each worker first pushes its local model to the PS for global aggregation, and then pulls the updated global model from the PS for further training. Such a training process is regarded as a communication round. $\boldsymbol{w}_i^{(k,h)}$ denotes the local model of worker $i$ at the $k$-th iteration within communication round $h$. Therefore, the local updating step of worker $i$ is expressed as follows:

$$\boldsymbol{w}_i^{(k+1,h)} = \boldsymbol{w}_i^{(k,h)} - \eta \mathbf{g}_i^{(k,h)} \tag{3}$$

where $\mathbf{g}_i^{(k,h)} = \nabla F_i(\boldsymbol{w}_i^{(k,h)}, \xi_i^{(k,h)})$ is an unbiased stochastic gradient estimator and $\eta$ is the local learning rate.

TABLE 1: Key Notations.

| Symbol | Semantics |
|---|---|
| $N$ | number of workers |
| $\mathbb{D}_i$ | local dataset of worker $i$ |
| $\boldsymbol{w}$ | global model |
| $\boldsymbol{w}_i$ | local model of worker $i$ |
| $F(\boldsymbol{w})$ | global loss function |
| $F_i(\boldsymbol{w}_i)$ | local loss function of worker $i$ |
| $F(\boldsymbol{w}^*)$ | the optimal value of loss function $F(\boldsymbol{w})$ |
| $H$ | total number of communication rounds |
| $t^h$ | completion time of round $h$ |
| $t_i^h$ | completion time of round $h$ on worker $i$ |
| $\mathcal{W}^h$ | average waiting time of round $h$ |
| $B$ | bandwidth resource budget |
| $b_i$ | bandwidth consumption of worker $i$ |
| $C$ | computing resource budget |
| $c_i$ | computing resource consumption of worker $i$ |
| $\mu_i^h$ | computing time of one local iteration of worker $i$ at round $h$ |
| $\eta_i^h$ | communication time of one full model of worker $i$ at round $h$ |
| $\gamma_i^h$ | model compression ratio at round $h$ on worker $i$ |
| $\tau_i^h$ | local updating frequency at round $h$ on worker $i$ |
| $\varepsilon$ | waiting time threshold |

In the global aggregation step, the PS updates the global model $\boldsymbol{w}$ by aggregating the local models with global aggregation weight $\frac{\alpha_i}{N}$ corresponding to worker $i$ as follows:

$$\boldsymbol{w}^{h+1} = \boldsymbol{w}^h + \sum_{i=1}^{N} \frac{\alpha_i}{N}(\boldsymbol{w}_i^h - \boldsymbol{w}^h) \tag{4}$$

where $\sum_{i=1}^{N} \frac{\alpha_i}{N} = 1$. Some important notations in this paper are listed in Table 1.

### 2.2 Joint Optimization of Local Updating Frequency and Model Compression Ratio

In this section, we propose FedLamp to optimize both the local updating frequency and model compression ratio. Due to system heterogeneity, the computing time of one local iteration and transmission time of one full model among workers are highly different. As illustrated in Section 5.1, the highest-performance (or fastest) worker can be 10 times faster than the lowest-performance (or slowest) one. However, in traditional synchronous schemes, local updating frequencies and model compression ratios among workers are usually identical or fixed at each communication round. Accordingly, some fast workers have to wait for slow ones, incurring non-negligible waiting time and significantly reducing the training efficiency [18], [20]. Whereas, we propose to dynamically adjust the local updating frequencies and model compression ratios for different workers to address the system heterogeneity and context dynamics in the resource-constrained EC systems.

Considering the heterogeneous computing and communication capabilities of workers, before global aggregation, the workers with higher computing capabilities can perform more local iterations while the workers with lower computing capabilities only perform less local iterations. During model transmission, for saving communication resource, model compression is employed to compress the model parameters to be transmitted through sparsification [27]–[31]. We employ $\mathrm{top}_k$ sparsification [27] as our compression operator while other compression operators (*e.g.*, $\mathrm{random}_k$ [30]) can also be applied in FedLamp. The definition of $\mathrm{top}_k$ sparsification is as follows:

***Definition 1.*** Given a parameter $1 \leq k \leq d$, the compression operator $\mathrm{top}_k$: $\mathbb{R}^d \to \mathbb{R}^d$ is defined for $\boldsymbol{w} \in \mathbb{R}^d$ as

$$\mathrm{top}_k(\boldsymbol{w})_i := \begin{cases} \boldsymbol{w}_{\pi(i)}, & \text{if } i < k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $\pi$ is a permutation of $[d]$ such that $|\boldsymbol{w}_{\pi(i)}| \geq |\boldsymbol{w}_{\pi(i+1)}|$ for $i = 0, \cdots, d-1$. The compression ratio of $\mathrm{top}_k$ compression operator is defined as $\gamma = \frac{k}{d}$, where $k$ is the number of parameters retained after compression and $d$ the number of original parameters. As stated in [30], the $\mathrm{top}_k$ compression operator satisfies the following contraction property:

$$\mathbb{E}\|\boldsymbol{w} - \mathrm{top}_k(\boldsymbol{w})\|^2 \leq (1-\gamma)\|\boldsymbol{w}\|^2, \quad \forall \boldsymbol{w} \in \mathbb{R}^d \tag{6}$$

According to the contraction property, a smaller model compression ratio $\gamma$ can help to save more bandwidth resource but deteriorate the model quality and reduce training accuracy. Section 3 will further explain the relationship between model compression ratio and local updating frequency.

Generally, the workers with higher communication capabilities can apply larger model compression ratios to reserve more model parameters, while the workers with lower communication capabilities can apply smaller model compression ratios. After that, the waiting time among workers would be significantly reduced. We define the local updating frequency and model compression ratio at the $h$-th communication round on worker $i$ as $\tau_i^h$ and $\gamma_i^h$, respectively. The computing time of one local iteration and communication time of one full model at the $h$-th communication round on worker $i$ are denoted as $\mu_i^h$ and $\beta_i^h$, respectively. Considering the upload bandwidth is usually much smaller than the download bandwidth in typical WANs [3], [42], we focus on the communication time of pushing the models from workers to PS during the model exchanging. Therefore, the completion time of communication round $h$ (include computing time and communication time) on worker $i$ can be formulated as:

$$t_i^h = \tau_i^h \cdot \mu_i^h + \gamma_i^h \cdot \beta_i^h \tag{7}$$

Note that, the $\mathrm{top}_k$ compression operator needs to traverse all the parameters to obtain the largest k parameters, which can be implemented with the efficient quicksort-based selection method. We conducted a pre-experiment for training AlexNet on CIFAR-10 and recorded the time required by the compression algorithm and the completion time of each communication. Specifically, the average time taken by $\mathrm{top}_k$ is about 0.02s, while the average completion time of a communication round is about 4s. The compression time accounts for only about 0.5% in a certain communication round, and thus can be reasonably negligible.

Then, the waiting time of worker $i$ can be expressed as $t^h - t_i^h$, where $t^h = \max\{t_i^h | \forall i \in [N]\}$ denotes the

completion time of round $h$ on the slowest worker. Then the average waiting time of all workers at round $h$ can be formulated as:

$$\mathcal{W}^h = \frac{1}{N}\sum_{i=1}^{N}(t^h - t_i^h) \tag{8}$$

FedLamp ensures that the average waiting time will be small enough to mitigate the effects of the synchronization barrier.

### 2.3 Problem Formulation

This section defines the problem of efficient FL with adaptive local updating and model compression given the constrained communication and computing resources in EC systems. Herein, we mainly consider the resource consumption from the system point of view. For the sake of description, we adopt the summation form to formulate the computing and communication resource consumptions on workers during the whole training procedure, which are expressed in Eqs. (9) and (10). Due to system heterogeneity among workers, the local updating on workers will consume different amounts of bandwidth and computing resources. We define the computing resource consumption of one local iteration as $c_i$ and the transmission bandwidth consumption of one full model as $b_i$ on worker $i$. Assuming that the total computing resource budget and bandwidth resource budget are $C$ and $B$, respectively, the computing and bandwidth resource constraint are formulated as follows:

$$\sum_{h=1}^{H}\sum_{i=1}^{N}\tau_i^h \cdot c_i \leq C \tag{9}$$

$$\sum_{h=1}^{H}\sum_{i=1}^{N}\gamma_i^h \cdot b_i \leq B \tag{10}$$

Given an FL task in the EC system, we will determine the value of $\tau_i^h$ and $\gamma_i^h$ so as to minimize the training time $\sum_{h=1}^{H} t^h$. Accordingly, we formulate the problem as follows:

$$\min \sum_{h=1}^{H} t^h$$
$$s.t. \begin{cases} F(w^H) - F(w^*) \leq \epsilon \\ t_i^h = \tau_i^h \cdot \mu_i^h + \gamma_i^h \cdot \beta_i^h, \quad \forall i \in [N], \forall h \in [H] \\ \mathcal{W}^h = \frac{1}{N}\sum_{i=1}^{N}(t^h - t_i^h) \leq \varepsilon, \qquad \forall h \in [H] \\ \sum_{h=1}^{H}\sum_{i=1}^{N}\tau_i^h \cdot c_i \leq C \\ \sum_{h=1}^{H}\sum_{i=1}^{N}\gamma_i^h b_i \leq B \end{cases} \tag{11}$$

The first inequality expresses the convergence requirement, where $\epsilon > 0$ is the convergence threshold (close to zero) of the training loss between the optimal $F(\boldsymbol{w}^*)$ and $F(w^H)$ after $H$ training communication rounds. The second set of equalities denotes the formulation of the completion time of communication round $h$ on worker $i$. The third set of inequalities indicates that the average waiting time of all workers at each communication round should not exceed the predefined threshold $\varepsilon > 0$ (close to zero). The fourth inequality indicates the accumulated computing resource can not exceed the computing resource budget after $H$ training communication rounds. The last inequality indicates the accumulated bandwidth resource can not exceed the bandwidth resource budget after $H$ training communication rounds. Our objective is to minimize the training time under the resource constraints and performance requirements (*e.g.*, convergence, average waiting time).

