Towards Efficient and Stable K-Asynchronous Federated Learning With Unbounded Stale Gradients on Non-IID Data

Zihao Zhou, Yanan Li, Xuebin Ren, *Member, IEEE*, and Shusen Yang, *Senior Member, IEEE*

Abstract—Federated learning (FL) is an emerging privacy-preserving paradigm that enables multiple participants collaboratively to train a global model without uploading raw data. Considering heterogeneous computing and communication capabilities of different participants, asynchronous FL can avoid the stragglers effect in synchronous FL and adapts to scenarios with vast participants. Both staleness and non-IID data in asynchronous FL would reduce the model utility. However, there exists an inherent contradiction between the solutions to the two problems. That is, mitigating the staleness requires to select less but consistent gradients while coping with non-IID data demands more comprehensive gradients. To address the dilemma, this paper proposes a two-stage weighted K asynchronous FL with adaptive learning rate (WKAFL). By selecting consistent gradients and adjusting learning rate adaptively, WKAFL utilizes stale gradients and mitigates the impact of non-IID data, which can achieve multifaceted enhancement in training speed, prediction accuracy and training stability. We also present the convergence analysis for WKAFL under the assumption of unbounded staleness to understand the impact of staleness and non-IID data. Experiments implemented on both benchmark and synthetic FL datasets show that WKAFL has better overall performance compared to existing algorithms.

Index Terms—Federated learning, asynchronous learning, data heterogeneity, prediction accuracy, training stability

1 Introduction

The availability of massive data has been the bottleneck of many machine learning (ML) algorithms, especially in many deep learning (DL) scenarios, such as video surveillance [1], and speech recognition [2]. In the 5G era, more and more data are being generated and scattered at massive users' devices (i.e., clients) locally. The aggregation of these distributed data promises a significant boost in the quality of ML models, prospering various artificial intelligence services. However, with the increasing awareness of data privacy, people are unwilling to share their own data, which would impede the DL development [3] (e.g., especially for data sensitive fields like autonomous driving [4] and disease detection [5]). To address this issue, a novel distributed learning paradigm, federated learning (FL) [6], [7], [8], [9] has been proposed to achieve data utilization from massive clients while not seeing their local data.

Generally, FL [10], [11] enables multiple clients with decentralized data to collaboratively train a shared model under the concerted control of a center server (e.g., service provider), which complies with privacy regulations such as GDPR [12]. By providing an effective framework to exploit the naturally generated data while respecting privacy, FL has not only attracted extensive interests from the academia [9], but also been deployed in various practical applications, such as e-health diagnosis [13], [14], fraud detection [15], and recommendation systems [16], [17].

Many algorithms have been proposed for FL [18], [19], [20], among which, FedAvg is one of the most classical FL algorithms and runs in a synchronous manner [6]. At each iteration, the clients complete several local training processes and upload the model parameter. The server randomly samples a portion of clients and aggregates their locally updated models. In such a case, the runtime of each FL iteration is determined by the stragglers [21], [22] who are the sampled slowest clients. To alleviate the stragglers effect, the server can update the global model once receiving the fastest K gradients out of total clients, namely, K-sync FL [23] or K-batch-sync FL [24]. However, these approaches can not eradicate stragglers due to the unpredictable statuses of sampled clients and network communication.

Besides the above synchronous methods, the K asynchronous FL (K-async FL) has also been extensively studied [25], [26], [27], [28], in which the server updates the global

- *Zihao Zhou and Yanan Li are with the National Engineering Laboratory for Big Data Analytics, School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China. E-mail: {wszzh15139520600, gogll2}@stu.xjtu.edu.cn.*
- *Xuebin Ren is with the National Engineering Laboratory for Big Data Analytics, School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China. E-mail: xuebinren@mail.xjtu.edu.cn.*
- *Shusen Yang is with the National Engineering Laboratory for Big Data Analytics, the Ministry of Education Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China. E-mail: shusenyang@mail.xjtu.edu.cn.*

Manuscript received 13 Nov. 2020; revised 20 Dec. 2021; accepted 25 Jan. 2022. Date of publication 11 Feb. 2022; date of current version 2 June 2022.
This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFA0713900, in part by the National Natural Science Foundation of China under Grants 62172329, 61802298, 61772410, U21A6005, U1811461, and 11690011, and in part by China Postdoctoral Science Foundation under Grants 2020T130513 and 2019M663726.
(Corresponding author: Shusen Yang.)
Recommended for acceptance by J. Wang.
Digital Object Identifier no. 10.1109/TPDS.2022.3150579

model once receiving the gradients from the first K clients and those clients who fail to participate in current iteration can continue their training for reducing the runtime in the next iteration. K-async FL can not only alleviate stragglers, but also save the total training time when the iteration time of clients follows a *new-longer-than-used distribution* [23]. Apart from that, Hannah *et al.* [29] has further pointed out that asynchronous FL (AFL) allows more iterations within the same time compared to synchronous FL. Therefore, compared to K-sync FL, K-async FL has demonstrated great benefits of achieving more efficient learning in highly heterogeneous systems. In this paper, we focus on achieving both efficient and high-utility K-async FL.

Though K-async FL can mitigate the stragglers effect and save total training time [23], there are still two obstacles to be overcome in practice. On the one hand, non-IID datasets generated at different FL clients can impact the model utility [30], [31]. On the other hand, stale gradients may harm the model utility, or even diverge the training process [32], [33]. The solutions to these two problems have been studied separately. For non-IID data, the essence of the existing solutions, e.g., momentum [31], [34] and variance reduction [35], is to fully utilize all available information for estimating the global data distribution. Hence, gradients from as many clients as possible need to be aggregated to make the aggregated gradients represent the overall data comprehensively. For staleness, most studies pointed out that the server should aggregate the received gradients [36], [25] or adjust the learning rate [33], [37] negatively correlated with the staleness. Therefore, only a few low-stale gradients are maintained and most of the high-stale gradients will be filtered out. Apparently, there will pose an essential contradiction when simply combining the existing methods to mitigate the impact of non-IID data and stale gradients. To better help comprehend the contradiction, we explore the interplay of mitigation strategies for non-IIDness and staleness on EMNIST MNIST [38] in Section 4.1. The experimental results show that the mitigation effect for non-IIDness is negatively correlated with the mitigation effect for staleness, which indicates their inherent contradiction. Therefore, it is of significance to design a novel asynchronous FL approach that can effectively deal with both stale gradients and non-IID data.

Though stale gradients may cause the model to diverge, Mitliagkas *et al.* [39] found that stale gradients can help converge faster and they leveraged staleness to accelerate the training process. However, not all gradients can speedup convergence. In fact, low-stale gradients are more likely to have a consistent descent direction and accelerate the training process. On the contrary, high-stale gradients may have different directions and the biased gradients will prevent model convergence. This motivates us to pick the consistent gradients, instead of only utilizing the low-stale gradients. Therefore, based on the potential advantage of consistent gradients and treatments of non-IID data, we propose a two-stage weighted K-async FL algorithm (WKAFL) with adjusting the learning rate to improve the prediction accuracy, training speed and stability. Our main contributions are as follows:

1) We propose WKAFL, a novel K-async FL algorithm with two stages, which improves the model utility of asynchronous algorithm and shows good robustness in non-IID settings. In stage one, stale gradients with consistent descent direction will be picked to accelerate the training process. In stage two, stale gradients with large norm will be clipped to stabilize the model. In both stages, the server adjusts the learning rate according to the least staleness of K gradients and accumulates selected consistent gradients to further mitigate the non-IIDness.
2) We present the convergence analysis for the non-convex optimization problem in AFL to analyze the impact of staleness and non-IID data, under both the assumptions of bounded and unbounded staleness. The analytical results (Theorem 5.1 and Theorem 5.3) validate that both staleness and non-IID data can decrease model utility.
3) We conducted extensive experiments on four federated learning datasets, including CelebA, EMNIST ByClass, EMNIST MNIST and CIFAR10 to validate the performance of WKAFL in terms of training speed, prediction accuracy and training stability. Ablation experiments indicate that each component is indispensable for WKAFL and a contradiction exists between mitigation strategies for staleness and non-IIDness. Comparison experiments show that WKAFL has the best overall performance, especially when the number of clients is large. In particular, WKAFL achieves up to 19.41% accuracy gain on CIFAR10 and 92.8% training stability improvement on EMNIST MNIST compared with existing staleness-aware algorithms. Besides, compared with existing non-IID mitigation algorithm on CIFAR10, WKAFL achieves 5.83-10.59% accuracy gain while maintaining a similar training stability.

The rest of the paper is organized as follows. Section 2 describes the related work on both staleness and non-IID data in AFL. Section 3 presents the system model and problem definition. Section 4 describes the details of our proposed algorithm WKAFL. The corresponding theoretical analysis is presented in Section 5. Section 6 shows the experiment results for validating the performance of WKAFL. Finally, we conclude the paper in Section 7.

2 Related Work

In this section, we introduce the related work of FL in terms of non-IID data and staleness.

2.1 Non-IID Data

Non-IID data is an essential characteristic of FL. However, since most of the well-established statistical theories are based on the IID data assumption, there are only a few analytical results about convergence under the non-IID setting. Some empirical results have shown the negative influence of non-IID data on the model utility. For example, Zhao *et al.* [30] empirically showed that the accuracy of FedAvg decreases significantly with the increase of data heterogeneity. Besides, Li *et al.* [40] proved that non-IID data can decrease the model utility. They assumed that there is a bounded difference between the gradients uploaded by clients and global unbiased gradients to guarantee the convergence, similar to [19],

[41], [42]. In this paper, we follow this assumption and provide a convergence analysis under the non-IID setting. To mitigating the impact of non-IID data, a number of methods have been proposed to utilize all available information to estimate the global knowledge of the decentralized non-IID datasets. For example, in [30], the server broadcasts some non-private data to all the clients to make the gradients to carry more common information. Apart from data sharing, each received gradient can also accumulate historical information by applying momentum [31], [34] or variance reduction [35] to ensure the aggregated gradient to be more representative of the global information. Li *et al.* [31] proposed GSGM which divides the whole training process into multiple rounds. In each round, the server eliminates the diversity of aggregated gradients by using a global momentum, which is accumulated by part of the gradients based on the scheduling strategy.

Since momentum can also help converge faster, WKAFL in this paper also takes advantages of the momentum to alleviate the impact of non-IID data as GSGM. However, WKAFL is essentially different from GSGM. Since GSGM needs all the clients to upload gradients, GSGM has a low scalability due to computational overhead in each round, while WKAFL adapts K-async FL and behaves robustly.

