FedFa: A Fully Asynchronous Training Paradigm for Federated Learning

Haotian Xu[1] , **Zhaorui Zhang**[1]*, **Sheng Di**[2] , **Benben Liu**[3] ,
Khalid Ayed Alharthi[4] and **Jiannong Cao**[1]
[1]The Hong Kong Polytechnic University, Hong Kong
[2]Argonne National Laboratory, USA
[3]The University of Hong Kong, Hong Kong
[4]Department of Computer Science, College Of Computing,
University of Bisha, Bisha 61922, P.O. Box 551, Saudi Arabia.
haotian.xu@connect.polyu.hk, {zhaorui.zhang, jiannong.cao}@polyu.edu.hk,
sdi1@anl.gov, benbenliu@hku.hk, kharthi@ub.edu.sa

Abstract

Federated learning has been identified as an efficient decentralized training paradigm for scaling the machine learning model training on a large number of devices while guaranteeing the data privacy of the trainers. FedAvg has become a foundational parameter update strategy for federated learning, which has been promising to eliminate the effect of the heterogeneous data across clients and guarantee convergence. However, the synchronization parameter update barriers for each communication round during the training significant time on waiting, slowing down the training procedure. Therefore, recent state-of-the-art solutions propose using semi-asynchronous approaches to mitigate the waiting time cost with guaranteed convergence. Nevertheless, emerging semi-asynchronous approaches are unable to eliminate the waiting time completely.

We propose a full asynchronous training paradigm, called *FedFa*, which can guarantee model convergence and eliminate the waiting time completely for federated learning by using a few buffered results on the server for parameter updating. Further, we provide theoretical proof of convergence rate for our proposed *FedFa*. Extensive experimental results indicate our approach effectively improves the training performance of federated learning by up to $6\times$ and $4\times$ speedup compared to the state-of-the-art synchronous and semi-asynchronous strategies while retaining high accuracy in both IID and Non-IID scenarios.

1 Introduction

Federated learning has achieved great success in recent years thanks to its data privacy protection mechanism while also facing challenges, such as data heterogeneity and communication bottlenecks. The goal of federated learning is to train a good machine learning model to be used by all participants together under the condition that each participant doesn't send out their own data. It not only greatly expands the size of the available training dataset but also utilizes the computational resources of each participant. However, a price often arises from what is gained. Federated learning gains data privacy protection but faces more challenges in communication, resource scheduling, and data heterogeneity[Kairouz *et al.*, 2021]. In federated learning, the training clients are often located in different places connected by low bandwidth networks. The training data set is also diverse across different clients, which affects the convergence behaviors of federated learning significantly.

To deploy federated learning in practice, it is necessary for the federated learning systems not only to be accurate but also to satisfy a number of pragmatic constraints regarding issues such as efficiency and system performance. Emerging synchronous parameter update strategies, such as FedAvg, [McMahan *et al.*, 2017] and its variants, have become fundamental parameter update strategies in federated learning. However, due to the arithmetic and communication heterogeneity of the devices, the time required to complete the training task and transmit the model parameters to the server varies significantly across clients. Specifically, in FedAvg, the server needs to wait for all the selected clients to complete local training and report their model updates before aggregation. Such a design results in the time required for each round of federated learning training being determined by the slowest client, which leads to great waiting time and inefficiency in computational resources.

Scheduling the parameter update in an asynchronous paradigm [Zhang and Wang, 2022] is a new direction aiming to reduce the wall-clock time with a well-preserved target accuracy. The asynchronous parameter update strategy eliminates wait times at the system level, where the server does not need to wait for all its selected clients to report their local training results for the aggregation and will update the global model as soon as the model updates from one client arrives [Chai *et al.*, 2020; Zhang and Wang, 2021]. However, the distribution of client speeds is often heavy-tailed in federated learning. Fast clients update the global model frequently,

* Zhaorui Zhang is the corresponding author.

while slower clients make very little progress with the global model. The slower clients often report outdated training results to the server, affecting the global model's performance. To address this issue, most of the recent asynchronous strategies are semi-asynchronous [Nguyen *et al.*, 2022; Su and Li, 2022] or semi-synchronous [Zhang *et al.*, 2022] strategies, where the server will buffer several model updates from clients and then aggregate to update the parameters. These solutions, however, can only reduce part of the waiting time of global barriers but cannot eliminate them.

Why do the asynchronous federated learning approaches nowadays almost not use a fully asynchronous scheme? We identify three challenges for the fully asynchronous parameter update strategy in federated learning environments.

Firstly, the slower participant clients affect both the synchronous and asynchronous parameter update strategies, which causes long barrier times for model aggregation on the server for the synchronous scheme and results in the model performance degradation for asynchronous mode due to its large staleness. How to migrate the effectiveness of the slower participant clients for the performance of the global model is challenging in the asynchronous parameter update scheme in federated learning.

Secondly, federated learning is designed to meet each data owner's requirements for secure transmission and privacy protection, while the privacy protection will be broken when adopting the fully asynchronous parameter update strategies. The client updates the parameters with the server individually, making it easy for other clients to access the updated information from this client in the vanilla fully asynchronous parameter update strategy, which results in information leakage. The semi-asynchronous algorithm eliminates such information leakage by aggregating model updates from several clients, which is the same as synchronous strategies. However, semi-asynchronous and semi-synchronous parameter update strategies bring strong synchronization overhead, thus slowing down the training.