## 3 Convergence Analysis

In this section, we analyze the convergence bound of the global loss function after $H$ communication rounds. For the sake of analysis, we make the following three assumptions as suggested in [24], [37], [38], [43].

***Assumption 1.*** (Lipschitz Continuous Gradient) There exists a constant $L > 0$, such that:

$$\|\nabla F_i(\mathbf{x}) - \nabla F_i(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d \tag{12}$$

***Assumption 2.*** (Unbiased Local Gradient Estimator) Let $\xi_i^h$ be a random local data sample at the $h$-th communication round on worker $i$. The local gradient estimator is unbiased as follows:

$$\mathbb{E}\left[\nabla F_i(\boldsymbol{w}_i^h, \xi_i^h)\right] = \nabla F_i(\boldsymbol{w}_i^h) \tag{13}$$

***Assumption 3.*** (Bounded Local Variance) There exists a constant $\sigma$, such that the variance of each local gradient estimator is bounded by:

$$\mathbb{E}\left[\|\nabla F_i(\boldsymbol{w}_i^h, \xi_i^h) - \nabla F_i(\boldsymbol{w}_i^h)\|^2\right] \leq \sigma \tag{14}$$

These assumptions are standard in non-convex optimization problems (*e.g.*, training deep neural networks) and commonly used in the analysis of FL with model compression [37], [38].

In order to facilitate the proof of Theorem 1, we present several important lemmas as follows. The proofs of the lemmas mainly refer to Lemmas D.1-D.3 in [37], which can be directly applied to support our lemmas with little modification. Therefore, to better highlight our main contributions, we omit the detailed proofs.

***Lemma 1.*** According to Assumption 1, the expected inner product between the full batch gradient and stochastic gradient can be bounded with:

$$\begin{aligned} &-\mathbb{E}\left[\left\langle \nabla F(\boldsymbol{w}^h), \mathbf{g}^h \right\rangle\right] \\ &\leq \frac{\eta}{2N}\sum_{i=1}^{N}\sum_{k=0}^{\tau-1}\left[-\|\nabla F(\boldsymbol{w}^h)\|^2 - \|\nabla F_i(\boldsymbol{w}_i^{(k,h)})\|^2\right] \\ &+ \frac{\eta}{2N}\sum_{i=1}^{N}\sum_{k=0}^{\tau-1}\left[L^2\|\boldsymbol{w}^h - \boldsymbol{w}_i^{(k,h)}\|^2\right] \end{aligned} \tag{15}$$

***Lemma 2.*** Under Assumption 2, Assumption 3 and contraction property Eq. (6) of $\mathrm{top}_k$, we have the following bound:

$$\mathbb{E}_{\mathrm{top}_k}\left[\|\mathbf{g}^h\|^2\right] \leq \tau(2-\gamma)\frac{1}{N}\sum_{i=1}^{N}\left[\sum_{k=0}^{\tau-1}\|\mathbf{g}_i^{(k,h)}\|^2 + \sigma^2\right] \tag{16}$$

***Lemma 3.*** Under Assumption 3, we have:

$$\mathbb{E}\left[\|\boldsymbol{w}^h - \boldsymbol{w}_i^{(k,h)}\|^2\right] \leq \eta^2\tau\sum_{k=0}^{\tau-1}\|\mathbf{g}_i^{(k,h)}\|^2 + \eta^2\tau\sigma^2 \tag{17}$$

Given these assumptions and lemmas, we have the following theorem about convergence bound.

***Theorem 1.*** If all local models are initialized at the same parameters $\boldsymbol{w}^0$, then the mean square gradient after $H$ communication rounds is bounded as follows:

$$\frac{1}{H}\sum_{h=0}^{H-1}\|\nabla F(\boldsymbol{w}^h)\|^2 \leq \frac{2\left(F(\boldsymbol{w}^0)-F(\boldsymbol{w}^*)\right)}{\eta\alpha\tau H}+\frac{\mathrm{L}\eta\,\alpha(2-\gamma)\sigma^2}{N} + L^2\eta^2\tau\sigma^2 \tag{18}$$

where $\tau=\max\{\tau_i^h\}$, $\gamma=\max\{\gamma_i^h\}$, $\alpha=\max\{\alpha_i^h\}$ and local learning rate $\eta$ satisfy:

$$\tau^2L^2\eta^2+(2-\gamma)\,\eta\alpha L\tau\leq 1 \tag{19}$$

The proof is presented in APPENDIX A.

On basis of Theorem 1, we can specify $\eta\alpha=\mathcal{O}(\frac{\sqrt{N}}{\sqrt{H\tau}})$ to achieve a linear speedup convergence rate, and further obtain the following convergence bound for our FedLamp with proper decisions of global aggregation weight $\frac{\alpha}{N}$ and local learning rate $\eta$.

***Corollary 1.*** Let $\eta=c_1\frac{\gamma}{\sqrt{H\tau L}}$ and $\alpha=c_2\sqrt{\tau N}$, where $c_1$ and $c_2$ are constant coefficients, then the convergence rate can be transformed as:

$$\frac{1}{H}\sum_{h=0}^{H-1}\|\nabla F\left(\boldsymbol{w}^h\right)\|^2 \leq \frac{2L\left(F(\boldsymbol{w}^0)-F(\boldsymbol{w}^*)\right)}{c_1c_2\gamma\sqrt{\tau HN}}+\frac{c_1c_2\gamma(2-\gamma)\sigma^2}{\sqrt{\tau HN}} + \frac{c_1^2\gamma^2\sigma^2}{H\tau} \tag{20}$$

With Corollary 1, FedLamp can achieve a linear speedup of convergence rate $\mathcal{O}(\frac{1}{\sqrt{\tau HN}})$ as many previous works [24], [37], [43], which indicates that FedLamp can contribute to saving resource consumption while still maintaining the similar convergence performance. Upon the setting $\alpha=c_2\sqrt{\tau N}$, we can formulate a rule to assign the global aggregation weight $\frac{\alpha_i^h}{N}$ for worker $i$ at round $h$, where $\alpha_i^h$ is expressed as follows:

$$\alpha_i^h=c_2\sqrt{\tau_i^hN} \tag{21}$$

which indicates that a model trained with larger local updating frequency will be assigned with a larger global aggregation weight, so as to improve overall training efficiency. Since $\sum_{i=1}^N\frac{\alpha_i^h}{N}=1$, we can obtain that $c_2=\frac{\sqrt{N}}{\sum_{i=1}^N\sqrt{\tau_i^h}}$.

Given the local learning rate $\eta=c_1\frac{\gamma}{\sqrt{H\tau L}}$, we can formulate a scaling rule to guide the setting of model compression ratios for different workers with respect to their local updating frequencies as follows:

$$\frac{\gamma_i^h}{\tau_i^h}=v \tag{22}$$

where $v=\frac{\eta L\sqrt{H}}{c_1}$ is a constant indicating that model compression ratio of worker is proportional to its local updating frequency. Then the local updating completion time (including computing time and communication time) at the $h$-th communication round on worker $i$ can be rewritten as:

$$t_i^h=\tau_i^h(\mu_i^h+v\cdot\beta_i^h) \tag{23}$$

Considering the above analysis, we can solve the problem in Eq. (11) by determining the local updating frequencies and model compression ratios for workers upon the computing time $\mu_i^h$ and communication time $\beta_i^h$, which will be elaborated in Section 4.

## 4 Algorithm Design

In this section, we first present an approximation of Eq. (11). Then, we propose an update algorithm that dynamically adjusts the local updating frequencies and model compression ratios for different workers. Finally, we analyze the time complexity of the proposed algorithm.

### 4.1 Approximation of Eq. (11)

In order to guarantee the convergence, we let the bound of mean square gradient be less than $\rho$ ($\rho$ is a positive number close to zero), which is equivalent to ensuring $F(\boldsymbol{w})-F(\boldsymbol{w}^*)\leq\epsilon$. Given the model compression ratio $0<\gamma\leq 1$ and $\gamma=v\cdot\tau$, we can formulate the convergence bound as:

$$g(H,\tau)=\frac{2L\cdot F(\boldsymbol{w}^0)}{v\cdot\tau\sqrt{\tau HN}}+\frac{v\cdot\tau\cdot\sigma^2}{\sqrt{\tau HN}}+\frac{v^2\cdot\tau\cdot\sigma^2}{H}\leq\rho \tag{24}$$

For purpose of minimizing the average waiting time of all workers, we let the $t_i^h$ among workers be approximately equal. Then we can have the following formulation:

$$\tau_i^h=\lfloor\tau_l^h\cdot\frac{\mu_l^h+v\cdot\beta_l^h}{\mu_i^h+v\cdot\beta_i^h}\rfloor \tag{25}$$

where $l$ denotes the index of the fastest worker with the largest local updating frequency $\tau$ at round $h$. Thus, $\tau_l^h=\tau$. Then the total training time can be formulated as follows:

$$T(H,\tau)=\sum_{h=1}^{H}\tau(\mu_l^h+v\cdot\beta_l^h) \tag{26}$$

Hence, the problem of Eq. (11) can be approximated as:

$$\min T(H,\tau)$$
$$s.t.\begin{cases} g(H,\tau)\leq\rho \\ \tau_i^h=\lfloor\tau\cdot\frac{\mu_l^h+v\cdot\beta_l^h}{\mu_i^h+v\cdot\beta_i^h}\rfloor,\quad\forall i\in[N],\forall h\in[H] \\ \sum_{h=1}^H\sum_{i=1}^N\tau_i^h\cdot c_i\leq C \\ \sum_{h=1}^H\sum_{i=1}^N v\cdot\tau_i^h\cdot b_i\leq B \end{cases} \tag{27}$$

### 4.2 Update Algorithm

On the basis of the above explanation, we propose an update algorithm to first estimate the maximum local updating frequency $\tau_l^{h+1}$ at the ($h$+1)-th communication round that minimizes $T(H,\tau_l^{h+1})$ according to the training information at the $h$-th communication round. Then $\tau_l^{h+1}$ is applied to calculate the local updating frequencies $\tau_i^{h+1}$ of other workers by Eq. (25) at round $h+1$ to minimize the average waiting time. With $\tau_i^{h+1}$ in each worker, $\gamma_i^{h+1}$ and $\alpha_i^{h+1}$ can