2.2 Staleness

Staleness is one of the conspicuous challenges in AFL. Dai *et al.* [32] found that stale gradients will slow down the training process and decease prediction accuracy. To guarantee model utility, most of existing methods make full use of low-stale gradients and prevent stale gradients to bring down the model utility. One effective path is to adjust the learning rate [33], [43]. In [33], Zhang *et al.* tuned the learning rate on a per-gradient basis inversely proportional to the staleness. However, when there are massive heterogeneous clients, the staleness may be very high, which leads to very small learning rate and extremely slow training. In [43], McMahan *et al.* extended the adaptive gradient methods [44], [45] to the asynchronous parallel setting and revised the learning rate based on previous gradient steps. Another feasible method is to aggregate the gradients with different weights negatively correlated with the staleness. Xie *et al.* [25] and Chen *et al.* [36] aggregated the stale gradients based on exponential weights $(e/2)^{-\tau}$ and e^{τ} respectively, where τ represents the staleness.

To alleviate the effect of staleness, WKAFL adopts the exponential weighting method, which is the same as [36] for the reason of ideal performance in asynchronous scenarios. However, all above algorithms either provide no theoretical analysis, or need the assumption of bounded staleness [46], [47] which has less practicality in many AFL scenarios [48]. In this paper, WKAFL can provide a convergence analysis under the assumption of unbounded staleness.

2.3 Interplay of Non-IID Data and Staleness

To better alleviate the impact of non-IID data, the server requires a greater number of gradients. However, to prevent the stale gradients from bringing down the model utility, only confined low-stale gradients are favorable in the aggregation. Therefore, a contradiction about the quantity of aggregated gradients occurs when simply combining the corresponding mitigation methods concerned about non-IID data and staleness. Facing the same contradiction, the proposed WKAFL aims to exploit the idea of gradient selection to mitigate it. The idea of gradient selection has been explored in [49], which proposes to pick high-quality gradients for tackling with Byzantine attacks. Different from WKAFL, ZENO++ in [49] requires the central server to have some high-quality data and relies on the assumption of IID data, which is impractical in the federated learning scenario. Particularly, with non-IID datasets, it has much poorer performance than WKAFL, as shown in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2022.3150579.

Despite the adverse impact of staleness on model utility, stale gradients can play a positive role in accelerating the training process. In [39], Mitliagkas *et al.* demonstrated that the stale gradients could have the same effect as momentum when the staleness follows a geometric distribution. In fact, consistent stale gradients can boost the convergence, analogous to the function of a large momentum [39], [28]. However, when the model is close to the optimal solution, the momentum may fluctuate the training process. To prevent this fluctuation, WKAFL uses the two-stage strategy as [50] to fully utilize the stale gradients. In [50], the two-stage strategy is used to accelerate training and reduce communication overhead, which is different from the aim of utilization of stale gradients in WKAFL.

3 Preliminary

3.1 Stochastic Optimization

Generally, the aim of a machine learning problem is to minimize the empirical risk function $F(w)$

$$\min_{w} F(w) = \frac{1}{N} \sum_{\xi_i \in \mathcal{D}} f(w, \xi_i),$$

where $\mathcal{D} = \{\xi_1, \xi_2, \ldots, \xi_N\} \subseteq R^n$ is the training dataset and $f(w, \xi_i)$ defines the composite loss function at the ith data point. In mini-batch stochastic gradient descent (SGD), we can minimize the objective function $F(w)$ through iteratively updating the model parameter vector w using

$$w_{j+1} = w_j - \frac{\eta_j}{|\mathcal{D}_j|} \sum_{\xi_i \in \mathcal{D}_j} \nabla f(w_j, \xi_i), \tag{1}$$

where $\mathcal{D}_j \subseteq \mathcal{D}$ and η_j define the mini-batch data and the learning rate at the jth iteration respectively.

3.2 Problem Framework

We consider the K-async FL framework with a central server and P clients. At the beginning, the server initializes the model parameter vector as w_0. Then, all the P clients fetch current parameter vector w_0 and compute their respective gradients on a single mini-batch data independently. Clients who finish computing gradients upload their gradients without waiting for other clients. The server waits for the first K out of P clients and the rest of clients continue computing gradients. As a result, at every iteration, the gradients received by the server might be computed based on stale parameters. Then, for the K-async FL, the updating

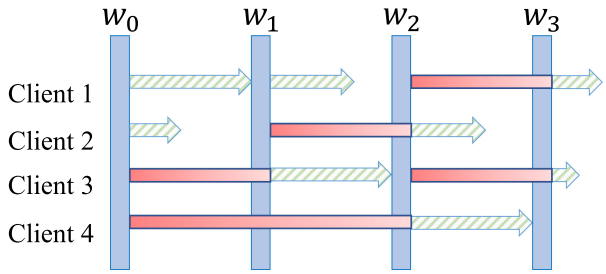


Fig. 1. K-async FL with $K=2$ and $P=4$. The green shadow arrows indicate the gradients selected for updating. Red arrows indicate the gradients which fail to update the global model in current iteration.

formula is expressed by

$$w_{j+1} = w_j - \frac{\eta_j}{K}\sum_{i=1}^{K} g(w_{j,i}, \xi_{j,i}), \tag{2}$$

where $g(w_{j,i}, \xi_{j,i})$ is the stale gradient received by the server at the jth iteration, $\xi_{j,i} = \{\xi_{j,i}^{(1)}, \ldots, \xi_{j,i}^{(m)}\}$ denotes the used m samples of client i at the jth iteration and $w_{j,i}$ defines the model parameter vector used to compute the stale gradient $g(w_{j,i}, \xi_{j,i})$ with staleness $\tau_{j,i}$. Fig. 1 illustrates the two-async FL when the total number of clients is four. At each iteration, four clients independently upload gradients to the server after finishing computing gradients. The server will update the global model once receiving any two gradients (green and shadow arrows), and clients who fail to upload gradients in limited time continue computing gradients based on a local mini-batch (red arrows). After updating the global model, the server broadcasts the new model parameters to the clients who upload their gradients.

Considering the different levels of clients' staleness and the impact of non-IID data, it is unreasonable to use the same weight $1/K$ to aggregate the received gradients in Equation (2). Therefore, in this paper, we will use the weighted K-async FL which is expressed as

$$w_{j+1} = w_j - \eta_j \sum_{i=1}^{K} p_{j,i} g(w_{j,i}, \xi_{j,i}), \tag{3}$$

where $p_{j,i}$ defines the weight of the ith client's gradient at the jth iteration and $\sum_i^K p_{j,i} = 1$. If $p_{j,i} = 1/K$, Equation (3) is back to Equation (2). Our main purpose is showing how to determine $p_{j,i}$ based on the staleness and non-IID data to improve both the model utility.

3.3 Definitions and Assumptions

To determine $p_{j,i}$, we first define what kind of stale gradients $g(w_{j,i}, \xi_{j,i}), i = 1, \ldots, K$ are "consistent" or "inconsistent", based on the cosine similarity comparison between $g(w_{j,i}, \xi_{j,i})$ and the globally unbiased gradient $\nabla F(w_j)$.

Definition 3.1 (Inconsistent gradient). *If the cosine similarity of a gradient and the globally unbiased gradient is smaller than a given constant $sim_{min} \in [0, 1]$, then the gradient is an inconsistent gradient.*

Definition 3.2 (Consistent gradient). *If the cosine similarity of a gradient and the globally unbiased gradient is larger than a given constant $sim_{min} \in [0, 1]$, then the gradient is a consistent gradient.*

TABLE 1
Notations

Variables	Meaning
η_j	Learning rate at the jth iteration
P	Number of total clients
K	Number of gradients needed at each iteration
m	Mini-batch size
L	Lipschitz constant
J	Total number of iteration
w^*	Global optimal solution
w_j	Global model at the jth iteration
τ_{max}	The maximal staleness of all the gradients in stage one
$\tau_{j,i}$	The staleness of the ith gradients at jth iteration
$p_{j,i}$	Weight of the ith gradient at jth iteration

Besides, some basic assumptions for convergence analysis are listed as follows.

Assumption 3.1 (Lipschitz Continuity). *Objective function $F(w)$ satisfies L-Lipschitz continuity: $\forall w_1, w_2, \exists$ constant L*

$$F(w_1) - F(w_2) \leq \nabla F(w_1)^T (w_2 - w_1) + \frac{L}{2}||w_2 - w_1||_2^2.$$

Assumption 3.2 (Client-Level Unbiased Gradient). *The gradient $g(w_j, \xi_{j,i})$ of client i is a client-level unbiased gradient which means that the expectation of gradient $g(w_j, \xi_{j,i})$ is equal to $\nabla F_i(w_j)$:*

$$E[g(w_j, \xi_{j,i})] = \nabla F_i(w_j).$$

Assumption 3.3 (Gradients with Bounded Variance). *The gradient $g(w_j, \xi_{j,i})$ of client i has client-level bounded variance: $\exists$ constants σ_c, M_c*

$$E[||g(w_j, \xi_{j,i}) - \nabla F_i(w_j)||_2^2] \leq \frac{\sigma_c^2}{m} + \frac{M_c}{m}||\nabla F_i(w_j)||_2^2.$$

where $\nabla F_i(w_j)$ is the unbiased gradient of client i. To guarantee the convergence of the model, we also need to assume $\nabla F_i(w_j)$ satisfies global-level bounded variance: $\exists$ constant G

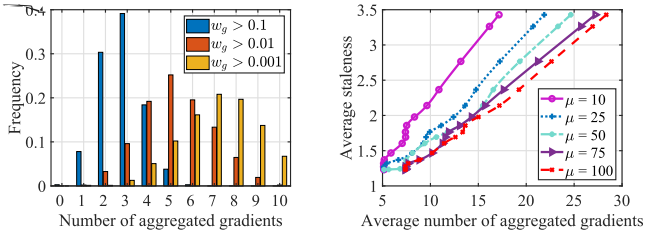
$$||\nabla F(w_j) - \nabla F_i(w_j)||_2^2 \leq G^2,$$

which means that the difference between the unbiased gradient of each client and the unbiased gradient of global data is limited.

Notations. In Table 1, we list some notations involved in the paper.

4 Weighted K-Async FL With Learning Rate Adaptation

In this section, we introduce WKAFL to alleviate the contradiction between the existing strategies for staleness and non-IIDness, thus improving the model utility. We first motivate our work by showing the experimental results about this contradiction in Section 4.1. In Section 4.2, we briefly introduce the basic idea. Section 4.3 then presents the algorithm details of WKAFL.



(a) K=20. (b) K=50.

Fig. 2. Motivation experiments. (a) Exploration of the effect of non-IIDness; (b) Relation of non-IID data (number of aggregated gradients) and staleness (average staleness).

