From Deterioration to Acceleration: A Calibration Approach to Rehabilitating Step Asynchronism in Federated Optimization

Feijie Wu, Song Guo, Fellow, IEEE, Haozhao Wang, Haobo Zhang, Zhihao Qu, Member, IEEE, Jie Zhang, Member, IEEE, and Ziming Liu

Abstract—In the setting of federated optimization, where a global model is aggregated periodically, step asynchronism occurs when participants conduct model training by efficiently utilizing their computational resources. It is well acknowledged that step asynchronism leads to objective inconsistency under non-i.i.d. data, which degrades the model’s accuracy. To address this issue, we propose a new algorithm FedGrac, which calibrates the local direction to a predictive global orientation. Taking advantage of the estimated orientation, we guarantee that the aggregated model does not excessively deviate from the global optimum while fully utilizing the updates of faster nodes. We theoretically prove that FedGrac holds an improved order of convergence rate than the state-of-the-art approaches and eliminates the negative effect of step asynchronism. Empirical results show that our algorithm accelerates the training and enhances the final accuracy.

Index Terms—Computational heterogeneity, data heterogeneity, federated learning.

I. INTRODUCTION

FEDERATED learning (FL) is thriving as a promising paradigm that refrains the leakage of users’ data, including raw information and labels distribution. With the rapid development of FL techniques over the past few years, a wide range of applications for computer vision [1], [2] and natural language processing [3], [4], [5] have deployed over a large set of edge devices (e.g., smartphones and tablets). Conventionally, clients perform a fixed number of local stochastic gradient descent (SGD) steps in each round; then, the server aggregates the updated models and finally acquires and distributes the global one to all clients [6], [7]. FedAvg follows the preceding procedure and has been proven to be a promising solution to data heterogeneity.

With an increasing number of nodes participating in the training, the traditional framework becomes infeasible because the computation capacities are substantially diverse among devices [8]. A practical framework allows clients to update the local model via a flexible number of local SGD steps in each round according to its available resource capacity. And we define such a procedure as step asynchronism (see Fig. 1 for visualized demonstration). To comprehensively understand the training performance of the traditional algorithm FedAvg, Table I compares the results in terms of test accuracy in two situations – step asynchronism and data heterogeneity. This experiment is under convex (i.e., logistic regression) and non-convex (i.e., 2-layer CNN) objectives using a public dataset Fashion-MNIST [9]. Performance deterioration is noticeable, especially in the logistic regression model the desired test accuracy cannot be reached.

A previous study [10] owes the performance deterioration to objective inconsistency, where the FL training converges

Table I: The number of communication rounds to reach the test accuracy of 80% under logistic regression (LR) and 2-Layer CNN on Fashion-MNIST with various settings when 10 devices participate in FedAvg. The number of local updates is 100 without step asynchronism, while under step asynchronism, clients perform at least 100 local updates. The learning rates set for LR and 2-Layer CNN are 0.001 and 0.003, respectively, which are also applied to Table II, Fig. 2, and Fig. 3

| FedAvg with | LR | 2-layer CNN |
|-------------|----|-------------|
| neither     | 2  | 20          |
| step async  | 1  | 8           |
| non-i.i.d.  | 91 | 265         |
| both        | 1K+| 339         |
not only fully utilizes the computational resources, but also achieves a better accuracy than FedNova (see Table II).

Our key contributions to this work are listed as follows:

1) To explore the factors that lead to performance deterioration, we analyze the convergence property under strongly-convex objectives. The theoretical result indicates that the expected loss never reaches the optimal one when both data heterogeneity and step asynchronism exist. In other words, a constant number of local updates eliminates the negative effect of data distribution differentiation, while step asynchronism magnifies the drawback of data heterogeneity.

2) We design a novel method named FedGrac to address the problem of objective inconsistency via predictive gradient calibration, which makes the direction of each local update close to the direction towards the global optimum. For the first time, our algorithm can jointly address statistical heterogeneity and computation heterogeneity at a time.