**Algorithm 1** Procedure at worker $i$

**Output:** $\boldsymbol{w}^H$

1: Estimate computing resource consumption $c_i$
2: Estimate bandwidth resource consumption $b_i$
3: Initialize $h \leftarrow 0$ and learning rate $\eta$
4: Initialize $\boldsymbol{m}_i^h$ as a zero vector
5: Send $c_i$ and $b_i$ to PS
6: **while** $STOP$ flag is not received **do**
7: Receive $\boldsymbol{w}^h$, $\tau_i^h$ and $\gamma_i^h$ from PS
8: Set $\hat{h} \leftarrow h$ and $h \leftarrow h+1$
9: **if** $h > 1$ **then**
10: Estimate $L_i \leftarrow \frac{\|\nabla F_i(\boldsymbol{w}^{\hat{h}}) - \nabla F_i(\boldsymbol{w}_i^{\hat{h}})\|}{\|\boldsymbol{w}^{\hat{h}} - \boldsymbol{w}_i^{\hat{h}}\|}$
11: Estimate
12: $\sigma_i \leftarrow \mathbb{E}\left[\|\nabla F_i(\boldsymbol{w}^{\hat{h}}, \xi_i^{\hat{h}}) - \nabla F_i(\boldsymbol{w}^{\hat{h}})\|^2\right]$
13: **end if**
14: Set $\boldsymbol{w}_i^{0,\hat{h}} \leftarrow \boldsymbol{w}^{\hat{h}}$
15: **for** each local iteration $k \in \{1, 2, ..., \tau_i^{\hat{h}}\}$ **do**
16: Compute $\mathbf{g}_i^{(k,\hat{h})} \leftarrow \nabla F_i(\boldsymbol{w}_i^{(k,\hat{h})}, \xi_i^{(k,\hat{h})})$
17: Update $\boldsymbol{w}_i^{(k+1,\hat{h})} \leftarrow \boldsymbol{w}_i^{(k,\hat{h})} - \eta \mathbf{g}_i^{(k,\hat{h})}$
18: **end for**
19: Set $\boldsymbol{w}_i^h \leftarrow \boldsymbol{w}_i^{(\tau_i^{\hat{h}},\hat{h})}$
20: Record computing time $\mu_i^{\hat{h}}$ and communication time $\beta_i^{\hat{h}}$
21: Push submodel $\text{top}_k(\boldsymbol{w}_i^h - \boldsymbol{w}^{\hat{h}} + \boldsymbol{m}_i^{\hat{h}})$, $\mu_i^{\hat{h}}$ and $\beta_i^{\hat{h}}$ to PS
22: $\boldsymbol{m}_i^h \leftarrow \boldsymbol{m}_i^{\hat{h}} + (\boldsymbol{w}_i^h - \boldsymbol{w}^{\hat{h}}) - \text{top}_k(\boldsymbol{w}_i^h - \boldsymbol{w}^{\hat{h}})$
23: **if** $h > 1$ **then**
24: Send $L_i$ and $\sigma_i$ to PS
25: **end if**
26: **end while**
27: Receive $\boldsymbol{w}^H$ from the server

**Algorithm 2** Procedure at the server

**Input:** computing and bandwidth resource budgets $C$ and $B$

**Output:** $\boldsymbol{w}^H$

1: Initialize $h \leftarrow 0$, $C^0 \leftarrow 0$, $B^0 \leftarrow 0$, $T^0 \leftarrow 0$
2: Initialize $\boldsymbol{w}^0$ as a random vector
3: Initialize $\tau_i^0$, $\gamma_i^0$, $\alpha_i^0$, $\tau_e$, $\tau_s$, $v$
4: Receive $c_i$ and $b_i$ from all workers
5: **while** $C^h \leq C$ and $B^h \leq B$ **do**
6: **for** $i = 1$ to $N$ **do**
7: Send $w^h$, $\tau_i^h$ and $\gamma_i^h$ to worker $i$
8: **end for**
9: Set $\hat{h} \leftarrow h$ and $h \leftarrow h+1$
10: Receive $\text{top}_k(\boldsymbol{w}_i^h - \boldsymbol{w}^{\hat{h}} + \boldsymbol{m}_i^{\hat{h}})$, $\mu_i^{\hat{h}}$ and $\beta_i^{\hat{h}}$ from all workers
11: $\boldsymbol{w}^h \leftarrow \boldsymbol{w}^{\hat{h}} + \sum_{i=1}^N \frac{\alpha_i^{\hat{h}}}{N} \cdot \text{top}_k(\boldsymbol{w}_i^h - \boldsymbol{w}^{\hat{h}} + \boldsymbol{m}_i^{\hat{h}})$
12: Calculate $t_i^{\hat{h}} \leftarrow \tau_i^{\hat{h}}(\mu_i^{\hat{h}} + v \cdot \beta_i^{\hat{h}})$
13: Update $C^h \leftarrow C^{\hat{h}} + \sum_{i=1}^N \tau_i^{\hat{h}} \cdot c_i$
14: Update $B^h \leftarrow B^{\hat{h}} + \sum_{i=1}^N \gamma_i^{\hat{h}} \cdot b_i$
15: Update $T^h \leftarrow T^{\hat{h}} + \max\{t_i^{\hat{h}}\}$
16: **if** $h \leq 1$ **then**
17: Update $\tau_i^h \leftarrow \tau_i^{\hat{h}}$, $\gamma_i^h \leftarrow \gamma_i^{\hat{h}}$, $\alpha_i^h \leftarrow \alpha_i^{\hat{h}}$
18: **else**
19: Estimate $\mu_i^h$ and $\beta_i^h$ with moving average
20: $l \leftarrow \arg\min_i(\mu_i^h + v \cdot \beta_i^h)$
21: Receive $L_i$ and $\sigma_i$ from all workers
22: $L \leftarrow \frac{1}{N}\sum_i^N L_i$
23: $\sigma \leftarrow \frac{1}{N}\sum_i^N \sigma_i$
24: Search $\tau_l^h \in [\tau_s, \tau_e]$ to minimize $T(H, \tau_l^h)$
25: **for** $i = 1$ to $N$ **do**
26: Update $\tau_i^h \leftarrow \lfloor \tau_l^h \cdot \frac{\mu_l^h + v \cdot \beta_l^h}{\mu_i^h + v \cdot \beta_i^h} \rfloor$
27: Update $\gamma_i^h \leftarrow v \cdot \tau_i^h$
28: **end for**
29: **for** $i = 1$ to $N$ **do**
30: Update $\alpha_i^h \leftarrow \frac{N\sqrt{\tau_i^h}}{\sum_{i=1}^N \sqrt{\tau_i^h}}$
31: **end for**
32: **end if**
33: **end while**
34: Send $STOP$ flag to all workers
35: Set $H \leftarrow h-1$ and send $\boldsymbol{w}^H$ to all workers

be obtained accordingly. It is noted that the variables $L$ and $\sigma$ may be unknown in practice, and need to be estimated in real-time during training, which will be explained later. Besides, the computing and bandwidth resource budgets $C$ and $B$, are initialized in advance and remain unchanged during training. $B^h$ and $C^h$ are regarded as the practical resource consumption by communication round $h$ and $T^h$ denotes the accumulated training time by communication round $h$. Furthermore, $\tau_i^{h+1}$ is explored in a search space to minimize $T(H, \tau_l^{h+1})$.

We present the update algorithm for workers (Alg. 1) and the server (Alg. 2) to achieve the efficient FedLamp. At the worker side, Alg. 1 begins with the estimation of computing and bandwidth resources consumption $c_i$ and $b_i$, which are sent to the server for calculating the resource consumption. The local updating is performed on each worker (Lines 14-18 of Alg. 1), and the rest of Alg. 1 aims to interact with the PS for global aggregation which is mainly implemented in Alg. 2. It is worth noting that we introduce error compensation (keeping track of accumulated errors in memory) in Lines 21-22 of Alg. 1, where $\boldsymbol{m}_i^h$ is initialized by a zero vector. Error compensation ensures that the model compression algorithm does not destroy the training effect, and better reflects the advantages of model compression. Let $\tau_i^h$ and $\gamma_i^h$ denote the local updating frequency and model compression ratio at round $h$ on worker $i$, respectively. In addition, $\frac{\alpha_i^h}{N}$ is the global aggregation weight of worker $i$ at round $h$. The value of $\tau_i^{h+1}$ is recomputed during each global aggregation step. There are two cases, depending on whether the latest variable estimations (*i.e.*, $L$ and $\sigma$) are accessible or not. (i) When the estimations are unavailable (*i.e.*, $h \leq 1$), $\tau_i^{h+1}$, $\gamma_i^{h+1}$ and $\alpha_i^{h+1}$ are set as $\tau_i^h$, $\gamma_i^h$ and $\alpha_i^h$ (Lines 16-17 in Alg. 2), respectively. (ii) When $h > 1$, we search the pre-defined space $[\tau_s, \tau_e]$ for a new $\tau_l^{h+1}$ which minimizes $T(H, \tau_l^{h+1})$, based on the updated estimations (Lines 19-24 in Alg. 2). Then the new $\tau_l^{h+1}$ is used to derive the local updating frequency $\tau_i^{h+1}$, model compression ratio $\gamma_i^{h+1}$, and $\alpha_i^{h+1}$ for global aggregation weight $\frac{\alpha_i^{h+1}}{N}$ for worker $i$ at round $h+1$ (Lines 25-31 of Alg. 2).

Next, we present the method to estimate the unknown variables such as $L$ and $\sigma$. The values of $L$ and $\sigma$ are estimated based on $L_i$ and $\sigma_i$ of worker $i$ (Lines 22-23 in Alg.