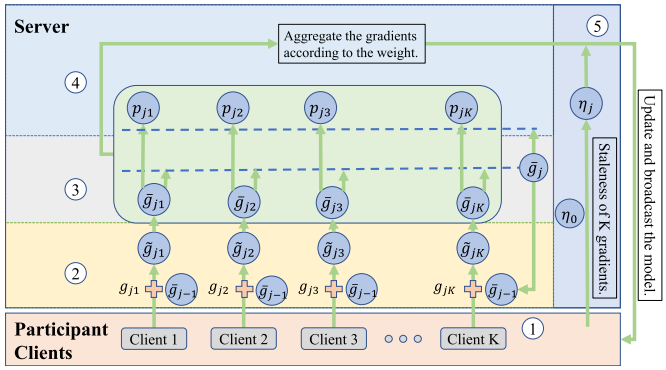


Fig. 3. WKAFL with five main components, one for clients (①) and four for the server (②- ⑤). Gradients computation (①). Improvement of stability on non-IID data (②). Estimation of globally unbiased gradient (③). Selection and aggregation of consistent gradients (④). Learning rate adaptation (⑤).

4.1 Motivation

To demonstrate the aforementioned contradiction, we implemented the K-async FL algorithm that simply integrates the mitigating strategies for staleness and non-IIDness, and trained it on EMNIST MNIST [38]. Specifically, the mitigation strategy for non-IIDness here is to accumulate historical gradients [31], while that for staleness is to assign exponential weights to the gradients in aggregation [36]. We use the average staleness and average number of aggregated gradients to represent the impact of staleness and non-IIDness, respectively. The detailed descriptions are as follows.

- Average staleness: the mean value of weighted staleness at each iteration and is defined as

$$\tau_{ave} = \frac{1}{J}\sum_{j=1}^{J}\sum_{i=1}^{K} p_{j,i}\tau_{j,i},$$

  where $p_{j,i}$ and $\tau_{j,i}$ denote the weight and staleness of the gradient. Large τ_{ave} reflects the poor mitigation effect for staleness.
- Average number of aggregated gradients: the number of gradients whose weights are larger than a threshold and is defined as

$$N_{ave} = \frac{1}{J}\sum_{j=1}^{J}\Big|\{p_{j,i}, i \in [K] | p_{j,i} \geq p_{j,max}/\mu\}\Big|,$$

  where $p_{j,max} = \max\{p_{j,i}, i \in [K]\}$ and $\mu \in \mathbb{R}^+$ is a constant. Similarly, large N_{ave} reflects the poor mitigation effect for non-IIDness.

Fig. 2a depicts the histogram over the number of predominated gradients in all aggregations during training. The predominated gradients refer to the gradients whose weights are larger than a threshold $w_g \in \mathbb{R}^+$. The more predominated gradients means the aggregated gradients are more representative, which enhances the mitigation for non-IIDness. However, in almost all iterations, there are at most five gradients with weights larger than 0.1. This reflects that only a small number of gradients dominate the aggregation, incurring an adverse effect on non-IIDness mitigation. Fig. 2b shows that the average staleness grows approximately linearly with the number of aggregated gradients, indicating a negative correlation between the two. In conclusion, there exists a contradiction between the mitigation strategies for non-IIDness and staleness.

4.2 Basic Idea of WKAFL

To break the above contradiction, we propose WKAFL to pick consistent gradients to alleviate the impact of non-IIDness while accelerating the training process.

Generally, gradients with low-stale are relatively reliable and high-stale gradients are more likely to be biased. However, some high-stale gradients may also have consistent descent direction. Then, we can estimate the unbiased gradients based on low-stale gradients and then pick the consistent high-stale gradients. If these gradients are selected, more gradients are aggregated to improve the mitigation effect of non-IID data while guaranteeing the stale gradients not to diverge the model. In such a case, the contradictory of mitigating the effect of non-IID data and staleness will be mitigated. Apart from this advantage, the selected gradients can also help converge faster. Since stale model is possibly farther from optimal solution and its derivative has a larger norm, stale gradients generally have larger norm for convex loss function. Therefore, the selected consistent stale gradients can help converge faster [39], [28].

4.3 Detailed Descriptions

Based on above basic idea, we proposed WKAFL to accelerate the training process and improve the training stability. Pseudo-code of WKAFL consists of two parts, Algorithm 1 for clients and Algorithm 2 for the server. The workflow of the whole process for WKAFL is shown in Fig. 3, which includes gradients computation (①) for clients and four main parts for the server, improvement of stability on non-IID data (②), estimation of globally unbiased gradient (③), selection and aggregation of consistent gradients (④) and learning rate adaptation (⑤). Detailed description of the four parts can refer to Section 4.3.1 - Section 4.3.4 respectively.

For clients (①), as shown in Algorithm 1, they first download the current model parameters and the iteration index from the server (Line 1). Then they compute gradients and upload the results to the server (Lines 2-4).

For the server (②–⑤), as shown in Algorithm 2, after initializing the model (Lines 1-3), the server broadcasts it to all clients for local computation (Line 4). The learning rate adaptation strategy and weighted strategy are presented in Lines

6-16. Particularly, the weighted strategy includes three components (Lines 8, 10-15), alleviating the impact of non-IID data (Line 8), estimating globally unbiased gradient (Lines 13-14), selecting and aggregating consistent gradients (Lines 15) while the learning rate adaptation strategy consists of one component, adjustment of learning rate (Line 16). We will describe the four components of Algorithm 2 and explain corresponding rationality in Sections 4.3.1 to 4.3.4.

Algorithm 1. WKAFL: Clients Side

Input: mini-batch size m.
Output: loss $l_{j,i}$, gradient $g(w_{j,i}, \xi_{j,i})$, delay $\tau(j)$.
1: Receive model parameters w_j and iteration j from the server.
2: Compute loss $l_{j,i}$ and gradient $g(w_{j,i}, \xi_{j,i})$ based on m samples $\xi_{j,i}$.
3: Set $\tau(j) \leftarrow j$.
4: Upload $l_{j,i}$, $g(w_{j,i}, \xi_{j,i})$ and $\tau(j)$.

Algorithm 2. WKAFL: Server Side

Input: learning rate η_0, number of gradients received by server K, gradients adjustment parameter B, learning rate adjustment parameter γ, clip-bound parameter CB, weighted parameter β, constant α.
Output: Optimal solution w^*.
1: Initialize model parameter w_0 and iteration $j = 1$.
2: Initialize the estimated gradient $\bar{g}(w_0) = 0$.
3: Initialize the stage state $stage = 1$;
4: Broadcast w_0 and j to all clients.
5: **while** the stopping criteria is not satisfied **do**
6: **for** $i = 1 \rightarrow K$ **do**
7: Receive loss value, the gradient and its staleness $(l_i, g(w_{j,i}, \xi_{j,i}), \tau_i)$ from the ith client.
8: $\tilde{g}(w_{j,i}, \xi_{j,i}) = g(w_{j,i}, \xi_{j,i}) + \alpha \bar{g}(w_{j-1})$;
9: **end for**
10: **if** $\sum_{i=1}^{K} l_i \leq \epsilon$ **then**
11: $stage = 2$;
12: **end if**
13: Clip all the gradients with bound CB.
14: Estimate globally unbiased gradient $\bar{g}(w_j)$ according to Equation (5).
15: $g(w_j) = SAGrad(B, \beta, \epsilon, \bar{g}(w_{j,i}, \xi_{j,i}), \bar{g}(w_j), stage)$;
16: Adjust the learning rate η_j adaptively according to Equation (10).
17: $w_{j+1} \leftarrow w_j - \eta_j g(w_j)$.
18: $j \leftarrow j + 1$.
19: **end while**

4.3.1 Improvement of Stability on Non-IID Data

It is efficient to decrease the effect of non-IID data by accumulating all the historical gradients. Then, the estimated gradient at $(j-1)$th iteration can be added to each of K gradients at jth iteration to decrease the impact of non-IID data

$$\tilde{g}(w_{j,i}, \xi_{j,i}) = g(w_{j,i}, \xi_{j,i}) + \alpha \bar{g}(w_{j-1}), \tag{4}$$

where $\alpha > 0$ is a constant and $\bar{g}(w_{j-1})$ is the estimated gradient at $(j-1)$th iteration.

Now, we explain the rationality from theoretical perspective. Every gradient received by the server is biased because of staleness and non-IID data. Then, we can decompose the gradient $g(w_{j,i}, \xi_{j,i})$ into three parts

$$E[g(w_{j,i}, \xi_{j,i})] = \nabla F(w_j) + E[g_s(w_{j,i}, \xi_{j,i})] + E[g_n(w_{j,i}, \xi_{j,i})],$$

where $\nabla F(w_j)$ is the unbiased gradient based on global data at jth iteration, $E[g_s(w_{j,i}, \xi_{j,i})]$ is the error resulted from staleness, and $E[g_n(w_{j,i}, \xi_{j,i})]$ is the error resulted from non-IID data.

Similarly, the estimated gradient can also be decomposed into three parts

$$E[\bar{g}(w_j)] = \nabla F(w_j) + E[\bar{g}_s(w_{j,i}, \xi_{j,i})] + E[\bar{g}_n(w_{j,i}, \xi_{j,i})].$$

Then, the expectation of $\tilde{g}(w_{j,i}, \xi_{j,i})$ is

$$\begin{aligned} E[\tilde{g}(w_{j,i}, \xi_{j,i})] &= \nabla F(w_j) + \alpha \nabla F(w_{j-1}) \\ &+ \underbrace{E[g_s(w_{j,i}, \xi_{j,i})] + \alpha E[\bar{g}_s(w_{j-1,i}, \xi_{j-1,i})]}_{E_S} \\ &+ \underbrace{E[g_n(w_{j,i}, \xi_{j,i})] + \alpha E[\bar{g}_n(w_{j-1,i}, \xi_{j-1,i})]}_{E_N} \end{aligned}$$

where E_N is resulted from non-IID data. $E[g_n(w_{j,i}, \xi_{j,i})]$ only contains the information of client i. While E_N contains the information of more clients and the information can accumulate as the iterative number increases. Therefore, Equation (4) can decrease the effect of non-IID data and stabilize the model.