Thirdly, providing theoretical proof and guaranteeing the convergence of asynchronous parameter update strategies is challenging. This is caused by the uncertain sequence in which the server updates parameters from clients, which is heavily influenced by the network status and computation resources of participating clients.

In this article, to address the above challenges, we provide new insights about the three-parameter update paradigms for federated learning: synchronous, asynchronous, and semi-asynchronous and design a fully asynchronous parameter update strategy for federated learning, called *FedFa*.

We summarize our contributions as follows:

- We propose a fully asynchronous parameter update strategy without any barrier setting for federated learning, called *FedFa*, which updates the global model on the server once it receives an update request from clients, thus eliminating waiting time completely to improve the training performance (different from FedBuff). To mitigate the impact of model updates from slower clients, *FedFa* merges multiple historical model updates into currently received updates through a sliding window.
- We conduct an in-depth statistical analysis for the convergence rate of our proposed *FedFa*, deriving a theoretical bound, which can be extensively used to guarantee the quality of service in practice.
- We perform a comprehensive evaluation for *FedFa* on the most popular models, including the Language Model (Bert), CNN model (ResNet18), and Language Model FineTune, in both IID and Non-IID scenarios. Evaluation results show that *FedFa* improves the wall clock time by a factor of $6\times$ and $4\times$, respectively, and the number of communication rounds by a factor of $1.4\times$ and $1.9\times$, when compared to state-of-the-art synchronization and semi-synchronization methods.

2 Related Work and Motivations

Symbol	Description
k	a certain client
$[m]$	the full set of clients
w_g^t	t_{th} version parameters on server
$w_k^{t-\tau_k(t)}$	$t-\tau_k(t)$ version parameters on server
$\tau_k(t)$	staleness
τ_{max}	the max staleness
S^t	server's update buffer at phase t
K	size of the *Sliding Window*
η_g	global learning rate
η_l^q	local learning rate in step q
q	the q_{th} local steps
Q	the number of local update steps
T	communication rounds for converging
$\nabla F_k(w)$	gradient calculated on client k
$g_k(w;\zeta_k))$	stochastic gradient
α	data heterogeneity coefficient
β_t	aggregation weight for FedAsync
$\triangle_k^t$	accumulated gradients at t_{th} steps from client k
$s(\cdot)$	function of staleness for adaptive β_t
$f(\cdot)$	loss function
$\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(w^t\right)\right\|^2$	the convergence rate
L,σ_l,σ_g,G	symbolic upper bounds used in the proof
E	the number of local epochs
lb	local batch size

Table 1: Frequently Used Notations in this Article

2.1 Synchronous Federated Learning

Federated learning uses the synchronous parameter update strategy of when it was first proposed. However, as Federated Learning is put into practical use, the problem of wait times due to device heterogeneity becomes more and more serious. The synchronization paradigm can reduce wait times in several ways. For example, Fedprox [Li *et al.*, 2020] adaptively adjusts the strength of the training tasks assigned to different clients based on their computational power, such as freezing a certain percentage of model parameters based on the computing power of different clients to speed up training. All these methods aim to align the training time of different clients to reduce the waiting time. However, there are some problems with such approaches. On the one hand, such methods either require fine-tuning a very large number of parameters about the device or require the user to elaborate usage schemes, which is not easy to realize in practice. On the other hand, it may lead to additional communication consumption due to the increase in communication rounds. Moreover, this type of method can only reduce, but not eliminate, the waiting time since it is still a synchronized parameter update strategy.

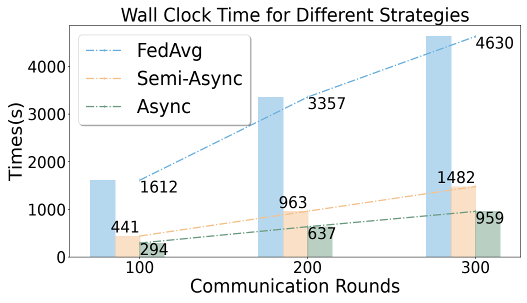


Figure 1: The wall clock time comparison.

2.2 Semi-asynchronous and Asynchronous FL

The synchronized parameter update strategy is not necessary for federated learning. Semi-asynchronous and asynchronous algorithms can greatly reduce or even eliminate waiting times right at the system level. Fig.1 shows the wall clock time required for the three parameter update strategies to complete the same amount of federated learning training. FedBuff [Nguyen *et al.*, 2022] is the most popular semi-asynchronous federated learning algorithm nowadays. FedBuff waits for several clients to complete the training task before performing parameter updates, which is **different** from our *FedFa*.