2 and Lines 9-13 in Alg. 1), and each worker has to maintain a replica of global variables and a version of local variables. Note that $F(\boldsymbol{w}^0)$ can be calculated straightforwardly when we initialize $\boldsymbol{w}^0$ as a random vector. As shown in Lines 13-14 of Alg. 2, in terms of the limited computing and bandwidth resource budgets in Eq. (9) and Eq. (10), the algorithm terminates when $C^h$ exceeds the budget $C$ (or $B^h$ exceeds the budget $B$). Then the penultimate communication round is regarded as the last one when the constraints Eq. (9) or Eq. (10) are satisfied. This explains why the server sets $H = h-1$ and sends $\boldsymbol{w}^H$ to each device (Lines 35 in Alg. 2).

### 4.3 Time Complexity Analysis

We now compare the time complexity of the typical distributed SGD and our algorithm. For the sake of analysis, we denote $U_f$, $U_b$ and $U_p$ as the time complexity of forward pass, backward pass and parameter update, respectively. For typical SGD, the forward pass, backward pass and parameter update are conducted iteratively, and the time complexity of typical SGD on a worker and the PS is $\mathcal{O}(\sum_{h=1}^{H}\tau^h(U_f+U_b+U_p))$ and $\mathcal{O}(H\cdot N\cdot U_p)$ [44], where $\tau^h = \frac{1}{N}\sum_{i=1}^{N}\tau_i^h$. On the worker side, compared with the typical SGD, Alg. 1 introduces extra cost on worker $i$ to estimate $L_i$ and $\sigma_i$ and compress $(\boldsymbol{w}_i^h - \boldsymbol{w}^{\hat{h}} + \boldsymbol{m}_i^{\hat{h}})$ with $\text{top}_k$ compression operator. To estimate $L_i$ and $\sigma_i$, we demand to calculate $\|\nabla F_i(\boldsymbol{w}^{\hat{h}}) - \nabla F_i(\boldsymbol{w}_i^{\hat{h}})\|$, $\|\boldsymbol{w}^{\hat{h}} - \boldsymbol{w}_i^{\hat{h}}\|$ and $\|\nabla F_i(\boldsymbol{w}^{\hat{h}}, \xi_i^{\hat{h}}) - \nabla F_i(\boldsymbol{w}^{\hat{h}})\|$ when $h \geq 2$, and the time complexity of $\|\cdot\|$ is $\mathcal{O}(2U_p)$. Hence, the time complexity to calculate $L_i$ and $\sigma_i$ is $\mathcal{O}((H-1)\cdot 6U_p)$. For the $\text{top}_k$ compression operator, it needs to traverse all the parameters to obtain the largest $k$ parameters, which can be implemented with the efficient quicksort-based selection method [27], and the $\text{top}_k$ compression operator incurs $\mathcal{O}(U_f)$ computational cost at a certain communication round. Therefore, the total time complexity of Alg. 1 is $\mathcal{O}(\sum_{h=1}^{H}\tau^h(U_f+U_b+U_p)+(H-1)\cdot(6U_p)+H\cdot U_f)$. Considering that $\mathcal{O}(\sum_{h=1}^{H}\tau^h(U_f+U_b+U_p))$ is far larger than $\mathcal{O}((H-1)\cdot(6U_p)+H\cdot U_f)$, the time complexity of Alg. 1 is $\mathcal{O}(\sum_{r=1}^{R}\kappa^r(U_f+U_b+U_p))$, which is similar to the time complexity of typical SGD. On the PS side, Alg. 2 requires more computational cost than typical SGD on calculating $\mu_i^h$, $\beta_i^h$, $\tau_i^h$ and $\alpha_i^h$, each of which requires computational cost of $\mathcal{O}(R\cdot N)$. Thus, the time complexity of Alg. 2 is $\mathcal{O}(R\cdot N\cdot(U_p+4))$, *i.e.*, $\mathcal{O}(R\cdot N\cdot U_p)$, which is also similar to that of typical SGD.

## 5 Experimentation and Evaluation

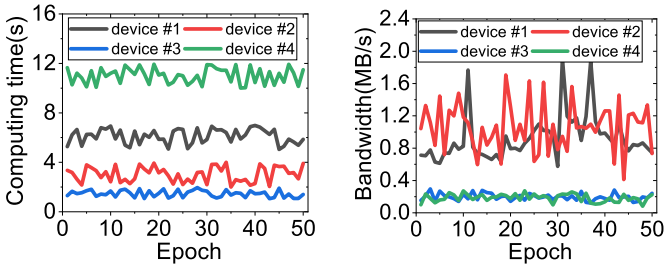
This section first conducts some intuitive tests to exhibit system heterogeneity and dynamics (Section 5.1). Then the datasets and models are described for experiments (Section 5.2), and we introduce baselines and metrics for performance comparison (Section 5.3). Simulation settings and results are given in (Section 5.4). Finally, we evaluate FedLamp through a small-scale test-bed experiment and give the results (Section 5.5).

### 5.1 Testing of System Heterogeneity and Dynamics

First of all, we conduct some intuitive tests to exhibit the heterogeneous capacities of the commercial edge devices, *e.g.*, NVIDIA Jetson TX2[1] and NVIDIA Jetson Xavier NX[2]. Both Jetson TX2 and Jetson Xavier NX can be configured to work under different performance modes, specifying the number of working CPUs and the frequency of CPU/GPU, so that they have different computing capacities. Moreover, Jetson Xavier NX accelerates the NVIDIA software stack with more than 10× the performance of Jetson TX2. The detailed technical specifications of the Jetson TX2 and Jetson Xavier NX are listed in Table 2. It is worth noting that the proposed algorithm and theoretical analysis in this paper are not limited to mentioned devices, like TX2 or NX. They can also be applied to various edge devices with different computing and communicating capabilities, *e.g.*, mobile phones, laptops and so on.

TABLE 2: Device Technical Specifications.

| | AI Performance | GPU Type |
|---|---|---|
| Jetson TX2 | 1.33 TFLOPS | 256-core Pascal |
| Jetson NX | 21 TOPS | 384-core Volta |
| | **CPU Type** | **ROM** |
| Jetson TX2 | Denver 2 and ARM 4 | 8 GB LPDDR4 |
| Jetson NX | 6-core Carmel ARM 8 | 8 GB LPDDR4x |
| | **CPU Frequency** | **GPU Frequency** |
| Jetson TX2 | 2.0GHz | 1.12GHz |
| Jetson NX | 1.9GHz | 1100MHz |




(a) Computing heterogeneity (b) Communication heterogeneity

Fig. 1: Illustrations of computing and communication heterogeneity of devices.

For computing test, we adopt two NX devices (device #1 with low-performance mode and device #3 with high-performance mode) and two TX2 devices (device #2 with high-performance mode and device #4 with low-performance mode) as the heterogeneous workers. Both the TX2 and NX devices are implemented to perform matrix multiplication over two 1500×1500 sized tensors for 30 times, and the computing time of each device is illustrated in Fig. 1(a). Device #4 takes the highest average computing time of 11.2s among all the devices, while device #3 needs the lowest average computing time (*i.e.*, 1.6s). The average computing time of device #2 and device #3 is 3.2s and 6.3s, respectively. The computing time of the low-performance device is almost seven times as long as that of the high-performance one, which obviously illustrates the computing heterogeneity.

1. More details about NVIDIA Jetson TX2 are available at https://developer.nvidia.com/embedded/jetson-tx2-developer-kit.

2. More details about NVIDIA Jetson Xavier NX are available at https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit.

Except for the computing heterogeneity among workers, the bandwidth of each worker may also be different (*i.e.*, communication heterogeneity). In EC, workers are usually connected to the edge server through wireless links, and the signal strength of the wireless link may vary with the transmission distance [45]. We put two NX devices and two TX2 devices in two different rooms, which are about 50 meters far from each other for the communication test. Concretely, device #1 and device #2 are close to the wireless router, while device #3 and device #4 are far away from the wireless router. Four devices synchronously send a 1500×1500 sized tensor to the server, which is also connected to the wireless router, and we record their communication bandwidth. By Fig. 1(b), the bandwidth of device #1 and device #2 is almost 6 times as much as that of device #3 and device #4, which reveals the communication heterogeneity.

### 5.2 Datasets and Models

**Datasets:** We conduct extensive experiments on three real-world datasets: (i) Fashion-MNIST[3] (FMNIST for short), (ii) CIFAR-10[4], and (iii) CIFAR-100[5]. Specifically, FMNIST [46] contains a training set with 60,000 samples and a test set with 10,000 samples. Each sample in FMNIST is a 28 × 28 grayscale fashion products from 10 categories. CIFAR-10 contains 60,000 32 × 32 color images labeled in 10 classes with 50,000 samples for training and 10,000 samples for test. CIFAR-100 has the same total number of image samples as CIFAR-10 but consists of 100 classes, which is more challenging to train models for classification. Except for the experiments for evaluating model performance on non-IID data, all datasets are distributed uniformly across workers by default.

**Models:** Three models with different types and structures are implemented on the above three real-world datasets for performance evaluation: (i) CNN on FMNIST, (ii) AlexNet on CIFAR-10, (iii) ResNet9 on CIFAR-100. The plain CNN model [3] specialized for the FMNIST dataset has two 5 × 5 convolutional layers, a fully-connected layer with 512 units, and a softmax output layer with 10 units. The CNN model is about 5M in size. An 8-layer AlexNet [47] which is composed of three 3 × 3 convolutional layers, one 7 × 7 convolutional layer, one 11 × 11 convolutional layer, two fully-connected hidden layers, and one fully-connected output layer is adopted for CIFAR-10. The size of AlexNet is about 15M, which is larger than the size of plain CNN model. For the CIFAR-100 dataset which is more challenging to train, a famous model ResNet9 [48] with size of 25M is adopted.

### 5.3 Baselines and Metrics

**Baselines:** We choose four classical and efficient algorithms as baselines for performance comparison, which are summarized as follows

- **FedAvg** [3] is a famous algorithm in federated learning with fixed (non dynamic) and identical (non diverse) local updating frequency for workers without model compression.
- **ADP** [24] adaptively determines the identical local updating frequency for all workers at each communication round on the basis of both the constrained resource and context dynamics so as to realize communication-efficient FL and speed up the training process.
- **Qsparse** [38] combines aggressive sparsification with quantization as model compression operator to reduce communication resource consumption without controlling the local updating frequency and model compression ratio.
- **FFL** [39] is a state-of-the-art algorithm, and it jointly and dynamically determines identical local updating frequency and model compression ratio for all workers at each communication round without considering the system heterogeneity.