3) We establish the convergence rate of FedGrac. Under non-convex objectives, the algorithm achieves a convergence rate of \(O(1/\sqrt{MTK})\), where \(M\) and \(T\) represent the number of clients and communication rounds, respectively, and \(K\) indicates the weighted averaged number of local updates. This convergence rate is also achieved by FedNova only under the condition that \(K_{\max}/K_{\min} = O(M)\), where \(K_{\max}\) and \(K_{\min}\) separately refer to the maximum and minimum number of local updates \([10]\). Otherwise, the actual convergence rate of FedNova should be \(O(\sqrt{K/MT})\). Apparently, our algorithm can achieve a faster convergence rate by a factor up to \(O(K)\).

4) We conduct extensive experiments to compare the proposed FedGrac with typical and latest works such as SCAFFOLD [11] and FedNova [10]. In terms of convergence rate, FedGrac achieves higher convergence efficiency compared to FedAvg and FedNova, especially in scenarios with high heterogeneity. For example, in terms of test accuracy, our algorithm can always preserve convergence while SCAFFOLD and FedNova cannot in some cases.

The rest of this paper is organized as follows. First, Section II provides related work and background knowledge of distributed SGD and existing solutions to heterogeneous training. Next, we...
state preliminaries and problem formulation for the heteroge-
neous Federated Learning in Section III. Then, in Section IV,
we design a novel algorithm FedaGrac to solve the problem.
In Section V, we analyze its convergence property. After that,
we present our experimental results to evaluate our method in
Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK

Federated Learning. Frequently, edge devices such as smart-
phones possess abundant data, which are highly sensitive but
useful to the model training [12], [13], [14], [15]. To utilize these
data, FL is conceived to search for a generalized model [16],
[17] or personalized models [18], [19] while safeguarding the
data privacy [6], [20]. Apparently, the data are heterogeneous
among clients because there are no predefined rules for the
data distribution for each client. Besides, due to the hardware
differences among devices, the computational capabilities are
various. In this section, we briefly investigate the flaws raised
by data heterogeneity and computation heterogeneity and review
the existing work to tackle these two issues.

Data Heterogeneity. Generally, in FL settings, the data distri-
buted among clients are agnostic and therefore, each data
portfolio has its exclusive optimal parameters. As a classical
algorithm to combat data heterogeneity, FedAvg inherits the
training features from local SGD [21], [22], [23], a framework
that runs for multiple local updates prior to a global synchro-
nization. Obviously, this strategy significantly reduces the total
communication overhead when compared to parallel SGD that
synchronizes the gradient at every local update. Recent studies
[7], [24], [25] show that FedAvg can have a great performance
from theoretical and empirical perspectives. Also, FedAvg can
seamlessly adopt communication-efficient approaches such as
quantization [26], [27] and sparsification [28], [29] to further reduce the cost of transmission [30], [31], [32], [33].

Nevertheless, numerous studies [11], [34], [35], [36], [37]
thereoretically prove that the issue raise the client-drift effect and degrades the convergence property. To mitigate the nega-
tive impact, existing solutions include cross-client variance
reduction [11], [38], client clustering sampling [39], [40], [41]
and reinforcement learning driven incentive mechanism [42].
Among these approaches, SCAFFOLD [11] is a superior option
that adjusts every local update with the help of the global
and a client’s local reference orientation, such that every local
update keeps close to the global direction. However, as shown in
Fig. 1, SCAFFOLD cannot completely remove the drift. A phys-
ical explanation for the result is that the local reference directions
of the faster nodes with more number of local updates lead to
a significant deviation from the orientation towards the local
optimizer. Since the global reference direction is aggregated by
clients’ local ones, it is intuitively dominated by the faster nodes
(see Fig. 1), which betrays its origin intention. Although we use a
similar design philosophy that ensures every local update along
with the global orientation, the global orientation consists of the
gradient that depends on the number of local updates, either the
normalized gradient or the initial gradient.

Computational Heterogeneity. The computation capabilities
vary among clients because they use different devices. To mini-
mize the computation differences, some existing works adopt
a client sampling strategy [43], [44], [45], [46], where only
a small portion of clients transmit the gradients to the server.
Compared to the case that requires full-worker participation,
this scheme reduces the total training time. However, there still
exists resource underutilization as the fastest client should wait
for others’ completion.