Fully asynchronous algorithm, Fedasync performs aggregation every time it receives an updated parameter, and the aggregation process is shown in the formula (1), where w_g^t and w_l^t indicate the parameters on the server and the parameters received by the server from a client at t_{th} steps, respectively. The aggregation weight β_t is calculated based on the staleness of the parameters by multiplying the initial aggregation weight β and $s(\cdot)$, which is a function of staleness for adaptive β_t. We also list the frequently used symbols in this article in Tab. 1. Previous works and our experimental results show that Fedasync often fails to converge to the target accuracy due to the high data heterogeneity.

$$\begin{cases} w_g^{t+1} = (1-\beta_t)\, w_g^t + \beta_t w_l^t \\ \beta_t = \beta \times s(t-\tau) \end{cases} \tag{1}$$

2.3 Security and Privacy Protection

SecAgg is a privacy-enhancing technique based on cryptographic primitives [Bonawitz *et al.*, 2016; So *et al.*, 2021] or hardware-based Trusted Execution Environment (TEE) [Karl *et al.*, 2020]. The server can only see the gradients after aggregation yet cannot know the original ones, which are private to each client. In this way, SecAgg ensures that the uploaded model is not accessible to the server and other malicious nodes or semi-honest nodes who follow the rules of engagement but are curious about the information of others.

Since most cryptographic algorithms nowadays are based on homomorphic encryption, they cannot handle the situation in semi-asynchronous, where the parameters are aggregated from different versions each time, and asynchronous algorithms, where only one parameter is aggregated at a time. Nowadays there are also new encryption algorithms, such as BASecAgg[So *et al.*, 2021], that can be applied to semi-asynchronous algorithms to fulfill the requirements of secagg.

Regardless of whether cryptographic algorithms or TEEs are used, the basic requirement for these methods to be usable is that updates from one client can be hidden from updates from multiple clients.

3 *FedFa*: Fully Asynchronous Federated Average for Federated Learning

The fully asynchronous parameter update strategy has been promised to eliminate the waiting time and reduce the wall clock time for federated learning, while it also faces challenges for poor convergence performance and model accuracy degradation. In this section, we provide a detailed explanation for our *FedFa*, a fully asynchronous parameter update strategy for federated learning that can eliminate the waiting time and guarantee the convergence of the federated learning.

3.1 The Design of *FedFa*

The fully asynchronous parameter update strategy in the federated learning environment often leads to the performance oscillation of the global model on the server due to the heterogeneity of the data across different clients. Thus, to avoid this performance degradation, we propose merging the historical version of the model collected by the server into the most updated one that arrives in the server for parameter update. We define a *Sliding Window* to buffer K historical version of the model that arrived in the server and merge them into the current one. The workflow of the server and client for our proposed *FedFa* is introduced in Algorithm 1 and 2.

Algorithm 1: *FedFa*-Server

```
Input: Sliding window size $K$; Client learning rate $\eta_l$;
       Client steps $Q$, Buffer Queue $S$
Output: Converged model
1  initialize t = 0 , $w_g^0$;
2  Broadcast the initialized server model parameters to the
     clients which are selected in the first round;
3  repeat
4      if get update $w_l^t$ then
5          if $t \leq K$ then
6              S.Enqueue($w_l^t$) ;
7          end
8          if $t > K$ then
9              S.Enqueue($w_l^t$) ;
10             S.Dequeue();
11             $w_g^{t+1} = \frac{1}{K} \sum_{k \in \mathcal{S}^t} w_k^{t-\tau_k(t)}$;
12         end
13         t = t+1;
14     end
15 until converge;
```

During training, once the number of updates received in the buffer equals K on the server, the regular training phase begins. When the buffer receives an update, the oldest update is removed so that the buffer always maintains the size of K. The parameters of each round on the server side are equal to the arithmetic mean of all parameters in the buffer. Aggregation is performed in a sliding window-like manner. If we do not perform this sliding window aggregation but wait for all the updates in the buffer to be replaced with newly received updates for each communication round, this is exactly

Algorithm 2: *FedFa*-Client

```
Input: Client learning rate $\eta_l$; Client steps $Q$; Server model
       parameters $w_g$; Local parameters at $q_{th}$ steps $y_q$;
       Function to calculate gradient $g(\cdot)$
Output: Client updated parameters $w_l$
1 $y_0 = w_g$;
2 $q = 0$;
3 while $q < Q$ do
4     $y_q = y_{q-1} - \eta_l g(y_{q-1})$;
5     $q = q + 1$;
6 end
7 $w_l = y_Q$;
8 send $w_l$ to server;
```

equivalent to the FedBuff with buffer size K. Instead, we use sliding windows for aggregation, tapping into some intermediate states between the two rounds to approach the global optimization goal quickly.

3.2 Gradient OR Parameter Transmission

In the federated learning environment, exchanging the parameters and gradients between the server and clients are two common approaches for parameter updates. In synchronous parameter update strategies, transmission parameters are equivalent to transmission gradients or accumulated gradients. This can be obtained from the following formula (2). In synchronous parameter update strategies, the server receives the updated parameters as $w_k^t = w_g^{t-1} + \triangle_k^t$, where the $\triangle_k^t$ is the accumulated gradient in client k, and uses it to update the model based on the following formula (2).

$$\begin{aligned} w_g^t &= \tfrac{1}{K}\sum_{k\in S_t} w_k^t = \tfrac{1}{K}\sum_{k\in S_t}(w_g^{t-1} + \triangle_k^t) \\ &= w_g^{t-1} + \tfrac{1}{K}\sum_{k\in S_t}\triangle_k^t \end{aligned} \tag{2}$$

In asynchronous and semi-asynchronous update strategies, the parameter update is calculated by $w_k^t = w_g^{t-\tau_k^t} + \triangle_k^t$. If parameters are transmitted between the server and clients, the parameter update of $w_k^t = w_g^{t-\tau_k^t} + \triangle_k^t$ is not equivalent to the above formula (2) due to the different phases at the start of training for each weight participating in the aggregation.