3. https://github.com/TalwalkarLab/leaf
4. https://www.cs.toronto.edu/~kriz/cifar.html
5. https://www.cs.toronto.edu/~kriz/cifar.html

TABLE 3: Different strategies of baselines and FedLamp.

| algorithms | strategies | | | |
|---|---|---|---|---|
| | LUF* | | MCR* | |
| | dynamic | diverse | dynamic | diverse |
| FedAvg | × | × | - | - |
| ADP | ✓ | × | - | - |
| Qsparse | × | × | × | × |
| FFL | ✓ | × | ✓ | × |
| FedLamp | ✓ | ✓ | ✓ | ✓ |

* LUF denotes local updating frequency.
* MCR denotes model compression ratio.

The strategies of adjusting local updating frequency and model compression ratio in FedLamp and the baselines are summarized in Table 3.

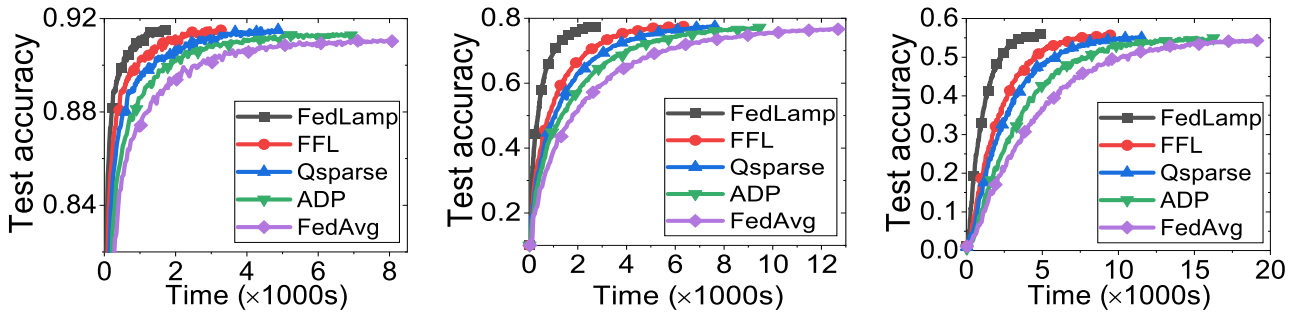
**Metrics:** The following metrics are adopted to evaluate the performance of our FedLamp and the baselines.

- **Test accuracy** is measured by the proportion between the amount of the right data predicted by the model and that of all data. Specifically, during the training process, at each communication round, we evaluate the test accuracy of the global model trained with different algorithms on the test sets.
- **Training loss** is the quantification difference of probability distributions between model outputs and the corresponding labels.
- **Completion time** is defined as the total training time until the global model achieves a target accuracy. In addition, we also record the average waiting time to reflect the training efficiency of different algorithms.
- **Network traffic** is calculated by summing the network traffic for model transmission between workers and the PS when achieving a target accuracy.

### 5.4 Simulation Experiments

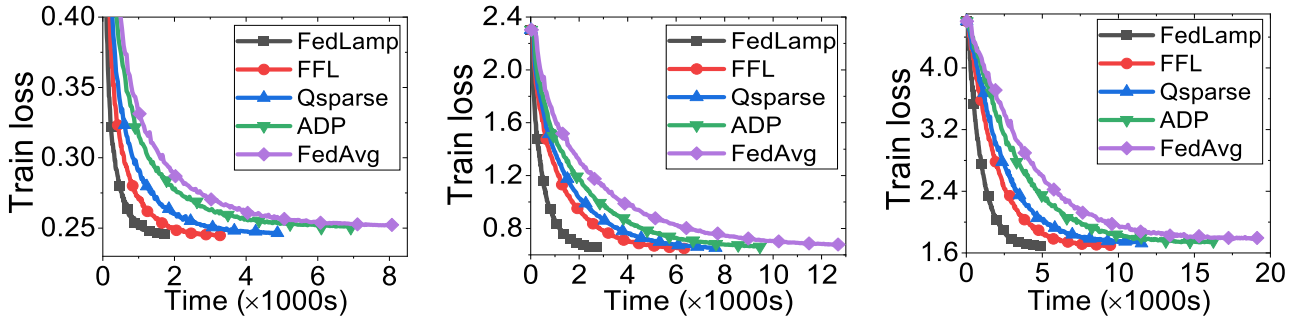
#### 5.4.1 Simulation Setup

We evaluate the performance of FedLamp through extensive simulation experiments, which are conducted on an AMAX deep learning workstation equipped with an Intel(R) Core(TM) i9-10900X CPU, 4 NVIDIA GeForce RTX 2080Ti GPUs and 128 GB RAM. On the workstation, we simulate a resource-constrained EC system with 10 workers and 1




(a) FMNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 2: Test accuracy of five algorithms on the three datasets in simulation.




(a) FMNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 3: Training loss of five algorithms on the three datasets in simulation.

PS (each is implemented as a process in the system) for federated learning. The implementation for model training on each worker is based on the PyTorch framework [49], and we use the socket library of Python to build up the communication between workers and the PS.

We consider the common situation where each worker communicates with the PS through either LANs or WANs. To reflect the heterogeneity and dynamics of networks in our simulations, we let the inbound bandwidth of each worker fluctuate between 10Mb/s and 20Mb/s. Considering the outbound bandwidth in typical WANs is usually smaller than the inbound bandwidth [3], we configure it to fluctuate between 0.5Mb/s and 5Mb/s. In addition, for simulating the computing heterogeneity, we assume the computing time of one local iteration on a certain simulated worker is subject to the Gaussian distribution. Different simulated workers are randomly assigned with a specific Gaussian function whose mean and variance are derived from the time records of performing one local update on a commercial device (*e.g.*, laptop, Jetson TX, Xavier NX).

In simulation experiments, the number of communication rounds are specified as 400, 500 and 500 for FMNIST, CIFAR-10 and CIFAR-100, respectively, which will guarantee the convergence of the models. For CNN on FMNIST and AlexNet on CIFAR-10, the learning rates are separately initialized as 0.05 and 0.1, and the corresponding decay rates are specified as 0.99 and 0.993. Besides, for ResNet9 on CIFAR-100, the momentum-SGD optimizer is adopted in our experiments to optimize the models, and the momentum is set as 0.9 while the weight decay is 0.001 [48]. Then the learning rate and the learning rate decay is initialized as 0.1 and 0.99, respectively. The batch size is set as 32 for all three models.

TABLE 4: Training performance of CNN on FashionMNIST.

| Metrics | | *FedLamp* | FFL | Qsparse | ADP | FedAvg |
|---|---|---|---|---|---|---|
| Accuracy = 91% | Accuracy | 91.51% | 91.38% | 91.32% | 91.31% | 91.03% |
| | Time (s) | 900 | 1778 | 2571 | 3805 | 6167 |
| | Traffic (MB) | 527 | 1436 | 718 | 3231 | 4491 |

TABLE 5: Training performance of AlexNet on CIFAR-10.

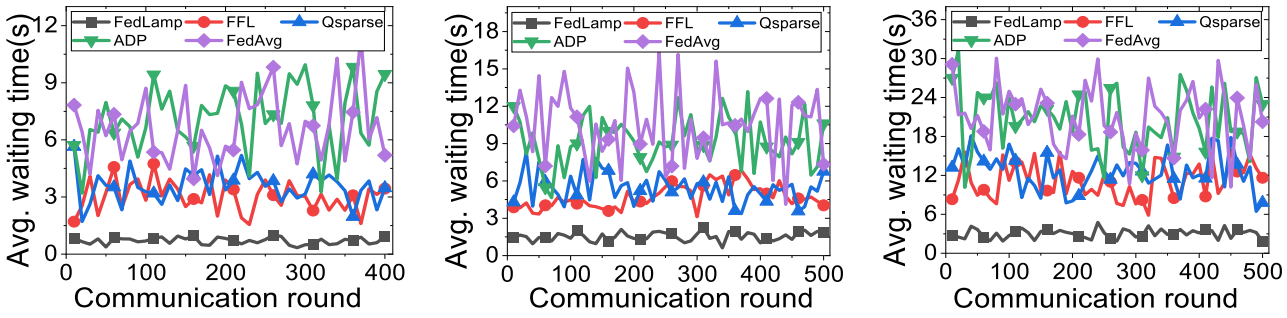
| Metrics | | *FedLamp* | FFL | Qsparse | ADP | FedAvg |
|---|---|---|---|---|---|---|
| Accuracy = 76% | Accuracy | 77.54% | 77.46% | 77.15% | 77.42% | 76.83% |
| | Time (s) | 2089 | 4918 | 5878 | 7570 | 10733 |
| | Traffic (MB) | 1352 | 3368 | 1684 | 10104 | 11227 |

TABLE 6: Training performance of ResNet9 on CIFAR-100.

| Metrics | | *FedLamp* | FFL | Qsparse | ADP | FedAvg |
|---|---|---|---|---|---|---|
| Accuracy = 54% | Accuracy | 55.93% | 55.72% | 54.99% | 54.9% | 54.31% |
| | Time (s) | 4057 | 7903 | 9716 | 13678 | 17075 |
| | Traffic (MB) | 2654 | 7455 | 3727 | 16773 | 18637 |