4.3.2 Estimation of Globally Unbiased Gradient

It is assumed that gradients with lower staleness have high probability to be consistent and gradients with higher staleness have high probability to be inconsistent because gradients based on stale model generally deviate from consistent direction. Therefore, we can aggregate gradients by weighting them according to their staleness to estimate globally unbiased gradient. The rule of aggregation is

$$\bar{g}(w_j) = \sum_{i=1}^{K} \frac{a_{j,i}}{a} \bar{g}(w_{j,i}, \xi_{j,i}), \tag{5}$$

where $\bar{g}(w_{j,i}, \xi_{j,i})$ is a clipped gradient of $\tilde{g}(w_{j,i}, \xi_{j,i})$, $a = \sum_{i=1}^{K} a_{j,i}$ and $a_{j,i}$ is a function of staleness. Cong *et al.* [25] empirically figured out that a better utility was achieved when $a_{j,i}$ was chosen as an exponential function. Chen *et al.* [36] found that $a_{j,i}$ was more properly set as $(\frac{e}{2})^{-\tau_{j,i}}$ than $e^{-\tau_{j,i}}$, where $\tau_{j,i}$ is the staleness of gradient $g(w_{j,i}, \xi_{j,i})$ and e is natural logarithm. Based on their work, we set $a_{j,i}$ to $(\frac{e}{2})^{-\tau_{j,i}}$ in our experiments.

Before estimating the globally unbiased gradient, it is necessary to clip the gradients for two reasons. On the one hand, it is known that clipping can prevent gradient explosion. On the other hand, under the assumption that gradients with lower (higher) staleness have high probability to be consistent (inconsistent), the aggregated gradient after clipping will be closer to the globally unbiased gradient. A straightforward illustration is shown in Fig. 4. When aggregating the gradients before clipping, the norm of stale gradients may be so large that the aggregated gradient deviates from the globally unbiased gradient. When aggregating the gradients after clipping, the aggregated gradient is closer to

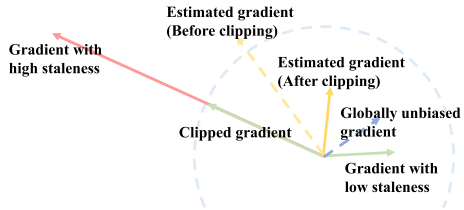


Fig. 4. Estimated gradient before and after clipping the gradients respectively.

the globally unbiased gradient due to the alleviation of impact of gradients with higher staleness.

4.3.3 Selection and Aggregation of Consistent Gradients

Some stale gradients may have consistent update direction. Then, we can pick these gradients to accelerate the training process. When the staleness is low, the estimated gradient is generally consistent with globally unbiased gradient. Therefore, when judge whether a gradient is consistent or inconsistent, we treat the estimated gradient as globally unbiased gradient for convenience. To determine which gradients are consistent and the corresponding degrees, the server will first compute the cosine similarity $sim_{j,i}$ of the estimated globally unbiased gradient $\bar{g}(w_j)$ and corresponding gradient $\bar{g}(w_{j,i}, \xi_{j,i})$. Then the server assigns each gradient with weight according to the cosine similarity. The rule for assigning weight is given by

$$sim_{j,i} = \cos\left\langle \bar{g}(w_{j,i}, \xi_{j,i}), \bar{g}(w_j) \right\rangle; \tag{6}$$

$$p'_{j,i} = \begin{cases} \exp(\beta sim_{j,i}), & \text{if} \quad sim_{j,i} \geq sim_{min}\,; \\ 0, & \text{if} \quad sim_{j,i} < sim_{min}. \end{cases} \tag{7}$$

$$p_{j,i} = p'_{j,i} / sum_{i=1}^{K} p'_{j,i}; \tag{8}$$

where β is a constant, $p_{j,i}$ defines the weight of gradient $g(w_{j,i}, \xi_{j,i})$, and sim_{min} is a given threshold used to judge whether a gradient is consistent or not. Gradients whose cosine similarity is lower than the threshold will be judged as inconsistent gradients and abandoned. On the contrary, consistent gradients will be aggregated according to their weights. The aggregation rule is as follow:

$$g(w_j) = \sum_{i=1}^{K} p_{j,i} \bar{g}(w_{j,i}, \xi_{j,i}).$$

However, when the model is going to converge, the norm of stale gradients may be too large to make the model fluctuate around the optimal solution. To deal with this problem, we divide the whole training process into two stages. In stage one, the training process can be accelerated by taking advantage of stale gradients. When the average loss of K participant clients is smaller than a constant threshold ϵ , the training process enters into stage two. Though the norm of gradient also reflects the state of global model, it is improper to be selected as the stage division criterion because the global model may enter the local optimization area where the norm of gradient is approximate to zero but loss value is not the minimum. In stage two, to guarantee the training stability, consistent stale gradients will be clipped before aggregating based on the norm of estimated gradients. The clipping rule is given by

$$\bar{g}(w_{j,i}, \xi_{j,i}) = \begin{cases} \frac{B||\bar{g}(w_j)||_2}{||\bar{g}(w_{j,i}, \xi_{j,i})||_2} \bar{g}(w_{j,i}, \xi_{j,i}), \\ \text{if} \quad B||\bar{g}(w_j)||_2 \leq ||\bar{g}(w_{j,i}, \xi_{j,i})||_2\,; \\ \bar{g}(w_{j,i}, \xi_{j,i}), \\ \text{if} B||\bar{g}(w_j)||_2 > ||\bar{g}(w_{j,i}, \xi_{j,i})||_2. \end{cases} \tag{9}$$

where the gradients adjustment parameter $B \in (0, \infty)$ is a constant.

Algorithm 3. Select and Aggregate Consistent Gradients

Input: B, β, ϵ, $\bar{g}(w_{j,i}, \xi_{j,i})$, $\bar{g}(w_j)$, $stage$.
Output: $g(w_j)$.
1: **function** SAGrad$B$, β, ϵ, $\bar{g}(w_{j,i}, \xi_{j,i})$, $\bar{g}(w_j)$, $stage$
2: **for** $i = 1, \ldots, K$ **do**
3: $sim_{j,i} = \cos\left\langle \bar{g}(w_{j,i}, \xi_{j,i}), \bar{g}(w_j) \right\rangle$;
4: **if** $sim_{j,i} \geq sim_{min}$ **then**
5: $p'_{j,i} = \exp(\beta * sim_{j,i})$;
6: **else**
7: $p'_{j,i} = 0$;
8: **end if**
9: **end for**
10: **if** $stage = 2$ **then**
11: **if** $B||\bar{g}(w_j)||_2 \leq ||\bar{g}(w_{j,i}, \xi_{j,i})||_2$ **then**
12: $\bar{g}(w_{j,i}, \xi_{j,i}) = \frac{B||\bar{g}(w_j)||_2}{||\bar{g}(w_{j,i}, \xi_{j,i})||_2} \bar{g}(w_{j,i}, \xi_{j,i})$;
13: **end if**
14: **end if**
15: $p_{j,i} = p'_{j,i} / sum_{i=1}^{K} p'_{j,i}$;
16: $g(w_j) = \sum_{i=1}^{K} p_{j,i} \bar{g}(w_{j,i}, \xi_{j,i})$.
17: **return** $g(w_j)$
18: **end Function**

The workflow can refer to Algorithm 3. The server first computes the cosine similarity between estimated gradient and every gradient received by the server (Line 3). Then the server assigns gradient $g(w_{j,i}, \xi_{j,i})$ with a weight $p_{j,i}$ according to cosine similarity (Lines 4-8) and aggregates the gradients (Lines 15-16) eventually. If the model enters stage two, stale gradients will be clipped (Lines 10-14).

4.3.4 Learning Rate Adaptation

In WKAFL, the estimated gradient may deviate seriously from the globally unbiased gradient when most of the K gradients have high staleness, leading to a seriously biased estimated gradient and a poor improvement of model utility. To alleviate it, a staleness-based adaptive learning rate is more reasonable than using a constant learning rate. Intuitively, a small learning rate is required for alleviating the impact of high staleness. In [33], Zhang *et al*. divided the initial learning rate by the staleness to adjust the learning rate. However, it can not be extended to large scale AFL due to the possibly high staleness. In such a case, the adjusted learning rate will be so small that the training process will be prolonged due to the very slight improvement at each iteration. To address it, we adjust the learning rate according to the minimal staleness of the K gradients because the estimated gradient is mainly determined by gradients with low staleness, which is expressed as follows:

$$\eta_j = \eta_0 * \frac{1}{\tau_{min,j} * \gamma + 1}, \tag{10}$$

where η_0 is the initial learning rate, $\tau_{min,j}$ is the minimal staleness of K gradients at jth iteration, and $\gamma \in (0, 1)$ is a constant. Note that $\eta_j \leq \eta_0$ and the equality holds only if $\tau_{min,j} = 0$ (i.e., no staleness).

5 Analysis

In Section 4, we have explained the rationality of weighting and adjustment for learning rate to improve the prediction accuracy and stabilize the training process. In this section, we further analyze the convergence rate of the proposed WKAFL in which the loss function is non-convex, considering both non-IID data and unbounded staleness. Theorems 5.1 and 5.3 show the convergence analysis results with respect to two stages in WKAFL respectively. *In stage one, the staleness is bounded while in stage two, the staleness is unbounded.* Most convergence analysis of AFL algorithm is based on the theory of stochastic optimization and one challenge is how to bridge the stale gradient and globally unbiased gradient. In stage one, the aggregated gradient at jth iteration is $g(w_j) = \sum_{l=1}^{j} \alpha^{j-l} \sum_{i=1}^{K} p_{l,i} g(w_{l,i}, \xi_{l,i})$. Consequently, the difference between global unbiased gradient $\nabla F(w_j)$ and the aggregated gradient is the sum of the difference between each component gradient $g(w_{l,i}, \xi_{l,i})$ and the unbiased gradient. We connect the local gradient $g(w_{l,i}, \xi_{l,i})$ with the theoretical unbiased gradient $\nabla F(w_j)$ by two steps.

- Step 1: We connect the local gradient $g(w_{l,i}, \xi_{l,i})$ with globally unbiased gradient $\nabla F(w_{l-\tau_{l,i}})$ by taking expectation of $g(w_{l,i}, \xi_{l,i})$ and bounding the locally unbiased gradient $\nabla F_i(w_{l-\tau_{l,i}})$ with $\nabla F(w_{l-\tau_{l,i}})$ based on Assumption 3.3.
- Step 2: We connect the unbiased gradient $\nabla F(w_{l-\tau_{l,i}})$ with the unbiased gradient $\nabla F(w_j)$. The difference can be bounded by accumulating all aggregated gradients from $\max(l - \tau_{max}, 1)$ to $j - 1$ iterations.

However, in stage two, the staleness can be infinite in theory due to the possibly infinite number of total iterations. Therefore, the method for dealing with finite staleness will not hold in stage two. To deal with it, we use three steps to connect the local gradient $g(w_{l,i}, \xi_{l,i})$ of lth iteration contained in the momentum term with $\nabla F(w_j)$.