A practical solution is to adopt step asynchronism, where
each client performs an inconsistent number of local updates.
Although FedAvg with step asynchronism can converge to a
stable point under non-convex objectives [22], Wang et al. [10]
point out that objective inconsistency takes place under quadratic
function. To alleviate the challenge of computational hetero-
genality, effective approaches are constituted with normalization-
based approach FedNova [10] and FedLin [47], regularization-
based approach FedProx [48] and architecture-based approach
HeteroFL [49]. Gradient normalization is the most ubiquitous
framework that overcomes step asynchronism under non-i.i.d.
data setting. However, this method cannot prevent the negative
impact of statistical heterogeneity on the convergence rate be-
cause the update deviation still exists after averaging. Fig. 1
compares FedNova [10] and FedLin [47] with our proposed
method, and we notice that the global model deviates to the
one with less updates in FedNova [10]. The reason is obvious:
clients update the models bias to their local datasets such that
the normalized gradients collected by the server are sparse. Besides,
with the local models approaching the local minimizers, the
update becomes so trivial that those clients with more local
updates have a dispensable influence on the global model update.

III. PRELIMINARY AND PROBLEM FORMULATION

Formally, the learning problem can be represented as the
following distributed optimization problem across $M$ FL clients

$$
\min_{x \in \mathbb{R}^d} F(x) = \sum_{i=1}^{M} \omega_i F_i(x),
$$

where the weight $\omega_i = |D_i|/|D|$ is the ratio between the size of
local dataset $D_i$ and overall dataset $D \triangleq \bigcup_{i=1}^{M} D_i$, and $F_i(x) \triangleq \mathbb{E}_{e_i \sim D_i} [f_i(x; e_i)]$ is the the local objective, i.e., the expected loss
value of model $x$ with respect to random sampling $e_i$ for client $i$.

FedAvg with step asynchronism. Naive weighted aggrega-
tion [6], [7], [21], [22] is an effective and communication-effi-
cient way to solve Problem (1) for both convex and non-
convex objectives. With the increasing number of edge devices
participating in model training, the framework is neither eco-
nomic nor fair to require all clients to run a certain number
of local updates. Instead, a practical approach is that client
$i \in \{1, \ldots, M\}$ runs for a flexible number of SGD steps (i.e., $K_i$)
according to its resource capability before the model aggregation
at the server:

- **(Pull):** Pulls the current parameter $x_0$ from the server.
- **(Compute):** Samples a realization $\varepsilon$ randomly from the
local dataset $D_i$ and compute the gradient $\nabla f_i(x_k, \varepsilon)$. 

**Fig. 1** Scaffolding cannot completely remove the drift.
\textbf{Update):} Performs $k$-th local update of the form $\eta$ by $x_{k+1} = x_k - \eta g_k$, where $k \in \{0, \ldots, K\}$ and $\eta$ is the stepsize.

\textbf{Push):} Pushes the local parameter $x_K$ to the server.

Under this framework, we let $K_{\text{max}}$ and $K_{\text{min}}$ separately be the maximum and the minimum number of local updates among all clients, i.e., $K_{\text{max}} = \max_{i \in \{1, \ldots, M\}} K_i$ and $K_{\text{min}} = \min_{i \in \{1, \ldots, M\}} K_i$. In addition, $K = \sum_{i=1}^{M} \omega_i K_i$ is defined as the weighted averaged number of local updates. Formally, step asynchronism is defined as the following mathematical expression:

$$\exists i, j \in \{1, \ldots, M\}, \ K_i \neq K_j.$$  

Therefore, $K_{\text{max}} \neq K_{\text{min}}$ when step asynchronism exists. Without extra explanations, these notations are adopted throughout the paper.