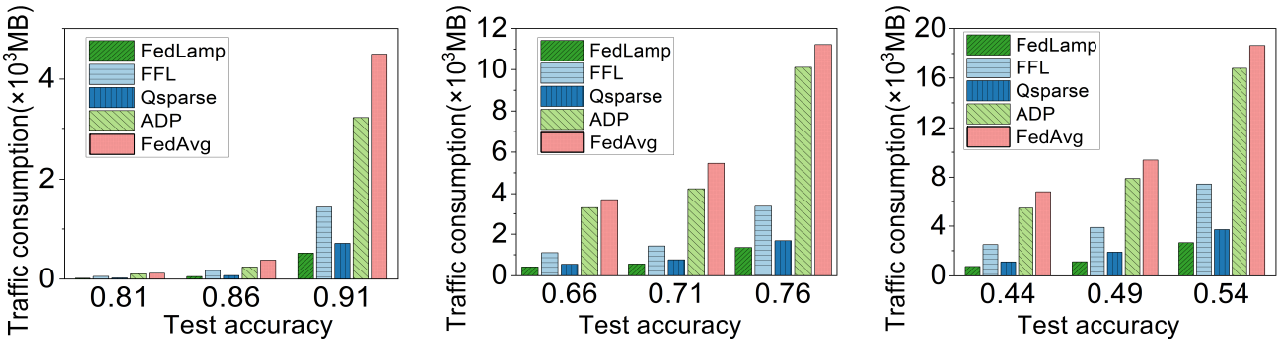
### 5.4.2 *Simulation Results*

We implement one group of experiments of these algorithms for the three models and datasets. The training processes of FedLamp and the baselines are presented in Fig. 2 and Fig. 3. The results demonstrate that all the algorithms can achieve similar accuracy but FedLamp achieves the fastest convergence rate for all the three datasets, which indicates the effectiveness of adaptively determining diverse and appropriate local updating frequencies and model compression ratios for heterogeneous workers. For example, by Table. 5 and Fig. 2(b), FedLamp takes 2089s to achieve 76%



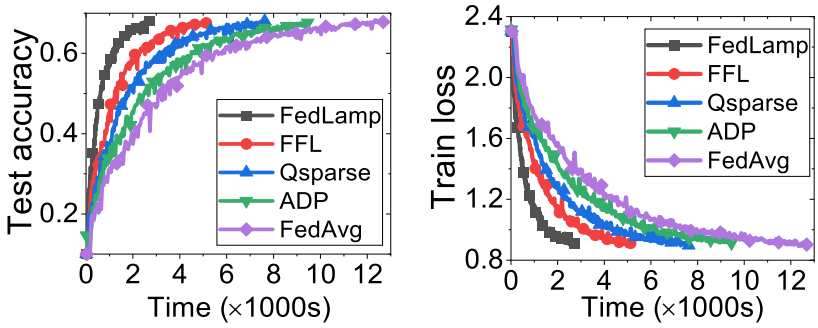

(a) FMNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 4: Average waiting time of five algorithms on the three datasets in simulation.




(a) FMNIST (b) CIFAR-10 (c) CIFAR-100

Fig. 5: Network traffic consumption of five algorithms when achieving the target accuracy on the three datasets in simulation.




(a) Test accuracy (b) Training loss

Fig. 6: Training process of five algorithms on the non-IID (p=0.8) CIFAR-10 dataset.

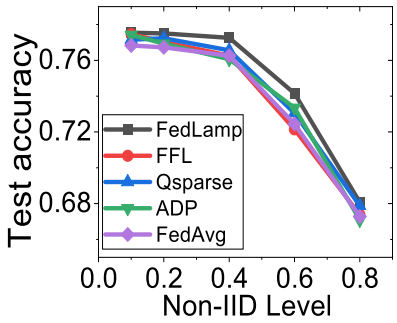



Fig. 7: Training performance for different non-IID levels.

accuracy for AlexNet on CIFAR-10, while FedAvg, ADP, Qsparse, FFL takes 10733s, 7570s, 5878s, 4918s, respectively. Besides, by Table. 4 and Fig. 2(a), we observe that FedLamp can speed up training by about 6.8×, 4.2×, 2.8× and 1.9 ×, compared to FedAvg, ADP, Qsparse and FFL, respectively. Moreover, by Table. 6 and Fig. 2(c) for ResNet9 on CIFAR-100, FedLamp can reduce the completion time of training by about 48%, 58%, 70% and 76%, compared to FedAvg, ADP, Qsparse and FFL, respectively. It is worth noting that in Tables 4, 5 and 6, the accuracy is set as the target that all the algorithms can achieve.

To further illustrate the efficiency of FedLamp, the average waiting time of all algorithms for three datasets are listed in Fig. 4, where the horizontal axis denotes the number of training communication rounds. By Fig. 4, we can find that FedLamp takes much less waiting time than FedAvg and ADP. For instance, by Fig. 4(b), the average waiting time of FedLamp is 1.5s while FedAvg and ADP incur 13s and 11.5s average waiting time, respectively. This is because both FedAvg and ADP assign identical local updating frequencies for workers without considering the impacts of computing heterogeneity and synchronization barrier, resulting in non-negligible waiting time. Moreover, the variance of waiting time in FedAvg is quite large while the waiting time in FedLamp is relatively stable, since FedAvg cannot deal with the challenge of context dynamics

in FL. Besides, ADP, which adaptively determines the local updating frequency at each communication round on the basis of both the constrained resource and dynamic context, can reduce the variance of waiting time to some extent. That explains why FedLamp can achieve much faster converge rate than FedAvg and ADP in Figs. 2 and 3 while ADP takes less completion time than FedAvg. The results in Fig. 4 demonstrate that FedLamp can well overcome the challenges of system heterogeneity and context dynamics.

To demonstrate the communication efficiency of FedLamp, we show network traffic consumption of different algorithms when they achieve different target accuracy in Fig. 5. We can find that the network traffic consumption of all algorithms for all datasets increases with the increasing accuracy. However, FedLamp can always consume the minimum network traffic. In addition, the algorithms with model compression can save much more network traffic than the algorithms without model compression. For example, by Table. 6 for ResNet9 on CIFAR-100, FedLamp, FFL and Qsparse consume 2654MB, 7455MB and 3727MB while ADP and FedAvg consume 16773MB and 18637MB, respectively, when they achieve the target accuracy. Moreover, model compression can reduce the communication time, which explains why Qsparse and FFL incur less average waiting time and completion time than ADP and FedAvg. However, both FFL and Qsparse assign identical local updating frequency and model compression ratio for different workers, without overcoming the system heterogeneity (including communication heterogeneity and computing heterogeneity), so that they take more waiting time and training time than FedLamp. With dynamic local updating frequencies and model compression ratios for workers, FFL can achieve faster convergence rate than Qsparse, but it cannot assign low model compression ratio, resulting in more network traffic compared to Qsparse. FedLamp, which is faster than FFL, can still achieve lower model compression ratio than Qsparse. Concretely, by Tables. 5 and 4, FedLamp can save network traffic consumption by about 59%, 19%, 86%, 87% for AlexNet on CIFAR-10, and by about 909MB, 191MB, 2704MB, 3964MB for CNN on FMNIST, compared to the baselines (FFL, Qsparse, ADP, FedAvg), respectively.

In a word, since FedLamp is designed to adaptively determine diverse and appropriate local updating frequencies and model compression ratios for heterogeneous workers, it can overcome the challenges of system heterogeneity, context dynamics in resource-constrained EC systems, and can take few waiting time and network traffic consumption to achieve fast convergence rate.

#### 5.4.3 Impacts of non-IID levels

Considering that the workers in FL collect data from their physical locations directly, the data on different workers in FL are not independent and identically distributed (*i.e.*, non-IID). Thus, we further evaluate FedLamp and the baselines on non-IID data. Concretely, we create non-IID data of the CIFAR-10 dataset with different partition schemes for the workers. Each worker has p (p = 0.1, 0.2, 0.4, 0.6 and 0.8) of a unique class in 10 classes and the remaining samples of each class are partitioned to other workers uniformly. Note that p = 0.1 is a special case, where the distribution of training dataset is IID. We denote the non-IID levels of CIFAR-10 as 0.1, 0.2, 0.4, 0.6 and 0.8. Note that the test datasets are partitioned uniformly among workers for a fair comparison. For the non-IID CIFAR-10 dataset, the training process of FedLamp and the baselines are presented in Fig. 6(a) and Fig. 6(b) in terms of training time. The experiment results demonstrate that even in the non-IID settings, FedLamp can still achieve the fastest convergence with higher test accuracy compared to the baselines. Besides, we show the impacts of non-IID levels on test accuracy of different algorithms in Fig. 7, where the horizontal axis is non-IID levels and the vertical axis is test accuracy. The result indicates that all algorithms suffer from a loss of accuracy with the increasing of non-IID level on CIFAR-10, while FedLamp is more robust than other baselines. In the future, we will pay more attention to the challenge of non-IID data and attempt to overcome it with more advanced techniques for FedLamp.

TABLE 7: Training performance of test-bed experiments.

| Metrics | | *FedLamp* | FFL | Qsparse | ADP | FedAvg |
|---|---|---|---|---|---|---|
| Accuracy = 76% | Accuracy | 77.06% | 76.06% | 76.52% | 76.17% | 76.86% |
| | Time (s) | 2109 | 4697 | 6748 | 10508 | 13450 |
| | Traffic (MB) | 1453 | 3488 | 1886 | 9134 | 10257 |

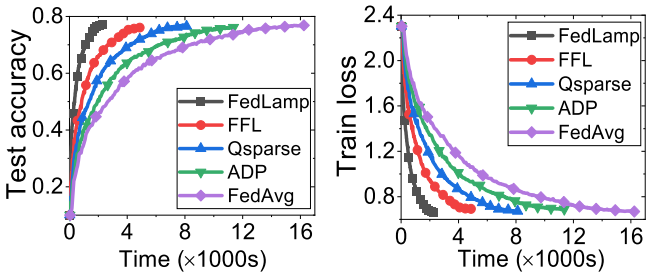
### 5.5 Test-bed Experiments

#### 5.5.1 Test-bed Setup

To better evaluate the performance of different algorithms on practical hardware platform, we further conduct some test-bed experiments. The test-bed experiments are performed on an AMAX deep learning workstation (CPU: Intel(R) E5-2620v4, GPU: NVIDIA GeForce RTX TITAN), 5 NVIDIA Jetson TX2 developer kits and 5 NVIDIA Jetson Xavier NX developer kits, where the AMAX workstation acts as the server while Jetson TX2 and Jetson Xavier NX serve as workers. The detailed technical specifications of the Jetson TX2 and Jetson Xavier NX are listed in Table 2. The experimental network is established via a router, where the server is directly connected to the router by Ethernet while Jetson TX2 and Jetson Xavier NX are accessed via wireless link. Besides, the Jetson TX2 and Jetson Xavier NX devices are powered with different modes to represent the computing heterogeneity and dynamics, and the different distances between the devices and the router as well as the random channel noise can result in the communication heterogeneity and dynamics. Our software implementation is based on the PyTorch deep learning framework [49], which enables the model training on the workers. Furthermore, communication among workers is established based on socket, which provides a set of sending and receiving functions contributing to much more efficient parallel communication. In addition, we can directly record the average waiting time, completion time and network traffic consumption in the test-bed experiments.