We implement and evaluate our *FedFa* based on both the parameters transmission and the difference transmission strategy, called *FedFa-Param* and *FedFa-Delta*, respectively.

4 Convergence Analysis of *FedFa*

4.1 Problem Formulation

The iterative formulation of *FedFa-Param* and *FedFa-Delta* can be outlined by equation (3) and (4). Both of them can achieve the same convergence rate as FedBuff [Nguyen *et al.*, 2022] as long as the following 5 assumptions hold.

$$w_g^{t+1} = \frac{1}{K}\sum_{k\in\mathcal{S}^t}\left(w_k^{t-\tau_k(t)}\right) \tag{3}$$

$$w_g^{t+1} = w_g^t + \frac{1}{K}\sum_{k\in\mathcal{S}^t}\left(\triangle_k^{t-\tau_k(t)}\right) \tag{4}$$

Assumptions: (without loss of generality)

- 1. *Unbiasedness of client stochastic gradient*
$$\mathbb{E}_{\zeta_k}[g_k(w;\zeta_k))] = \nabla F_k(w) \tag{5}$$
- 2. *Bounded local and global variance for all clients*
$$\mathbb{E}_{\zeta_k|k}\left[\left\|g_k\left(w;\zeta_k\right) - \nabla F_k(w)\right\|^2\right] \le \sigma_\ell^2 \tag{6}$$
- 3. *Bounded gradient* $\|\nabla F_k\|^2 \le G, k \in [m]$
- 4. *Lipschitz gradient for all client $k \in [m]$ and the gradient is L-smooth*
$$\left\|\nabla F_k(w) - \nabla F_k\left(w'\right)\right\|^2 \le L\left\|w - w'\right\|^2 \tag{7}$$
- 5. *Bounded Staleness.* We use $\tau_k(t)$ to denote the staleness of client k where we assume the server is in round t, and $\tau_k(t)$ has an upper bound τ_{max}:
$$1 \le \tau_k(t) \le \tau_{max} \tag{8}$$

4.2 The Proof for Convergence Rate of *FedFa*

Inspired by FedBuff [Nguyen *et al.*, 2022], we provide an analysis of the convergence rate for *FedFa* in this section. Several previous works also analyze the convergence of synchronous aggregation strategies [Li *et al.*, 2020; U, 2018; Wang *et al.*, 2020; Zhao *et al.*, 2018], and semi-asynchronous strategies [Nguyen *et al.*, 2022; Mania *et al.*, 2015; Fraboni *et al.*, 2023; Koloskova *et al.*, 2022]. Based on the above assumptions, the convergence rate of *FedFa* can be calculated as formula (9), where the $\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(w^t\right)\right\|^2$ indicates the convergence rate. The detailed proof of the convergence rate can be found in the full version of our paper.

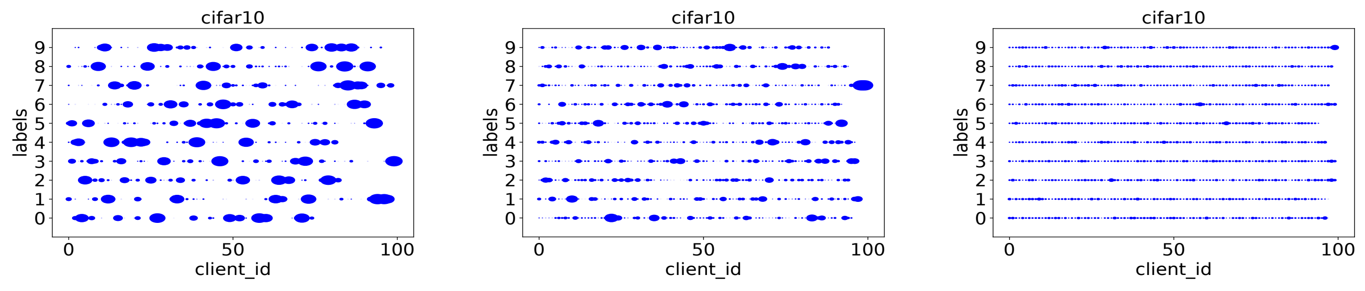
Theorem 1

$$\begin{aligned} &\frac{1}{T}\sum_{t=0}^{T-1}\left\|\nabla f\left(w_g^t\right)\right\|^2 \le \frac{2\left(f\left(w^0\right) - f\left(w^*\right)\right)}{\eta_g\sum_{q=0}^{Q-1}\eta_\ell^{(q)}T} \\ &+ 3L^2Q\sum_{q=0}^{Q-1}\left(\eta_\ell^{(q)}\right)^2\left(\eta_g^2\tau_{\max}^2 + 1\right)\left(\sigma_\ell^2 + \sigma_g^2 + G\right) \\ &+ \frac{L}{2}\frac{\eta_g\sum_{q=0}^{Q-1}\left(\eta_\ell^{(q)}\right)^2}{\sum_{q=0}^{Q-1}\eta_\ell^{(q)}}\sigma_\ell^2 \end{aligned} \tag{9}$$