- First, we connect the local gradient $g(w_{l,i}, \xi_{l,i})$ with estimated gradient $\bar{g}(w_j)$ by clipping $g(w_{l,i}, \xi_{l,i})$ and selecting consistent gradients. Based on Equation (9), we have $||\bar{g}(w_j, \xi_{j,i})||_2^2 \leq B^2 ||\bar{g}(w_j)||_2^2$. Apart from that, since the accumulated gradients are consistent with the estimated gradients $\bar{g}(w_l)$ and inconsistent gradients will be endowed with weight 0, then the weighted differences between clipped gradients at lth iteration and the estimated gradients can be bounded by $||\bar{g}(w_l)||_2^2$, i.e., $\exists \sigma_e^2, M_e$

$$\sum_{i=1}^{K} p_{l,i} ||\bar{g}(w_l) - \bar{g}(w_{l,i}, \xi_{l,i})||_2^2 \leq \sigma_e^2 + M_e ||\bar{g}(w_l)||_2^2. \tag{11}$$

- Second, after alleviating the effect of non-IID data and staleness, for the reason that the estimated gradient $\bar{g}(w_l)$ is closed to the unbiased gradients, we can assume

$$||\bar{g}(w_l) - \nabla F(w_l)||_2^2 \leq \sigma_c^2, \quad \exists \sigma_c^2, \tag{12}$$

  which bridges the connection between $\bar{g}(w_l)$ and $\nabla F(w_l)$.
- Third, the differences between $\nabla F(w_l)$ and $\nabla F(w_j)$ can be bounded by the aggregated gradients from l to j iterations.

After providing the proof sketch for the two stages, we will present the analysis results of WKAFL. The convergence result of stage one is as follows.

Theorem 5.1 (Stage One). *Assume Assumption 3.1, 3.2, 3.3 hold. To guarantee the convergence, learning rate η_j satisfies that $\eta_j \leq \frac{1}{2L(1+\frac{M_G}{m})}$ and $L^2 \sum_{t=l}^{J} (J - \tau_t) \alpha^{-t} \sum_{k=l}^{t} \eta_k^2 s_k \alpha^k (1 + \frac{M_c}{m}) < \frac{1}{4}, l \in [J]$. Then, we have the following convergence result*

$$\frac{1}{J} \sum_{j=1}^{J} \frac{\eta_j s_j}{2} \mathbb{E}[||\nabla F(w_j)||_2^2] \leq \frac{1}{J} \sum_{j=1}^{J} (A_j \eta_j^2 + \eta_j s_j G^2 + \eta_j \sum_{l=1}^{j} \alpha^{j-l} (j - \tau_l) \sum_{t=\tau_l}^{j-1} \eta_t^2 B_j) + \frac{F(w_1) - F(w^*)}{J}, \tag{13}$$

where $s_j = \sum_{k=1}^{j} \alpha^{j-k}$, $\tau_l = \max(1, l - \tau_{max})$, $A_j = \frac{L\sigma_c^2}{2m} s_j \sum_{l=1}^{j} \alpha^{j-l} \mathbb{E}[\sum_{i=1}^{K} p_{l,i}^2]$ and $B_j = L^2 s_t \sum_{k=1}^{t} \alpha^{t-k} \frac{\sigma_c^2}{m} \mathbb{E}[\sum_{i=1}^{K} p_{k,i}^2]$.

Proof. See Appendix B, available in the online supplemental material. □

Based on Theorem 5.1, we can qualitatively analyze the impact of staleness and different levels of non-IID data. *With respect to staleness*, it is observed that the maximal staleness τ_{max} has a negative impact on convergence rate. In AFL, the staleness is increasing with the increase of P/K, the ratio of number of total clients to the participants. Therefore, for large scale AFL system, it is critical to decrease P/K to ensure a high model utility. Alternatively, we demonstrate how to further improve the model utility under the given P/K by exploiting the stale gradients. *With respect to non-IID data* measured by the parameter G^2, it is observed that the convergence rate is decreasing with G^2. That is, the non-IID data has a negative impact on the model utility. However, the convergence result is too complex to acquire the convergence rate. Therefore, we consider the constant learning rate to simplify Equation (13).

Remark 5.2. If we set the weight of each received gradients to $1/K$ and set the learning rate to a constant η in Theorem 5.1, the right part of the Equation (13) is $\mathcal{O}(\frac{\eta^2}{K} + \eta G^2 + \frac{\eta^3 (1 + J^2 \alpha^J)}{K} + \frac{1}{J})$. If we further set the learning rate to $\eta = \frac{\sqrt{K}}{\sqrt{J}}$, then for any large enough J that satisfies $J \geq 4KL^2 (1 + \frac{M_G}{m})^2$ and $L^2 K \sum_{t=1}^{J} (J - \tau_t) \alpha^{-t} \sum_{k=l}^{t} s_k \alpha^k (1 + \frac{M_c}{m}) < \frac{J}{4}$, the convergence rate is $\mathcal{O}(\frac{1}{\sqrt{KJ}} + \frac{1}{J} + G^2)$ where G measures the level of non-IID data. We can notice that the the non-IID data will cause the model to fluctuate near the optimal solution. In IID settings, we have $G = 0$ and the convergence rate is $\mathcal{O}(\frac{1}{\sqrt{KJ}} + \frac{1}{J})$ which indicates that WKAFL achieves a linear speedup.

Next, we will present the convergence result for stage two. Since the staleness can be unbounded and the gradients are clipped which is different from stage one, two additional assumptions are required and the rationality has been explained in the proof sketch.

Theorem 5.3 (Stage Two). *Assume Assumption 3.1, 3.2, 3.3 hold. To guarantee the convergence, we also assume Equation (11) and (12) holds. Learning rate satisfies that* $\eta_j \leq \frac{1}{Ls_j}$ *and subjects to* $V_j \geq 0$, *where* V_j *is*

$$V_j = \frac{\eta_j s_j}{2} - \sum_{l=j}^{J} 3M_e \eta_l \alpha^{l-j} - \sum_{l=j}^{J} \eta_l \sum_{t=j}^{l-1} \eta_t^2 \sum_{k=1}^{t} \alpha^{t-j} I_{l,k,t}.$$

Then, we can obtain the following convergence result

$$\frac{1}{J}\sum_{j=1}^{J} V_j ||\nabla F(w_j)||_2^2 \leq \frac{1}{J}\sum_{j=1}^{J} \left(\frac{3\eta_j s_j}{2} C + \eta_j \sigma_c^2 \sum_{l=1}^{j} \sum_{t=l}^{j-1} \eta_t^2 s_t I_{j,l,t} \right) + \frac{F(w_1) - F(w^*)}{J}, \quad (14)$$

where $C = \sigma_c^2 + 2M_e\sigma_c^2 + \sigma_e^2$ *and* $I_{l,k,t} = 3L^2B^2 s_t \alpha^{l-k}(l-k)$.

Proof. See Appendix C, available in the online supplemental material. □

Parameters σ_c^2 represents the difference between the estimated gradient and the globally unbiased gradient. When σ_c^2 is small, the estimated gradient is closer to the globally unbiased gradient. The right part of Equation (14) becomes small and the model will converge faster. Similarly, as B decreasing, the left part of Equation (14) becomes large while the right part becomes small. Then, the model will also converge faster. In summary, both σ_c^2 and B have negative impacts on the convergence rate. The same as stage one, we will also analyze the settings with a constant learning rate for stage two to present the convergence rate.

Remark 5.4. If we set the learning rate to a constant η in Theorem 5.3, the right part of Equation (14) is $\mathcal{O}(\eta C + \eta^3 + \frac{1}{J})$. If we further set the learning rate to $\eta = \frac{1}{\sqrt{J}}$, then for any $J \geq \frac{L^2}{(1-\alpha)^2} \geq L^2 S_J^2$ and large enough J to make $\frac{1}{J}\sum_{l=j}^{J}\sum_{t=j}^{l-1}\sum_{k=1}^{t}\alpha^{t-j} I_{l,k,t}$, we have $\mathcal{O}(\frac{1}{\sqrt{J}} + C)$. Since the term $\sum_{l=j}^{J}\sum_{t=j}^{l-1}\sum_{k=1}^{t}\alpha^{t-j} I_{l,k,t}$ contains $\alpha < 1$, then the term is $\mathcal{O}(\alpha^J J^y), y \in \mathbb{R}$. Therefore, there exists large enough J to meet the conditions. The parameter C reflects the level of non-IID data and the convergence rate demonstrates that non-IID data will decrease the prediction accuracy. In IID settings, we have $C = 0$ and the convergence rate will be $\mathcal{O}(\frac{1}{\sqrt{J}})$.

6 Experiment

In this section, we conducted extensive experiments[1] under the PySyft framework [51] to validate the performance of proposed algorithm WKAFL in terms of the training speed, prediction accuracy and training stability. The proposed WKAFL was compared with three existing algorithms, which will be described in Section 6.1. Meanwhile, several benchmark datasets were adopted with varying levels of non-IIDness. In Section 6.2, ablation experiments demonstrate the impacts of each component of WKAFL. In Section 6.3, detailed comparison results are illustrated to validate the advantage of WKAFL.

6.1 Experiment Setup

6.1.1 Datasets and Models

In our experiments, we used four datasets: CelebA [52], EMNIST ByClass [38], EMNIST MNIST [38] and CIFAR10 [53]. The first two benchmark FL datasets were directly used. EMNIST MNIST and CIFAR10 were manually split to have different levels of non-IIDness.

- CelebA partitions the Large-scale CelebFaces Attributes Dataset by the celebrity in the image.
- EMNIST ByClass contains 81,4255 character images of 62 unbalanced classes. It is partitioned over around 3,500 clients according to the writers in FL.
- EMNIST MNIST has 70,000 characters in ten balanced labels. We partitioned it into 10 subsets by the label. Then, the digit images were divided over clients, which sampled local data according to Algorithm 4. Every client has L_{num} classes of labels and the total number of data ranges from D_{min} to D_{max}.
- CIFAR10 is a labeled dataset of 60,000 tiny color images in 10 classes, and consists of 50,000 training images and 10,000 test images. We partitioned CIFAR10 into FL according to Algorithm 4 similarly.