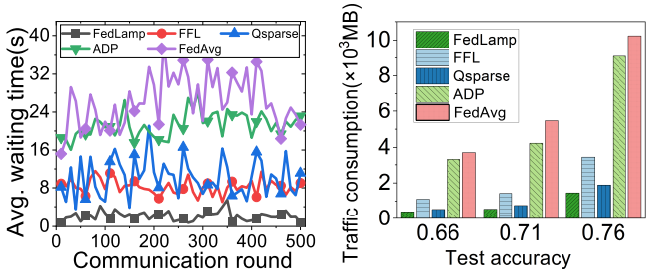
#### 5.5.2 Test-bed Result

We implement the training of AlexNet on CIFAR-10 in the test-bed environment. First of all, the training process of FedLamp and the baselines are shown in Fig. 8 and Fig. 9. Furthermore, we show the average waiting time and network traffic consumption of different algorithms in Fig. 9(a)




(a) Accuracy (b) Loss

Fig. 8: Training processes of the algorithms on CIFAR-10 in test-bed experiments.




(a) Average waiting time (b) Traffic vs. Accuracy

Fig. 9: Average waiting time and network traffic consumption of the algorithms on CIFAR-10 in test-bed experiments.

and Fig. 9(b). The results which are similar with the simulation results indicate that FedLamp can achieve similar accuracy with the baselines, but incurs shortest average waiting time and least network traffic, so that FedLamp achieves the fastest convergence rate. Concretely, the training performance of FedLamp and the baselines is listed in Table 7. We set the target accuracy as 76% as it is the accuracy that all algorithms can achieve. FedLamp takes 2109s and consumes 1453MB network traffic to achieve the target accuracy, which are the best performance compared with the baselines. ADP with dynamic local updating frequency takes less waiting time and converges faster than FedAvg, so that ADP saves more network traffic consumption than FedAvg. For instance, FedAvg needs 13450s and 10257MB network traffic while ADP needs 10508s and 9134MB network traffic when they achieve the target accuracy. Qsparse and FFL with model compression save a lot of network traffic and reduce the communication time. Thus, their average waiting time is less than that of ADP and FedAvg. Qsparse takes 6748s and consumes 1886MB network traffic while FFL only needs 4697s but consumes more network traffic (3488MB). This is because FFL with dynamic local updating frequency can further speed up training compared to Qsparse but cannot achieve lower model compression ratio than Qsparse. In a word, compared to the baselines (FFL, Qsparse, ADP, FedAvg), FedLamp can reduce the traffic consumption by 58%, 22%, 84%, 85% and the completion time by 55%, 68%, 79%, 84%, respectively. The test-bed results further prove the effectiveness and efficiency of our FedLamp.

## 6 Related Work

### 6.1 FL with Multiple Local Updates

Federated learning has become a practical and promising approach for distributed machine learning over distributed local data [2]–[6]. The concept of federated learning was first proposed in [3], which proved the effectiveness of federated learning through extensive experiments on various datasets. However, McMahan *et al.* [3] did not provide theoretical convergence guarantee for federated learning with fixed local updating frequency. Although Li *et al.* [50] analyzed the convergence of FedAvg on heterogeneous local data, they assumed the problems were strongly-convex and smooth, while we concentrate more on the popular non-convex problems involving the majority of Deep Neural Networks (DNNs). Besides, the approaches in [23], [51]–[53] only performed one local update before global aggregation, which significantly increase the model transmission frequency and result in a large amount of bandwidth consumption. From a theoretical perspective, the convergence analysis of distributed machine learning with multiple local updates was obtained in [23], [51]–[53]. Hsieh *et al.* [54] developed a geo-distributed machine learning system, which employed an intelligent communication mechanism with multiple local updates before global aggregation and a new synchronization method, called Approximate Synchronous Parallel (ASP). But the local updating frequency in [54] varies with the thresholding procedure and cannot be specified as a given constant. Communication-efficient distributed SGD algorithms with fixed local updating frequency were proposed in [55] and [56]. Whereas, the communication-efficient approaches in [55] and [56] were not suitable for federated learning in EC systems, since they did not consider the constrained computing resource, context dynamics and system heterogeneity.

In contrast to the above research, Wang *et al.* [24] proposed to dynamically determine the local updating frequency to optimize the model training in resource-constrained EC systems.

Moreover, Luo *et al.* [25] analyzed the convergence upper bound and established the relationship between the total cost (including computing and communication resources) and the local updating frequency. Nevertheless, they only considered the resource constraint but ignored the effect of synchronization barrier, resulting in non-negligible waiting time and low convergence rate. In addition, Yang *et al.* [43] explained that training with large local updating frequency would increase the noise of the system, resulting in the model training converging to the local optimal solutions instead of the global optimal solution [35], and they formulated the maximum local updating frequency in FL. Shi *et al.* [57] investigated how to develop energy-efficient FL over 5G+ mobile devices by making a trade-off between energy consumption for local updating and that for model transmission in order to boost the overall energy efficiency. The above methods always assign identical local updating frequencies for different workers. However, due to the system heterogeneity and context dynamics in EC, workers with identical local updating frequency have to wait for the slowest workers, resulting in low training efficiency. Thus, it is necessary to employ diverse local updating frequencies

for different heterogeneous workers, so as to reduce the waiting time and enhance the training efficiency.

### 6.2 FL with Model Compression

Performing multiple local updates in FL can reduce the overall communication rounds for global aggregation. Moreover, model compression is proposed as a commonly-used technique in distributed machine learning or FL to further reduce communication data volume during training [26]. Model or gradient compression can produce a compact model update, rather than a full model for transmission, through sparsification schema [27]–[31] or quantization operator [32]–[38].

For sparsification schema, it reduces communication consumption by transmitting an "important" sparse subset of the gradient for global aggregation [27], [58]. The convergence bound of sparsification schema is derived in [28]. Since the compression may introduce noise and reduce model accuracy, Stich *et al.* [30] introduced error compensation (keeping track of accumulated errors in memory) to improve the training performance with model compression.

For quantization operator, it quantizes the gradient (perhaps with randomization) to a small number of bits to save communication bandwidth. The application of quantization for efficient communication has decades history [59]. Gradient compression using unbiased stochastic quantizers has been theoretically analyzed and justified in [33], [36]. Wu *et al.* [32] analyzed error compensation for [33], and proposed error compensated quantized SGD for large-scale distributed optimization. Moreover, Basu *et al.* [38] combined a (stochastic or deterministic 1-bit sign) quantizer and sparsification with error compensation to further compress model and save communication resource, and Prakash *et al.* [60] incorporated quantization and model pruning to reap the benefits of DNNs while meeting the capabilities of resource-constrained devices.

However, all the above literatures employ fixed and identical model compression ratio for model training, which can not make full use of the workers' capacities due to system heterogeneity and context dynamics. Although Han *et al.* [31] applied adaptive model compression to overcome the communication heterogeneity, they still cannot balance the computing time by assigning the workers with the identical local updating frequency.

### 6.3 Combination of Multiple Local Updates and Model Compression

Considering the benefits of the above two strategies, Nori *et al.* [39] explored to jointly determine local updating frequency and model compression ratio for the workers to speed up training and reduce resource consumption. However, they only assigned identical local frequency and model compression ratio for all the heterogeneous workers, which still cannot make full use of the workers' capabilities and incurs non-negligible waiting time.

On the contrary, when taking the impacts of resource constraint, system heterogeneity and context dynamics simultaneously into consideration, we adaptively determine diverse and appropriate local updating frequencies and compression ratios for heterogeneous workers, so that our FedLamp can achieve the promising training performance with convergence guarantee.

## 7 Conclusions

In this paper, we focused on the critical challenges, *i.e.*, resource constraint, system heterogeneity and context dynamics for FL in edge computing. To overcome these challenges, we proposed FedLamp to jointly optimize both the local updating frequency and model compression ratio in FL. We theoretically analyzed the model convergence rate and obtained a convergence upper bound related to the local updating frequency and model compression ratio. We proposed a control algorithm to adaptively determine diverse and appropriate local updating frequencies and compression ratios for heterogeneous edge nodes, which contributes to reducing the waiting time under resource constraints and enhancing the training efficiency. We evaluated the performance of FedLamp through extensive simulation and testbed experiments with baselines and the results demonstrated the efficiency of FedLamp.

## Acknowledgments

This article is supported in part by the National Key Research and Development Program of China (Grant No. 2021YFB3301501); in part by the National Science Foundation of China (NSFC) under Grants 62132019, 62102391 and 61936015; in part by the Jiangsu Province Science Foundation for Youths (Grant No. BK20210122).

## References

[1] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 2031–2063, 2020.

[2] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2019.

[3] 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, 2017, pp. 1273–1282.

[4] 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.

[5] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.

[6] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[7] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6341–6345.

[8] N. Mouhni, A. Elkalay, M. Chakraoui, A. Abdali, A. Ammoumou, and I. Amalou, "Federated learning for medical imaging: An updated state of the art," *Journal homepage: http://iieta. org/journals/isi*, vol. 27, no. 1, pp. 143–150, 2022.

[9] ai.intel, "WeBank and Swiss re signed cooperation MoU," Aug. 2019. [Online]. Available: https://finance.yahoo.com/news/webank-swiss-signed-cooperation-mou-112300218.html.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[11] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[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*, pp. 1–1, 2021.