Corollary. Final convergence rate is in formula (10).

$$\begin{aligned} &\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\left\|\nabla f\left(w_g^t\right)\right\|^2\right] \le \mathcal{O}\left(\frac{F^*}{\sqrt{TQ}}\right) \\ &+ \mathcal{O}\left(\frac{\sigma_\ell^2}{\sqrt{TQ}}\right) + \mathcal{O}\left(\frac{Q\sigma^2}{TK^2}\right) + \mathcal{O}\left(\frac{Q\sigma^2\tau_{\max}^2}{TK^2}\right) \end{aligned} \tag{10}$$

where $F^* := f\left(w^0\right) - f^*$, $\sigma^2 := \sigma_\ell^2 + \sigma_g^2 + G$, $\eta_\ell = \mathcal{O}(1/(K\sqrt{TQ}))$, $\eta_g = \mathcal{O}(K)$.



(a) $\alpha = 0.1$ (b) $\alpha = 0.5$ (c) $\alpha = 5$

Figure 2: Data distribution where a larger circle indicates a larger dataset size for each label 0-9 of CIFAR10.

The worst-case iteration complexity is captured by Corollary, which provides an upper bound on the ergodic norm-squared of the gradient—an essential metric in the study of non-convex stochastic optimization. If this norm-squared diminishes as the number of iterations (T) increases, it indicates that the norm-squared of the gradient at later iterations is approaching zero, suggesting convergence towards a first-order stationary point. Corollary 1 stems from Theorem 1 by employing a particular constant learning rate. It provides insights into the trade-offs involving loss convergence, local and global variance, the impact of client drift due to local steps, the influence of staleness, and the effect of buffer size. We can conclude that our both version of *FedFa* has the same convergence rate as Fedbuff statistically and theoretically.

5 Performance Evaluation and Analysis

5.1 Prototype Implementation

We implement *FedFa* on top of *Plota* [Su and Li, 2022] and *Pytorch*. Plota is a federated learning framework that supports temporal simulation for asynchronous federated learning on a single machine. Most of our experiments were performed on an NVIDIA GeForce RTX 3090 graphic card. For the task of fine-tuning the large language model, we performed it on 2× NVIDIA GeForce RTX 4090 graphic cards.

5.2 Evaluation Methodology

Benchmarks. For the CV task, we compare our method *FedFa* with other baselines on the most popular CV dataset, cifar10[K. and Geoffrey, 2009]. For the NLP task, we perform full-parameter fine-tuning on the sent140 [Caldas *et al.*, 2018] task and on the STT2 task using the parameter-efficient fine-tuning method LoRA [Hu *et al.*, 2021]. For these three tasks mentioned above, we perform experiments using the ResNet18 [He *et al.*, 2016], pre-trained Language Model Tiny-Bert [Bhargava *et al.*, 2021], and the pre-trained Language Model bert-base model, respectively. To simulate the Non-IID environments, we partition the whole training data based on the Dirichlet Distribution and use a coefficient α to control the heterogeneity of the data distribution across clients [Hsu *et al.*, 2019], where a small α represents higher data heterogeneity among each client, shown in Fig. 2. We subject the datasets Cifar10 and STT2 to the above division method that satisfies the Non-IID setting. The Sent140 is a dataset of sentiment categories collected from Twitter, which is naturally divided into federal settings by treating each user as a client, where we choose users with at least 100 samples.

Experiment setup. We deploy 100 clients and sample 10 clients for each communication round for ResNet18 trained on Cifar10. In the case of Tiny-Bert trained on Sent140, the number of clients is determined by the actual number of users in the dataset (i.e. 146) while maintaining 10 clients for training simultaneously. There was no manual setting for the number of clients. For the Bert-base model fine-tuned on the STT2 dataset, we deploy 10 clients and maintain 5 clients performing the training task simultaneously.

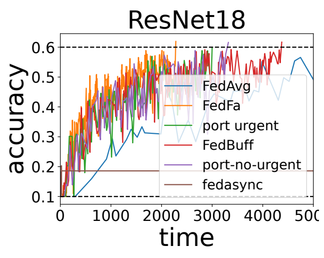
Simulation of delay time. In the experiments, each client needs to wait for a delay time t before communicating with the server for data transmission after completing the training task. This delay time t is simulated by the server and sent to each client. We set the simulated delay time for all clients with a long-tailed distribution. There are more clients with faster training and communication speeds and fewer clients with slower speeds.

Hyperparameters. For the ResNet18 on Cifar10, the learning rate η_g is set as 0.01 with local epoch $E = 10$ and local mini-batches $lb = 32$. For the Tiny-Bert experiments on the Sent140 dataset, we set $\eta_g = 0.0004, E = 15, lb = 5$, which is inspired by [Cho *et al.*, 2022]. For fine-tuning Bert on the STT2 dataset, we set $\eta_g = 1e-4, E = 1, lb = 32$. For the hyperparameter in LoRA settings, we set r = 1,$\alpha_{\text{LoRA}} = 1$.