Algorithm 4. Generator of Non-Iid Datasets

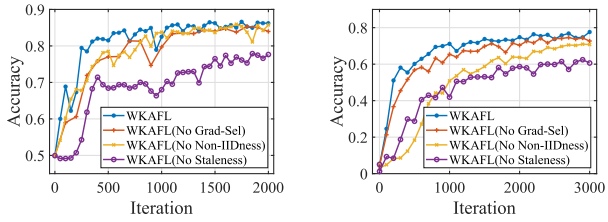
Input: D_{min}, D_{max}, L_{num}, P.
Output: $Datas = (D_1, D_2, \ldots, D_P)$.
1: **for** $i \in \{1, 2, \ldots, P\}$ **do**
2: $Classes \leftarrow$ Sample L_{num} labels from $\{0, 1, 2, \ldots, 9\}$;
3: $D_{num} \leftarrow$ RandInt(D_{min}, D_{max});
4: $weights \leftarrow$ Sample L_{num} number in interval (0,1);
5: $Data_{num} \leftarrow D_{num} * weights/sum(weights)$;
6: $D_i \leftarrow$ Sample data according to $Data_{num}$ and $Classes$.
7: **end for**

For the above four datasets, we deployed different CNN models. *For CelebA*, the model contains one convolutional layer with ReLU activation function followed by a maxpooling layer. The final pooled vector is passed to a dense layer with 16000 units. The loss was measured by the logarithmic softmax function. *For EMNIST ByClass*, it contains two convolutional layers with ReLU activation function followed by maxpooling layers. The final pooled vector is passed to a dense layer with 512 units. The loss was measured by the logarithmic softmax function. *For EMNIST MNIST*, the same CNN model for EMNIST ByClass's is deployed. *For CIFAR10*, we adopted LeNet [54], a simple and classical CNN model.

6.1.2 Comparison Algorithms

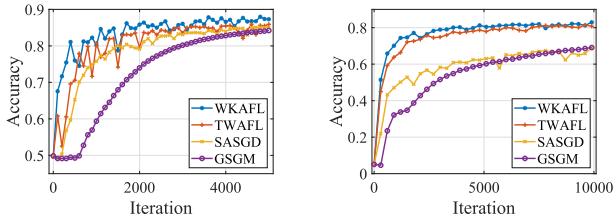
We compared three asynchronous algorithms in our experiments, including temporally weighted asynchronous federated learning (TWAFL [36]), staleness-aware async-SGD (SASGD, [33]), and gradient scheduling with global momentum (GSGM, [31]). TWAFL and SASGD, were proposed to

1. Source code is available at https://github.com/zzh816/WKAFL-code



(a) CelebA. (b) EMNIST ByClass.

Fig. 5. Relation between test accuracy and iteration on CelebA and EMNIST ByClass.



(a) CelebA. (b) EMNIST ByClass.

Fig. 6. Relation between test accuracy and iteration on CelebA and EMNIST ByClass.

alleviate the effect of staleness to improve the prediction accuracy, and GSGM was proposed to alleviate the effect of non-IID data to improve training stability.

TWAFL. Unlike aggregating stale gradients in WKAFL, the server in TWAFL directly aggregates the K stale model parameters by assigning the stale model parameters with staleness-based weights to improve the prediction accuracy. For the sake of fair comparison, we modify TWAFL in which the server aggregates K stale gradients instead of model parameters. In particular, the model update formula is expressed as follow:

$$w_{j+1} = w_j - \eta_j \sum_{i=1}^{K} \frac{m_i}{m} \left(\frac{e}{2}\right)^{-\tau_{j,i}} * g(w_{j,i}, \xi_{j,i}),$$

where $m_i, \tau_{j,i}, g(w_{j,i}, \xi_{j,i})$ are the mini-batch size, staleness, and gradients uploaded by client i at the jth iteration with $m = \sum_{i=1}^{K} m_i$.

SASGD. In SASGD, the sever first aggregates the K received stale gradients $g(w_{j,i}, \xi_{j,i})$ which are assigned with staleness-based weights $\eta_{j,i}$. Then, the server applies the aggregated gradient to update the model parameters. The specific updating formula is presented as follows:

$$\eta_{j,i} = \eta_0 / \tau_{j,i};$$
$$w_{j+1} = w_j - \frac{1}{K} \sum_{i=1}^{K} \eta_{j,i} g(w_{j,i}, \xi_{j,i}),$$

where η_0 is the initial learning rate and $\tau_{j,i}$ is the staleness for ith client at jth iteration.

GSGM. In GSGM with P clients totally, the server only uses one gradient at each iteration (fully AFL) and the whole training process is divided into several rounds. In each round, the server averages part gradients used in gradient scheduling strategy to modify the current inconsistent direction. According to this strategy, once the gradients submitted by a fast client are used for updating in one round, the client is blocked until next round begins. For the sake of fair comparison, we modify GSGM in which K gradients are selected to update the model at each iteration with other protocols unchanged and GSGM is also modified in the way that each client uploads only one gradient in one round to improve the accuracy of GSGM.

6.1.3 Metrics

In our experiments, we compared WKAFL with above algorithms in terms of training speed, model accuracy and training stability. The training speed is directly measured by the number of iterations before convergence. The model accuracy is measured by the prediction accuracy on test set. We describe the metric of training stability as follows.

Most studies focus more on the convergence rate and less on whether the training is stable. Particularly, a stable training indicates the training can be terminated with a relative stable model accuracy, instead of a fluctuated accuracy curve. In this paper, we use training stability to refer to the stability of last few model prediction accuracy on the test set before the final convergence. Particularly, it is measured by the standard deviation of the last A_{num} (which was set as 10) model accuracy values under logarithmic coordinates.

6.2 Ablation Experiment Result and Analysis

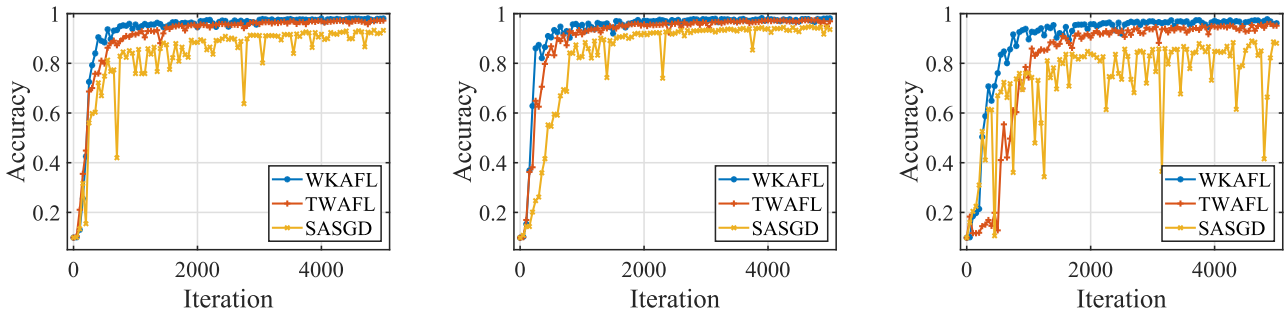
This part provides some ablation studies to demonstrate the impacts of each component of WKAFL. Specifically, WKAFL consists of three main components: gradients selection, and the strategies for staleness and non-IIDness. Therefore, in our ablation experiments, we removed them one by one, and the corresponding algorithms are named as WKAFL(No Grad-Sel), WKAFL(No Staleness) and WKAFL (No Non-IIDness) respectively.

Fig. 5 shows the test accuracy of four algorithms on both benchmark FL datasets CelebA and EMNIST ByClass. As shown, WKAFL can always converge fastest to the highest prediction accuracy, which indicates that every component in WKAFL is indispensable. WKAFL converges faster and more stable than WKAFL(No Grad-Sel), validating the existence of contradiction and the effectiveness of gradients selection. WKAFL(No Staleness) always has the poorest performance, which illustrates that staleness has more impacts on the model utility.

6.3 Comparison Experiment Result and Analysis

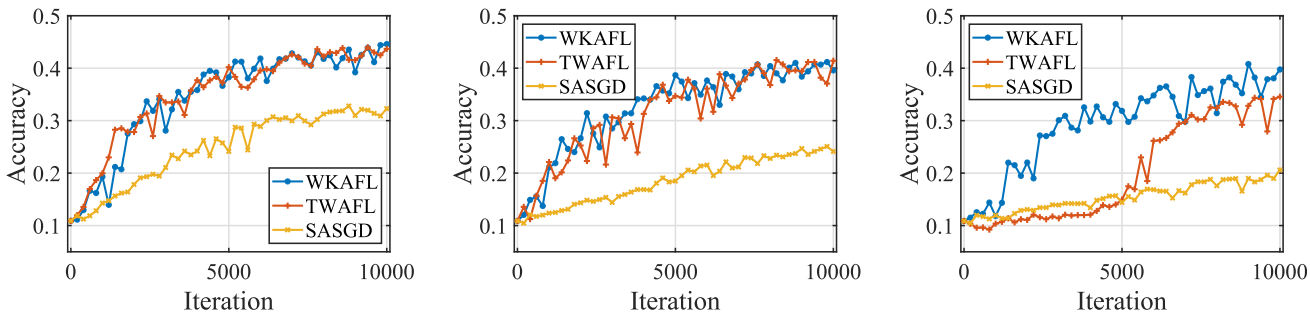
In this part, our purpose is to validate that the proposed WKAFL performs better not only on benchmark FL datasets, but also on modified FL datasets in terms of prediction accuracy and training stability. First, we implemented WKAFL and the comparison algorithms on two benchmark FL datasets, CelebA and EMNIST ByClass. Second, to comprehensively evaluate the performance of WKAFL, we implemented it on EMNIST MNIST and CIFAR10 which were modified to satisfy three levels of staleness and non-IID data respectively.

Fig. 6 shows the test accuracy of four algorithms on CelebA and EMNIST ByClass. The training speed of WKAFL is faster than the other three algorithms in the beginning process and WKAFL can always achieve a higher accuracy which validates that the proposed WKAFL behaves well.



(a) $P/K = 100$ (1000/10). (b) $P/K = 150$ (3000/20). (c) $P/K = 300$ (3000/10).

Fig. 7. Prediction accuracy on EMNIST MNIST with different levels of staleness measured by P/K. Here P is the number of total clients and K is number of participants at each iteration.



(a) $P/K = 50$ (1000/20). (b) $P/K = 150$ (3000/20). (c) $P/K = 300$ (3000/10).

Fig. 8. Prediction accuracy on CIFAR10 with different levels of staleness measured by P/K. Here P is the number of total clients and K is number of participants at each iteration.

The more comprehensive comparisons in terms of staleness and non-IID data are illustrated in Sections 6.3.1 and 6.3.2 respectively.