\textbf{Assumptions.} To establish the convergence theory of the FL optimization, we make the following assumptions that are adopted in previous works [7], [10], [48], [50], [51]:

Assumption 1 (L-smooth). The local objective functions are Lipschitz smooth: For all $v, \tilde{v} \in \mathbb{R}^d$

$$\|\nabla F_i(v) - \nabla F_i(\tilde{v})\|_2 \leq L \|v - \tilde{v}\|_2, \ \forall i \in \{1, \ldots, M\}.$$  

Assumption 2 ($\mu$-strongly convex). The local objective functions are $\mu$-strongly convex with the value of $\mu > 0$: For all $v, \tilde{v} \in \mathbb{R}^d$

$$F_i(v) - F_i(\tilde{v}) \geq \langle \nabla F(\tilde{v}), (v - \tilde{v}) \rangle + \frac{\mu}{2} \|v - \tilde{v}\|^2, \ \forall i \in \{1, \ldots, M\}$$

where $\langle \cdot, \cdot \rangle$ refers to the inner product of two gradients.

Assumption 3 (Bounded Variance). For all $v \in \mathbb{R}^d$, there exists a scalar $\sigma \geq 0$ such that

$$\mathbb{E}\|\nabla f_i(v, \varepsilon) - \nabla F_i(v)\|_2^2 \leq \sigma^2, \ \forall i \in \{1, \ldots, M\}$$

Assumption 4 (Bounded Dissimilarity). For some $v \in \mathbb{R}^d$ that $\|\nabla F(v)\|_2^2 > 0$ holds, there exists a scalar $B \geq 1$ such that

$$\mathbb{E}\|\nabla F_i(v)\|_2^2 \leq B^2 \|\nabla F(v)\|_2^2, \ \forall i \in \{1, \ldots, M\}.$$  

Obviously, when the data are independent and identically distributed, the value of $B$ should be 1.

Assumption 4 seems to be a little bit strong as $\|\nabla F_i(v)\|_2^2$ cannot be 0. However, considering $\varepsilon$-accuracy as the learning rate criterion, i.e., $\|\nabla F_i(v)\|_2^2 \leq \varepsilon$ under non-convex objectives such as deep neural networks which possess multiple local minimizers, the value of $\varepsilon$ cannot strictly be 0. In other words, there exists $\varepsilon_1 \leq \varepsilon$ such that $\|\nabla F_i(v)\|_2^2 \geq \varepsilon_1$ for all $v$ always holds.

Key factor that raises objective inconsistency. Although [10] indicates that objective inconsistency occurs when using FedAvg with step asynchronism under quadratic functions, the factor that makes it happen remains a mystery. To explore in depth, the following theorem analyzes FedAvg with step asynchronism under a strongly-convex objective.

Theorem 1. Suppose the local objective functions are non-negative. Denote the parameter at $t$-th communication round by $x_t$. Let $T$ be the total number of communication rounds.

Under Assumption 1, 2, 3 and 4, by setting the learning rate $\eta = O(1/\mu LT K) \leq 1/LK$, the output of FedAvg with step asynchronism satisfies

$$\lim_{T \to \infty} \mathbb{E}[F(x_T)] - F(x_*) \leq O \left( \sum_{i=1}^{M} \omega_i \left( \frac{K_i}{K_{\text{min}}} - 1 \right) F_i(x_*) \right)$$

where $x_1$ and $x_*$ indicates the initial and optimal model parameters, respectively.

\textbf{Remark.} The theoretical result in (3) is consistent with the result of FedAvg analysis in [11] as the number of local updates is identical, i.e., $K_i = K, \forall i \in \{1, \ldots, M\}$. Besides, when the data are identical and independent distributed among clients, where the global optimizer is not equivalent to the clients’ local minimizer, we can easily induce that $x_T$ is close to $x_*$ when $T \to \infty$. The conclusion holds regardless of the number of local updates. However, when data heterogeneity and step asynchronism coexist, the right-hand side of (3) is non-zero. As a result, when $T$ tends to be infinite, the model cannot converge to the optimal parameters, which can explain the result manifested in Table I under LR. Based on the theoretical discovery, we can draw a conclusion that step asynchronism leads to a significant accuracy drop in the non-i.i.d. cases, which impedes normal training.