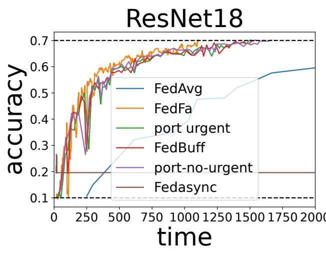
Baselines. We compare our *FedFa* with five other related approaches: (i) FedAvg [McMahan *et al.*, 2017], which is an approach in synchronous federated learning; (ii) FedBuff [Nguyen *et al.*, 2022], which is a semi-asynchronous approach for federated learning; (iii) FedAsync, the vanilla asynchronous approach for federated learning; (iv) Port [Su and Li, 2022], which is also a semi-asynchronous approach.

5.3 Results and Analysis

Time Efficiency of *FedFa*. In this section, we analyze the validation accuracy in terms of the wall clock time based on different benchmarks that are shown in Fig. 3(a)(b). These figures show that *FedFa* can achieve the same accuracy as semi-asynchronous and synchronous parameter update strategies using less time. Furthermore, the previous Fedasync strategy cannot converge to the target accuracy. We also give detailed results about the wall clock time to achieve the target accuracy for different benchmarks in Tab. 2. We find that in terms of time efficiency, the fully asynchronous algorithm is more efficient than the semi-asynchronous and synchronous algorithms due to the natural effects of the system design. The Fedasync algorithm fails to achieve the target accuracy.



(a) $\alpha = 0.1$



(b) $\alpha = 5$

Figure 3: Performance comparison of different training strategies on both IID and Non-IID data settings regarding the wall clock time.

To further evaluate the efficiency of our proposed *FedFa*, we also conduct experiments based on the parameter update strategy, called *FedFa-Delta*, in which clients send the parameter difference to the server, and the server sends the updated parameters to clients for each communication round. The parameter update strategy in which clients send parameters to the server and the server sends updated parameters to clients is called *FedFa-Para*. From Tab. 2, we observe that both of the two versions of our method outperform existing approaches, especially for *FedFa-Delta*. *FedFa-Delta* achieves up to $7\times$ and $5\times$ speedup compared to synchronous methods and semi-asynchronous methods, respectively. At the same time, from Fig. 3 and Tab. 2, we can see that Fedasync does not achieve the target accuracy, which also shows that our method can not only improve the time efficiency but also stabilize the training process.

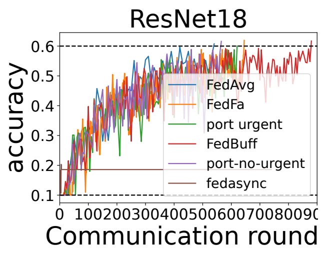
Dataset	Sent140	STT2	Cifar10	
	Non-IID	Non-IID	Non-IID	IID
Target-Acc	0.71	0.85	0.6	0.7
FedAvg	1139(6×)	4660(7×)	9833(5×)	5040(6×)
Fedasync	failed	failed	failed	failed
FedBuff	485(3×)	2985(4.5×)	4375(2×)	1541(2×)
Port-Urgent	767(5×)	2325(3.5×)	3000(1.6×)	1503(1.7×)
Port-Non-Urgent	662(4×)	2770(4.2×)	3320(1.7×)	1668(2×)
FedFa-Param	191(1.1×)	829(1.2×)	2282(1.2×)	1102(1.3×)
FedFa-Delta	**165**	**652**	**1917**	**855**

Table 2: Wall Clock Time for Target Accuracy

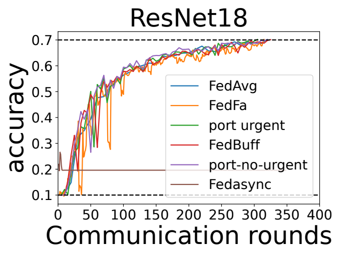
Communication Efficiency of *FedFa*. In this section, we analyze the validation accuracy of the communication rounds based on different benchmarks shown in Fig. 4. We also provide the communication rounds required to achieve the target accuracy for different benchmarks in Tab. 3. Typically, more communication rounds mean a longer period of time is used in the federated learning training. However, since semi-asynchronous and fully asynchronous federated learning algorithms reduce and eliminate the waiting time to some extent, these algorithms, while time efficient, may require more communication rounds to reach the target accuracy. In Tab. 3, we observe that our proposed *FedFa-Delta* outperforms other existing approaches by up to $3\times$. Combining the wall clock time that is shown in Tab. 2, the *FedFa-Delta* also outperforms the existing approaches for the whole training time.

Dataset	Sent140	STT2	Cifar10	
	Non-IID	Non-IID	Non-IID	IID
Target-Acc	0.71	0.85	0.6	0.7
FedAvg	70(2×)	120(1.6×)	540(1.2×)	320 (1.4×)
Fedasync	failed	failed	failed	failed
FedBuff	70(2×)	123(1.6×)	880(1.9×)	325(1.4×)
Port-Urgent	110(3×)	98(1.3×)	620(1.3×)	305(1.3×)
Port-Non-Urgent	95(2×)	112(1.5×)	565(1.2×)	320(1.4×)
FedFa-Param	45(1.3×)	133(2×)	645(1.4×)	315(1.4×)
FedFa-Delta	**36**	**73**	**465**	**230**

Table 3: Communication Rounds for Target Accuracy



(a) $\alpha = 0.1$



(b) $\alpha = 5$

Figure 4: Performance comparison of different training strategies on both IID and Non-IID data settings regarding communication round.