6.3.1 Different Degrees of Staleness

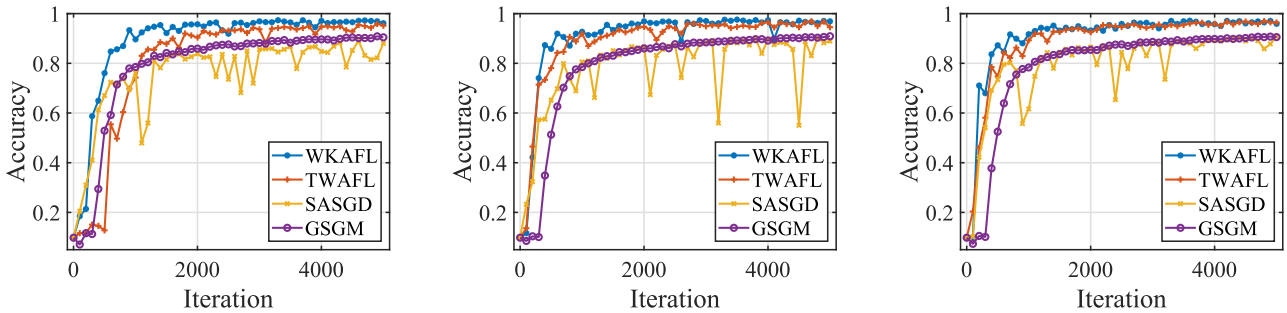
Based on that the minimum average of staleness approximately equals $P/(2K)$ (refer to Appendix D.1, available in the online supplemental material), we achieve different levels of staleness by setting different values of P (number of total clients) and K (number of participants at each iteration) and the level of staleness can be measured as P/K for convenience. For both EMNIST MNIST and CIFAR10, we set three levels of staleness, $1000/10 = 100, 3000/20 = 150$, $3000/10 = 300$ on EMNIST MNIST and $1000/20 = 50, 3000/20 = 150$, $3000/10 = 300$ on CIFAR10. The experimental results of test accuracy under the three levels on EMNIST MNIST and CIFAR10 is shown in Figs. 7 and 8 respectively and the final prediction accuracy is summarized in Table 2.

TABLE 2
Prediction Accuracy With Different Levels Of Staleness

Datasets	Staleness	Prediction Accuracy		
		WKAFL	TWAFL	SASGD
EMNIST MNIST	100	**0.9734**	0.962	0.9319
	150	**0.9756**	0.9649	0.8753
	300	**0.9728**	0.9572	0.8553
CIFAR10	50	**0.4433**	0.4246	0.3308
	150	**0.4368**	0.4241	0.2560
	300	**0.3970**	0.3468	0.2029

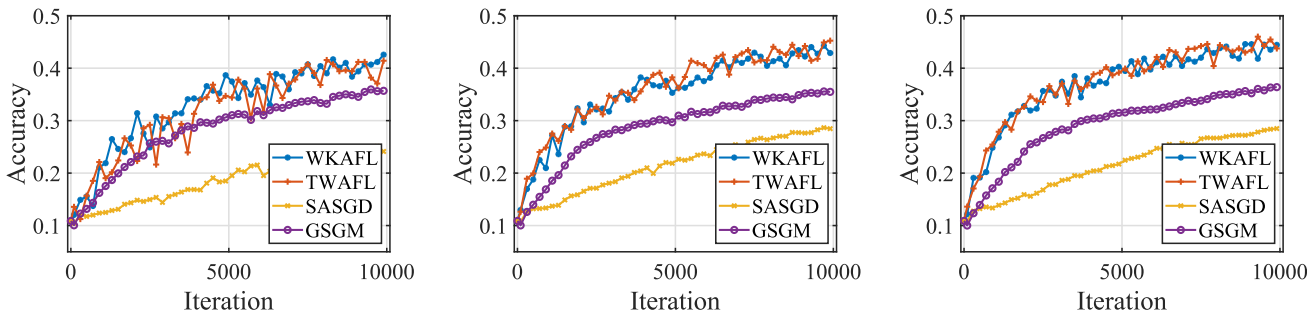
Comparisons on EMNIST MNIST. Three conclusions are obtained from Fig. 7 and Table 2 (rows 3-5). *First*, WKAFL has the fastest training speed compared to algorithms TWAFL and SASGD under the three levels of staleness as shown in Fig. 7, especially for high staleness in Fig. 7c. This validates the idea of exploiting the stale gradients to accelerate the training process (as discussed in Section 4.3.3). *Second*, Fig. 7 shows that WKAFL has higher prediction accuracy than the compared algorithms, especially for high staleness which benefits from the strategies of clipping bound the stale gradients (as discussed in Section 4.3.2), selecting consistent gradients (as discussed in Section 4.3.3) and adaptively adjusting the learning rate (as discussed in Section 4.3.4), especially when the model is going to converge (Stage two). *Third*, WKAFL is more robust to staleness. Based on the decreasing prediction accuracy in each column (rows 3-5) in Table 2, the staleness has a negative impact on the prediction accuracy for all algorithms which is consistent with the analysis of the negative effect of staleness (Theorems 5.1 and 5.3). We note that the accuracy reduction of WKAFL is the lowest while the accuracy reduction of SASGD is the highest. Apart from it, the curves of TWAFL and SASGD have more obvious changes than the curve of WKAFL when the level of staleness increases as shown in Fig. 7.

Comparisons on CIFAR10. Similar results are obtained from Fig. 8 and Table 2 (rows 6-8). When it comes to dealing



(a) $L_{num} = 1$ (b) $L_{num} = 5$ (c) $L_{num} = 10$

Fig. 9. Relation between test accuracy and iteration on EMNIST MNIST with different levels of Non-IID degrees (L_{num}). A smaller L_{num} means a higher level of non-IID data. The level of staleness was fixed as 300 ($P/K = 3000/10$).



(a) $L_{num} = 1$ (b) $L_{num} = 5$ (c) $L_{num} = 10$

Fig. 10. Relation between test accuracy and iteration on CIFAR10 with different levels of Non-IID degrees (L_{num}). A smaller L_{num} means a higher level of non-IID data. The level of staleness was fixed as 150 ($P/K = 3000/20$).

with stale gradients and using the adaptive learning rate, the proposed WKAFL outperforms the compared algorithms in terms of training speed and prediction accuracy. Besides, Table 2 shows that the robustness advantage of WKAFL is more significant than that on EMNIST MNIST. Particularly, when staleness increases from 100 to 300, the accuracy of WKAFL decreases from 0.4433 to 0.3970 while the accuracy significantly decreases from 0.4246 to 0.3468 (from 0.3308 to 0.2029)for TWAFL (SASGD) algorithms which validates that WKAFL is extensible for FL with large scale distributed clients in which the staleness is usually high.

Additionally, we explain the reason of low prediction accuracy (about 44.5 percent) on CIFAR10 achieved here which is significantly lower than the existing high value (around 95 percent)[2]. The main reason is the memory restrictions of using the complicated CNN architecture presented in [55], [56]. To validate the proposed WKAFL for scenarios with large scale distributed devices, the number of total clients P was set as 3000. In AFL, 3000 duplicates of CNN are needed to be stored and computed because of the asynchronous manner. However, this requires a computer with large memory size. Therefore, we adopted a light model LeNet whose prediction accuracy under centralized machine learning is only around 67 percent[3]. Based on it, the achieved decentralized accuracy affected by both non-IID data and high staleness is acceptable and our main purpose is to validate the performance of WKAFL for any given model and dataset.

6.3.2 Different Degrees of Non-IID Data

Non-IID data is a basic characteristic of FL, however, there is no exact metrics of it. As adopted in [9], we also use the number of label classes L_{num} to measure non-IID level. For example, for CIFAR10 which has labels $0, \ldots, 9$, $L_{num} = 1$ means that each client only owns data with one label, such as 5. Obviously, a small value L_{num} means high level of non-IID data. We set three levels of data heterogeneity on both EMNIST MNIST and CIFAR10, $L_{num} = 1, 5, 10$. The experimental results of test accuracy on EMNIST MNIST and CIFAR10 are shown in Figs. 9 and 10 respectively. Table 3 shows the final prediction accuracy and training stability.

Comparisons on EMNIST MNIST. Four conclusions are drawn from Fig. 9 and Table 3.

First, Fig. 9 shows that the proposed WKAFL converges faster than other three algorithms for all three levels of data heterogeneity, especially for the highest level (Fig. 9a with $L_{num} = 1$). This advantage benefits from full use of historical gradients (as discussion in Section 4.3.1). As the level of non-IID data increases, the direction of uploaded gradients will seriously deviate from the globally unbiased gradient and accumulated gradients can narrow the gap between uploaded gradients and consistent gradients. Therefore, the

2. https://github.com/junyuseu/pytorch-cifar-models
3. https://github.com/icpm/pytorch-cifar10

TABLE 3
Prediction Accuracy and Training Stability With Three Levels of Non-IID Degree Measured by L_{num}

Dataset	L_{num}	Final Accuracy				Model Stability			
		WKAFL	TWAFL	GSGM	SASGD	WKAFL	TWAFL	GSGM	SASGD
EMNIST MNIST	1	**0.9728**	0.9572	0.9057	0.8553	0.0060	0.0107	**0.0032**	0.0834
	5	**0.9754**	0.9649	0.908	0.8753	0.0068	0.0095	**0.0027**	0.0237
	10	**0.9658**	0.9623	0.9075	0.8619	0.0036	0.0060	**0.0018**	0.0206
CIFAR10	1	**0.4368**	0.4141	0.3568	0.2560	0.0284	0.0464	**0.0135**	0.0226
	5	0.4443	**0.4454**	0.3568	0.2855	0.0133	0.0300	**0.0083**	0.0141
	10	0.4489	**0.4525**	0.3430	0.2874	0.0180	0.0260	**0.0144**	0.0189

advantage of WKAFL is more significant for the high level of non-IID data.

Second, the proposed WKAFL achieves the highest prediction accuracy for all three levels of data heterogeneity as shown in columns 3-6 of Table 3 (rows 3-5) which illustrates that exploiting the historical gradients can alleviate the impact of non-IID data (as explained in Section 4.3.1), leading to a higher accuracy.

Third, the training stability decreases as the level of non-IID data increases, i.e., L_{num} from $10, 5$ to 1, as shown in columns 7-10 of Table 3 (rows 3-5). This is because the gradient's direction of non-IID data is likely to be deviated from the globally unbiased gradient. At different iterations, the direction of aggregated gradients based on K participants will differently deviate from the globally unbiased gradient. Therefore, higher heterogeneity means more uncertainty. Experiment results validate our analysis (Section 5).

Fourth, WKAFL has a better training stability as shown in columns 7-10 of Table 3 (row 3-5). A small value of training stability means a stable model. Although WKAFL does not have the best training stability, the comprehensive advantage of WKAFL is still significant, which can be described from two aspects. On the one hand, compared to GSGM with the best training stability, its prediction accuracy (row 3-5 in Table 3) and convergence rate (Fig. 9) is significantly lower than WKAFL. For example, when $L_{num} = 1$, the prediction accuracy for GSGM is only 85.53 percent, significantly lower than 97.28 percent for WKAFL. However, the training stability of WKAFL, i.e., the fluctuation degree in Figs. 9b and 9c, is almost the same as GSGM. On the other hand, compared to TWAFL who has a slight prediction accuracy reduction, its training stability is significantly weaker than WKAFL. For example, when $L_{num} = 1$, the training stability for TWAFL is 0.0107, almost with 44 percent increment than 0.006 for WKAFL. Besides, Fig. 9a shows that the fluctuation of TWAFL is more obvious than WKAFL.