[13] H. F. T. Ahmed, H. Ahmad, and C. Aravind, "Device free human gesture recognition using wi-fi csi: A survey," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103281, 2020.

[14] Q. Liu and T. Han, "Dare: Dynamic adaptive mobile augmented reality with edge computing," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 2018, pp. 1–11.

[15] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[16] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," *Advances in Neural Information Processing Systems*, vol. 27, pp. 19–27, 2014.

[17] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "Enorm: A framework for edge node resource management," *IEEE transactions on services computing*, 2017.

[18] 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*, 2021.

[19] 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.

[20] J. Zhang, H. Tu, Y. Ren, J. Wan, L. Zhou, M. Li, and J. Wang, "An adaptive synchronous parallel strategy for distributed machine learning," *IEEE Access*, vol. 6, pp. 19 222–19 230, 2018.

[21] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proceedings of the Web Conference 2021*, 2021, pp. 935–946.

[22] Y. G. Kim and C.-J. Wu, "Autofl: Enabling heterogeneity-aware energy efficient federated learning," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 183–198.

[23] S. U. Stich, "Local sgd converges fast and communicates little," in *ICLR 2019-International Conference on Learning Representations*, no. CONF, 2019.

[24] 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.

[25] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[26] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright, "Atomo: Communication-efficient learning via atomic sparsification," *Advances in Neural Information Processing Systems*, vol. 31, pp. 9850–9861, 2018.

[27] X. Sun, X. Ren, S. Ma, and H. Wang, "meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3299–3308.

[28] D. Alistarh, T. Hoefler, M. Johansson, S. Khirirat, N. Konstantinov, and C. Renggli, "The convergence of sparsified gradient methods," *Advances in Neural Information Processing Systems 31*, pp. 5973–5983, 2019.

[29] Y. Lin, S. Han, H. Mao, Y. Wang, and B. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *International Conference on Learning Representations*, 2018.

[30] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4447–4458, 2018.

[31] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 300–310.

[32] J. Wu, W. Huang, J. Huang, and T. Zhang, "Error compensated quantized sgd and its applications to large-scale distributed optimization," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5325–5333.

[33] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.

[34] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signsgd: Compressed optimisation for non-convex problems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 560–569.

[35] 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, 2020, pp. 2021–2031.

[36] A. T. Suresh, X. Y. Felix, S. Kumar, and H. B. McMahan, "Distributed mean estimation with limited communication," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3329–3337.

[37] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 2350–2358.

[38] D. Basu, D. Data, C. Karakus, and S. N. Diggavi, "Qsparse-local-sgd: Distributed sgd with quantization, sparsification, and local computations," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 217–226, 2020.

[39] M. K. Nori, S. Yun, and I.-M. Kim, "Fast federated learning by balancing communication trade-offs," *IEEE Transactions on Communications*, 2021.

[40] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 8, pp. 1754–1766, 2020.

[41] 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, 2018, pp. 63–71.

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

[43] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-iid federated learning," in *International Conference on Learning Representations*, 2020.

[44] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4615–4625.

[45] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[46] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[48] D. Rothchild, A. Panda, E. Ullah, N. Ivkin, I. Stoica, V. Braverman, J. Gonzalez, and R. Arora, "Fetchsgd: Communication-efficient federated learning with sketching," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8253–8265.

[49] 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," in *Advances in neural information processing systems*, 2019, pp. 8026–8037.

[50] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.

[51] 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, 2017, pp. 4120–4129.

[52] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," *Advances in Neural Information Processing Systems*, vol. 28, pp. 2737–2745, 2015.

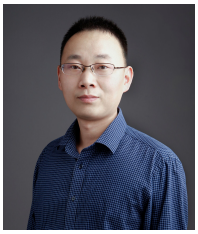
[53] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 2012, pp. 5451–5452.

[54] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: geo-distributed machine learn-

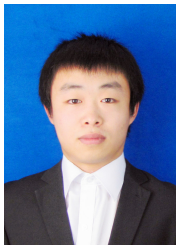
ing approaching lan speeds," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, 2017, pp. 629–647.
[55] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 212–229, 2019.
[56] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 5693–5700.
[57] D. Shi, L. Li, R. Chen, P. Prakash, M. Pan, and Y. Fang, "Towards energy efficient federated learning over 5g+ mobile devices," *arXiv preprint arXiv:2101.04866*, 2021.
[58] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," *arXiv preprint arXiv:1704.05021*, 2017.
[59] R. Gitlin, J. Mazo, and M. Taylor, "On the design of gradient algorithms for digitally implemented adaptive filters," *IEEE Transactions on Circuit Theory*, vol. 20, no. 2, pp. 125–136, 1973.
[60] P. Prakash, J. Ding, R. Chen, X. Qin, M. Shu, Q. Cui, Y. Guo, and M. Pan, "Iot device friendly and communication efficient federated learning via joint model pruning and quantization," *IEEE Internet of Things Journal*, pp. 1–1, 2022.

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

**Yunming Liao** received B.S. degree in 2020 from the University of Science and Technology of China. He is currently pursuing his M.S. degree in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include mobile edge computing and federated learning.



**Hongli Xu** (Member, IEEE) received the B.S. degree in computer science from the University of Science and Technology of China, China, in 2002, and the Ph. D degree in computer software and theory from the University of Science and Technology of China, China, in 2007. He is a professor with the School of Computer Science and Technology, University of Science and Technology of China, China. He was awarded the Outstanding Youth Science Foundation of NSFC, in 2018. He has won the best paper award or the best paper candidate in several famous conferences. He has published more than 100 papers in famous journals and conferences, including the IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, IEEE Transactions on Parallel and Distributed Systems, Infocom and ICNP, etc. He has also held more than 30 patents. His main research interest is software defined networks, edge computing and Internet of Thing.



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

**Lun Wang** received the B.S. degree in 2019 from the University of Electronic Science and Technology of China. He is currently pursuing his Ph.D. degree in the School of Computer Science and Technology, University of Science and Technology of China. His research interests include mobile edge computing and federated learning.

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

## APPENDIX A
## PROOF OF THEOREM 1

*Proof:* According to the smoothness property in Assumption 1 and the definition of stochastic gradient $\mathbf{g}^h$ in Eq. (3), we have:

$$F(\boldsymbol{w}^{h+1}) - F(\boldsymbol{w}^h) \leq -\alpha \left\langle \nabla F(\boldsymbol{w}^h), \mathbf{g}^h \right\rangle + \frac{\alpha^2 L}{2} \|\mathbf{g}^h\|^2$$

After applying the model compression technique with $\mathrm{top}_k$ operator, the expectation of the above inequality is expressed as:

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{E}_{\mathrm{top}_k}\left[F(\boldsymbol{w}^{h+1}) - F(\boldsymbol{w}^h)\right]\right] \\
&\leq -\alpha \mathbb{E}\left[\left\langle \nabla F(\boldsymbol{w}^h), \mathbf{g}^h \right\rangle\right] + \frac{\alpha^2 L}{2} \mathbb{E}\left[\mathbb{E}_{\mathrm{top}_k}\left[\|\mathbf{g}^h\|^2\right]\right]
\end{aligned}$$

For the sake of expression, we use $\mathbb{E}_{\mathrm{top}_k}\left[\nabla F(\boldsymbol{w}^h)\right]$ to denote $\mathbb{E}\left[\nabla F\left(\mathrm{top}_k(\boldsymbol{w}^h)\right)\right]$. We proceed to use Lemmas 1-3 to bound right hand side of the inequality, and obtain:

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{E}_{\mathrm{top}_k}\left[F(\boldsymbol{w}^{h+1}) - F(\boldsymbol{w}^h)\right]\right] \\
&\leq \frac{\alpha\eta}{2N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \left[-\|\nabla F(\boldsymbol{w}^h)\|^2 - \|\mathbf{g}_i^{(k,h)}\|^2\right] \\
&+ \frac{\alpha\eta}{2N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \left[L^2\eta^2 \sum_{c=0}^{\tau-1} \left[\tau \|\mathbf{g}_i^{(k,h)}\|^2 + \sigma^2\right]\right] \\
&+ \frac{(2-\gamma)\alpha^2 L}{2} \left[\frac{\eta^2\tau}{N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \|\mathbf{g}_i^{(k,h)}\|^2 + \frac{\tau\eta^2\sigma^2}{N}\right] \\
&\overset{①}{\leq} \frac{\alpha\eta}{2N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \left[-\|\nabla F(\boldsymbol{w}^h)\|^2 - \|\mathbf{g}_i^{(k,h)}\|^2\right] \\
&+ \frac{\alpha\eta}{2N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \left[\tau L^2\eta^2 \left[\tau \|\mathbf{g}_i^{(k,h)}\|^2 + \sigma^2\right]\right] \\
&+ \frac{(2-\gamma)\alpha^2 L}{2} \left[\frac{\eta^2\tau}{N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \|\mathbf{g}_i^{(k,h)}\|^2 + \frac{\tau\eta^2\sigma^2}{N}\right] \\
&= -\eta\alpha\frac{\tau}{2} \|\nabla F(\boldsymbol{w}^h)\|^2 \\
&- \left(1 - \tau^2 L^2\eta^2 - (2-\gamma)\eta\alpha L\tau\right) \frac{\eta\gamma}{2N} \sum_{i=1}^{N} \sum_{k=0}^{\tau-1} \|\mathbf{g}_i^{(k,h)}\|^2 \\
&+ \frac{L\tau\alpha\eta^2}{2N} \left(NL\tau\eta + \alpha(2-\gamma)\right)\sigma^2 \\
&\overset{②}{\leq} -\eta\alpha\frac{\tau}{2} \|\nabla F(\boldsymbol{w}^h)\|^2 + \frac{L\tau\alpha\eta^2}{2N} \left(NL\tau\eta + \alpha(2-\gamma)\right)\sigma^2
\end{aligned}$$

where in ① we incorporate outer summation $\sum_{k=0}^{\tau-1}$, and ② follows from Eq. (19). Finally, when summing up for all $H$ communication rounds and rearranging the terms, we can get Eq. (18), and the proof is completed. □