Best Accuracy. Tab. 4 shows the optimal accuracy of the three foundational paradigms within a fixed number of communication rounds. We observe that *FedFa-Delta* achieves the highest accuracy for both Sent140 and Cifar10-IID, and FedAvg achieves the highest accuracy for Cifar10-NonIID. The accuracy of all three paradigms is similar to each other, while our fully asynchronous approach *FedFa-Delta* is time efficient. Thus, *FedFa* is more suitable for scaling up the training scales in federated learning by eliminating the waiting time (without the barrier setting) compared to the synchronous and semi-asynchronous strategies.

Dataset	Sent140	STT2	Cifar10	
	Non-IID	Non-IID	Non-IID	IID
FedAvg	0.7209	**0.891**	**0.6570**	0.7690
FedBuff	0.7209	0.871	0.6557	0.7776
FedFa-Param	0.7291	0.88	0.6539	0.7781
FedFa-Delta	**0.7403**	0.883	0.6474	**0.7831**

Table 4: Best Accuracy Comparison

Effective of the Buffer Size K. Fig. 5 shows the current optimal model accuracy ladder diagram of *FedFa-Param* with different buffer sizes K. Changing the buffer size K within a reasonable range does not significantly impact the model accuracy, while vastly increasing the K will slow down the training (e.g., $K = 20$). This is reasonable because it will excessively average the outdated information during each aggregation when setting the $K = 20$. These results indicate that our proposed *FedFa* is robust to the buffer size K.
Effective of concurrency M_cThe number of clients sampled at the beginning of the federated learning training is called the number of concurrency M_c. Whenever K clients complete the

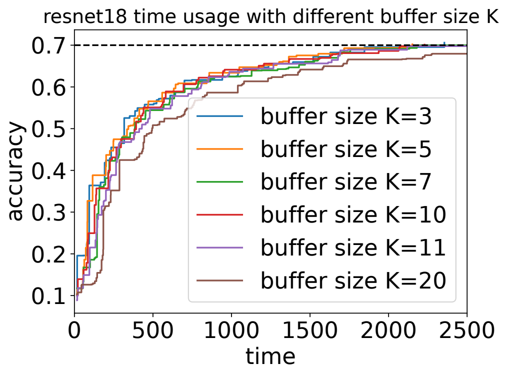


Figure 5: The comparison of different buffer sizes K.

training task and upload the parameters to the server, other K clients are resampled to maintain the number of concurrency M_c. Therefore, it also means the number of simultaneously active clients. In most federated learning experiment setups, M_c is set as $1/10$ to $1/5$ of the total number of clients. In asynchronous algorithms, increasing the number of sampling clients may lead to worse training results because staleness is directly related to the number of clients sampled. As shown in Tab. 5, the results of all the algorithms show some decrease in model accuracy as M_c increases. Moreover, our algorithm is less insensitive to M_c than other algorithms.

M_c	2	5	10	20
FedAvg	0.7101	0.6757	0.6207	0.6017
FedBuff	0.7137	0.6688	0.6327	0.6045
FedFa-Delta	0.718	0.6755	0.642	0.6356

Table 5: The best accuracy of the model which is trained within a defined period of time in different scales of M_c

Combined with Synchronous Optimization Methods. *FedFa* is highly portable and extensible since it only involves a change in the aggregation paradigm. It is easy to extend other synchronous federated learning optimization methods into their versions under the fully asynchronous paradigm of *FedFa*. Moreover, such an extension can also eliminate the waiting time and improve the efficiency of the federated learning training process.

We integrate two synchronous federated learning optimization algorithms, *Fedprox* [Li *et al.*, 2020] and *Fednova* [Wang *et al.*, 2020], into our proposed *FedFa-Delta*, which is a fully asynchronous paradigm. The synchronous federated learning versions of these two algorithms and the version under the full asynchronous paradigm, respectively, are used to perform the experiments on the Cifar10 dataset mentioned before. The number of communication rounds and the number of wall-clock times to achieve the target accuracy using these algorithms are shown in Tab. 6. We observe that the wall-clock time for the whole federated learning training process is reduced to $1/3$ of the original time with only $1/10$ more communication rounds consumption. As shown in Fig. 6, the convergence of the *FedFa* with *Fedprox* is similar to the original algorithm. However, the training process of the *FedFa* with *Fednova* has become unstable. This is reasonable because the training process of asynchronous algorithms is inherently more unstable than synchronous algorithms.

Metrics	Fednova w/o	Fedprox w/o
Comm-Rounds	331 / 390	320 / 290
Wall Time	1290 / 4576	1184 / 3469

Table 6: Comparison of communication rounds and wall clock time for target accuracy between optimization methods with (w/) and without (w/o) extending to the FedFa paradigm.