Comparisons on CIFAR10. Similar conclusions are drawn from Fig. 10 and Table 3, based on the same analysis on EMNIST MNIST. The only difference is that TWAFL achieves a slight improvement of prediction accuracy than the proposed WKAFL when $L_{num} = 5, 10$ (columns 2-3) because of the randomness in the experiments. In fact, TWAFL and WKAFL have almost the same training process in low level of data heterogeneity as shown in Figs. 10b and 10c.

7 Conclusion

In this paper, we propose a two-stage weighted K-async FL (WKAFL) algorithm to improve the model utility of AFL These improvements are achieved from three aspects. First, WKAFL estimates the globally unbiased gradient by accumulating historical gradients to alleviate the impact of non-IID data and aggregating K gradients based on the staleness. Second, WKAFL picks gradients consistent with the estimated gradient and assigns them with a high weight and vice versa to improve the effect for mitigating non-IID data while preventing the stale gradients to bring down model utility. Third, by further clipping the stale gradient in the second stage and adjusting the learning rate based on staleness, WKAFL improves the final prediction accuracy. The experiment results on four FL datasets validate that WKAFL can accelerate the training process and improve final prediction accuracy while guaranteeing a stable model, especially in settings with high staleness or high level of non-IID data.

References

[1] J. Chen, K. Li, Q. Deng, K. Li, and P. S. Yu, "Distributed deep learning model for intelligent video surveillance systems with edge computing," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/TII.2019.2909473.

[2] L. Deng *et al.*, "Recent advances in deep learning for speech research at microsoft," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 8604–8608.

[3] J. Chen, K. Li, K. Bilal, x. zhou, K. Li and P. S. Yu, "A Bi-layered parallel training architecture for large-scale convolutional neural networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 965–976, May 2019.

[4] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2722–2730.

[5] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[6] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*.

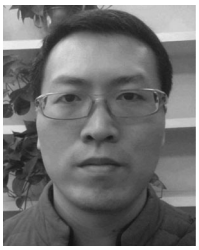
[7] M. Duan, D. Liu, X. Chen, R. Liu, Y. Tan, and L. Liang, "Self-balancing federated learning with global imbalanced data in mobile systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 59–71, Jan. 2021.

[8] C. Wang, Y. Yang, and P. Zhou, "Towards efficient scheduling of federated mobile devices under computational and statistical heterogeneity," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 2, pp. 394–410, Feb. 2021.

[9] W. Liu, L. Chen, Y. Chen, and W. Zhang, "Accelerating federated learning via momentum gradient descent," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 8, pp. 1754–1766, Aug. 2020.

[10] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, 2019, Art. no. 12.
[11] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.
[12] P. Voigt and A. Von dem Bussche, "The EU general data protection regulation (GDPR)," in *A Practical Guide*, 1st Ed., Berlin, Germany: Springer, 2017.
[13] M. J. Sheller *et al.*, "Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data," *Sci. Rep.*, vol. 10, no. 1, pp. 1–12, 2020.
[14] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *Int. J. Med. Informat.*, vol. 112, pp. 59–67, 2018.
[15] W. Yang, Y. Zhang, K. Ye, L. Li, and C.-Z. Xu, "Ffd: A federated learning based method for credit card fraud detection," in *Proc. Int. Conf. Big Data*, 2019, pp. 18–32.
[16] T. Qi, F. Wu, C. Wu, Y. Huang, and X. Xie, "Fedrec: Privacy-preserving news recommendation with federated learning," 2020, *arXiv:2003.09592*.
[17] B. Malle, N. Giuliani, P. Kieseberg, and A. Holzinger, "The more the merrier-federated learning from local sphere recommendations," in *Proc. Int. Cross-Domain Conf. Mach. Learn. Knowl. Extraction*, 2017, pp. 367–373.
[18] L. Lyu *et al.*, "Towards fair and privacy-preserving federated deep models," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2524–2541, 2020.
[19] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," 2019, *arXiv:1907.02189*.
[20] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. A. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, May 2021.
[21] J. Cipar *et al.*, "Solving the straggler problem with bounded staleness," in *Proc. 14th Workshop Hot Top. Oper. Syst.*, 2013, pp. 14–19.
[22] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 3368–3376.
[23] S. Dutta, G. Joshi, S. Ghosh, P. Dube, and P. Nagpurkar, "Slow and stale gradients can win the race: Error-runtime trade-offs in distributed SGD," 2018, *arXiv:1803.01113*.
[24] S. Dutta, V. Cadambe, and P. Grover, "Short-Dot: Computing large linear transforms distributedly using coded short dot products," in *Proc. 30th Conf. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2100–2108.
[25] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.
[26] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Differentially private asynchronous federated learning for mobile edge computing in urban informatics," *IEEE Trans. Ind. Informat.*, vol. 16, no. 3, pp. 2134–2143, Mar. 2020.
[27] Y. Chen, Y. Ning, and H. Rangwala, "Asynchronous online federated learning for edge devices," 2019, *arXiv:1911.02134*.
[28] Y. Li, S. Yang, X. Ren, and C. Zhao, "Asynchronous federated learning with differential privacy for edge intelligence," 2019, *arXiv:1912.07902*.
[29] R. Hannah and W. Yin, "More iterations per second, same quality–why asynchronous algorithms may drastically outperform traditional ones," 2017, *arXiv:1708.05136*.
[30] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.
[31] C. Li *et al.*, "Gradient scheduling with global momentum for non-IID data distributed asynchronous training," 2019, *arXiv:1902.07848*.
[32] W. Dai, Y. Zhou, N. Dong, H. Zhang, and E. P. Xing, "Toward understanding the impact of staleness in distributed machine learning," 2018, *arXiv:1810.03264*.
[33] W. Zhang, S. Gupta, X. Lian, and J. Liu, "Staleness-aware Async-SGD for distributed deep learning," 2015, *arXiv:1511.05950*.
[34] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5132–5143.
[35] X. Liang, S. Shen, J. Liu, Z. Pan, E. Chen, and Y. Cheng, "Variance reduced local sgd with lower communication complexity," 2019, *arXiv:1912.12844*.
[36] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.
[37] C. Xie, O. Koyejo, I. Gupta, and H. Lin, "Local adaalter: Communication-efficient stochastic gradient descent with adaptive learning rates," 2019, *arXiv:1911.09030*.
[38] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 2921–2926.
[39] I. Mitliagkas, C. Zhang, S. Hadjis, and C. Ré, "Asynchrony begets momentum, with an application to deep learning," in *Proc. 54th Annu. Allerton Conf. Commun. Control Comput.*, 2016, pp. 997–1004.
[40] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization for heterogeneous networks," 2018, *arXiv:1812.06127*.
[41] A. Khaled, K. Mishchenko, and P. Richtárik, "First analysis of local GD on heterogeneous data," 2019, *arXiv:1909.04715*.
[42] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," 2019, *arXiv:1905.09435*.
[43] B. McMahan and M. Streeter, "Delay-tolerant algorithms for asynchronous distributed online learning," in *Proc. 27th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2915–2923.
[44] H. B. McMahan and M. Streeter, "Adaptive bound optimization for online convex optimization," 2010, *arXiv:1002.4908*.
[45] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, no. 7, pp. 2121–2159, 2011.
[46] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Proc. 28th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2015, pp. 2737–2745.
[47] J. H. Lee, J. Sim, and H. Kim, "BSSync: Processing near memory for machine learning workloads with bounded staleness consistency models," in *Proc. Int. Conf. Parallel Archit. Compilation*, 2015, pp. 241–252.
[48] R. Hannah and W. Yin, "On unbounded delays in asynchronous parallel fixed-point algorithms," *J. Sci. Comput.*, vol. 76, no. 1, pp. 299–326, 2018.
[49] C. Xie, S. Koyejo, and I. Gupta, "Zeno++: Robust fully asynchronous SGD," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10495–10503.
[50] M. Chen, B. Mao, and T. Ma, "A staleness-aware asynchronous federated learning algorithm with non-iid data," *Future Gener. Comput. Syst.*, vol. 120, pp. 1–12, 2021.
[51] T. Ryffel *et al.*, "A generic framework for privacy preserving deep learning," 2018, *arXiv:1811.04017*.
[52] S. Caldas *et al.*, "Leaf: A benchmark for federated settings," 2018, *arXiv:1812.01097*.
[53] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's thesis, Comput. Sci. Dep., Univ. Toronto, Toronto, ON, Canada, 2009.
[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[55] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2261–2269.
[56] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5987–5995.

Zihao Zhou received the bachelor's degree in 2020 from the School of Mathematics and Statistics, Xi'an Jiaotong University, China, where he is currently working toward the PhD degree with the School of Mathematics and Statistics. His research interests include federated learning and edge-cloud intelligence.



Yanan Li received the bachelor's and master's degrees from Henan Normal University, China, in 2004 and 2007, respectively. He is currently working toward the PhD degree with the School of Mathematics and Statistics, Xi'an Jiaotong University. Before that, he was a lecturer with Henan Polytechnic University from 2007 to 2017. His research interests include machine learning, federated learning, and edge-cloud intelligence.

Xuebin Ren (Member, IEEE) received the PhD degree from the Department of Computer Science and Technology, Xi'an Jiaotong University (XJTU), Xi'an, China, in 2017. He is currently an associate professor with the School of Computer Science and Technology and a member of the National Engineering Laboratory for Big Data Analytics (NEL-BDA), XJTU. From 2016 to 2017, he was a visiting PhD student with the Department of Computing, Imperial College London, U.K. His research interests include data privacy protection, federated learning, and privacy-preserving machine learning. He is a member of the ACM.

Shusen Yang (Senior Member, IEEE) received the PhD degree in computing from Imperial College London in 2014. He is currently a professor and the director of National Engineering Laboratory for Big Data Analytics, and the deputy director of the Ministry of Education(MoE) Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University (XJTU), Xi'an, China. Before joining XJTU, he was a lecturer (assistant professor) with the University of Liverpool from 2015 to 2016, and a research associate with Intel Collaborative Research Institute (ICRI) on sustainable connected cities from 2013 to 2014. His research interests include distributed systems and data sciences, and their applications in industrial scenarios, including data-driven network algorithms, distributed machine learning, edge-cloud intelligence, industrial internet and industrial intelligence. He is a DAMO academy young fellow, and an honorary research fellow with Imperial College London. He is a member of the ACM.