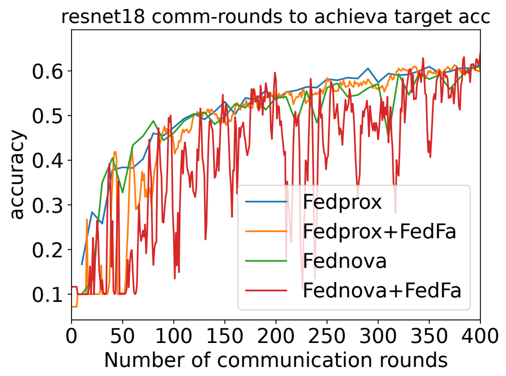


Figure 6: Comparison of the model convergence between Original and FedFa versions of the optimization algorithm

6 Conclusion and Future Work

We propose *FedFa*, a fully asynchronous parameter update strategy for federated learning. Unlike previous fully asynchronous algorithms, it satisfies the paradigm of secure aggregation and guarantees a stable and fast training process. Compared to the synchronous and semi-asynchronous algorithms, it improves the training performance considerably by up to $6\times$ and $4.0\times$ speed up for different benchmarks. Also, unlike other asynchronous algorithms, our approach provides some speedup in the number of communication rounds. We also give theoretical proof of the convergence rate of our proposed *FedFa*, which has the same convergence upper bound as the widely used FedBuff. Finally, we also briefly implement the fully asynchronous version of some synchronous federated optimization algorithms in our *FedFa* paradigm to make them more time efficient. We will explore more parameter update strategies for federated learning in the future.

Acknowledgments

This research was supported by the National Natural Science Foundation of China 62302420 and the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR), under contract DE-AC02-06CH11357. The authors extend their appreciation to the Deanship of Scientific Research, University of Bisha, for funding this research through the promising program under grant number (UB-Promising-40-1445). This work is partially supported by HK RGC General Research Fund (Ref. No.: PolyU-15220922 and PolyU-15228623) and NSFC/RGC Collaborative Research Scheme (Ref. No.: CRS_PolyU501/23).

References

[Bhargava *et al.*, 2021] Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. Generalization in nli: Ways (not) to go beyond simple heuristics, 2021.

[Bonawitz *et al.*, 2016] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[Caldas *et al.*, 2018] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. Leaf: A benchmark for federated settings. *arXiv preprint:1812.01097*, 2018.

[Chai *et al.*, 2020] Zheng Chai, Yujing Chen, Liang Zhao, Yue Cheng, and Huzefa Rangwala. Fedat: A communication-efficient federated learning method with asynchronous tiers under non-iid data. *ArXivorg*, 2020.

[Cho *et al.*, 2022] Yae Jee Cho, Andre Manoel, Gauri Joshi, Robert Sim, and Dimitriadis. Heterogeneous ensemble knowledge transfer for training large models in federated learning. *arXiv preprint arXiv:2204.12703*, 2022.

[Fraboni *et al.*, 2023] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. A general theory for federated optimization with asynchronous and heterogeneous clients updates. *Journal of Machine Learning Research*, 24(110):1–43, 2023.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Hsu *et al.*, 2019] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[Hu *et al.*, 2021] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[K. and Geoffrey, 2009] Alex K. and H. Geoffrey. Learning multiple layers of features from tiny images. 2009.

[Kairouz *et al.*, 2021] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[Karl *et al.*, 2020] Ryan Karl, Jonathan Takeshita, Nirajan Koirla, and Taeho Jung. Cryptonite: A framework for flexible time-series secure aggregation with online fault tolerance. Cryptology ePrint Archive, Paper 2020/1561, 2020.

[Koloskova *et al.*, 2022] Anastasiia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. *Advances in Neural Information Processing Systems*, 35:17202–17215, 2022.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of MLSys*, 2:429–450, 2020.

[Mania *et al.*, 2015] Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. *arXiv preprint arXiv:1507.06970*, 2015.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[Nguyen *et al.*, 2022] John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR, 2022.

[So *et al.*, 2021] Jinhyun So, Başak Güler, and A Salman Avestimehr. Turbo-aggregate: Breaking the quadratic aggregation barrier in secure federated learning. *IEEE Journal on Selected Areas in Information Theory*, 2021.

[Su and Li, 2022] Ningxin Su and Baochun Li. How asynchronous can federated learning be? In *2022 IEEE/ACM 30th International Symposium on Quality of Service (IWQoS)*, pages 1–11. IEEE, 2022.

[U, 2018] Sebastian U. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[Wang *et al.*, 2020] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *NeurIPS*, 2020.

[Zhang and Wang, 2021] Zhaorui Zhang and Choli Wang. Sapus: Self-adaptive parameter update strategy for dnn training on multi-gpu clusters. *IEEE Transactions on Parallel and Distributed Systems*, 33(7):1569–1580, 2021.

[Zhang and Wang, 2022] Zhaorui Zhang and Choli Wang. Mipd: An adaptive gradient sparsification framework for distributed dnns training. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):3053–3066, 2022.

[Zhang *et al.*, 2022] Zhaorui Zhang, Zhuoran Ji, and Choli Wang. Momentum-driven adaptive synchronization model for distributed dnn training on hpc clusters. *Journal of Parallel and Distributed Computing*, 159:65–84, 2022.

[Zhao *et al.*, 2018] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.