IV. FedGrac Algorithm

To ensure that $\mathbb{E}[F(x_T)] - F(x_*)$ is close to 0 when $T \to \infty$, we target to remove the constant term in the right-hand side of (3). Based on the remark in Section III, a practical approach is to minimize the effectiveness of data heterogeneity. In this section, we elaborate our proposed algorithm, FedGrac Accelerating Gradient Calibration, to avoid the objective inconsistency as well as enhance the convergence performance when step asynchronism is adopted to improve the resource utilization. The implementation details are presented as Algorithm 1.

At first, apart from the hyperparameters such as learning rate $\eta$ and calibration rate $\lambda$, we initialize a $d$-dimension model with arbitrary parameters $x_1$. Besides, to ease the theoretical analysis in Section V, we set $\nu^{(i)}$ as $\nabla f_i(x_1, D_i)$ for all $i \in \{1, \ldots, M\}$. Then, we define $\nu$ as

$$\nu = \sum_{i=1}^{M} \omega_i \nu^{(i)} = \sum_{i=1}^{M} \omega_i \nabla f_i(x_1, D_i).$$

In this algorithm, client $i$ performs the local updates for $K_i$ times in parallel. During each local update, clients calibrate the local client deviation with reference to the global reference orientation, which is estimated at every global synchronization. In the following two subsections, we separately discuss the effectiveness of two main components, namely,

Calibrating the local client deviation (Line 9 in Algorithm 1) migrates the data heterogeneity;

Estimating the global reference orientation (Line 14 in Algorithm 1) accelerates the training process.
Algorithm 1: Federated Accelerating Gradient Calibration (FedAgrac).

Require: Initialize $M$ clients, set the initial model to be $x_1 \in \mathbb{R}^d$. Set $\nu^{(i)}$ and $\nu$ for all clients $i \in \{1, \ldots, M\}$. Set learning rate $\eta > 0$, calibration rate $\lambda > 0$, the number of global synchronizations $T$ and the number of local iterations of each client $K_i$ for all clients $i \in \{1, \ldots, M\}$.

1. On client $i \in \{1, \ldots, M\}$:
2. for $t = 1$ to $T$ do
3. Pull $\tilde{x}_t, \nu$ from server
4. Set $x_t^{(i)} = \tilde{x}_t$
5. Set $c = \nu - \nu^{(i)}$
6. for $k = 0$ to $K_i - 1$ do
7. Randomly sample a realization $\varepsilon^{(i)}_k$ from $\mathcal{D}_i$
8. $g^{(i)}_{t,k} = \nabla f_i(x_t^{(i)}; \varepsilon^{(i)}_k)$
9. $x_{t+1}^{(i)} = x_t^{(i)} - \eta (g^{(i)}_{t,k} + \lambda c)$
10. end for
11. Set $\nu^{(i)} = \frac{1}{K_i} \sum_{k=0}^{K_i-1} g^{(i)}_{t,k}$
12. Push $x_t^{(i)}; K_i$ to the server
13. Receive $\tilde{K}$ from the server
14. if $K_i \leq \tilde{K}$, then send $\nu^{(i)}$; else send $g^{(i)}_{t,0}$
15. end for
16. On server:
17. for $t = 1$ to $T$ do
18. Pull $\tilde{x}_t, \nu$ to clients
19. Pull $x_t^{(i)}; K_i$ from client $i \in \{1, \ldots, M\}$
20. $\tilde{x}_{t+1} = \sum_{i=1}^{M} \omega_i x_t^{(i)}$
21. $\bar{K} = \sum_{i=1}^{M} \omega_i K_i$
22. Push $\bar{K}$ to clients and receive $\nu_{\text{transit}}^{(i)}$ from clients
23. $\nu = \sum_{i=1}^{M} \omega_i \nu_{\text{transit}}^{(i)}$
24. end for

A. Calibrating the Local Client Deviation

As a classical approach, FedAvg updates the parameters using stochastic gradient descent (SGD), where the gradient is computed in accordance with Line 8. Suppose the gradient is equivalent to the first order derivative of the true local objective, i.e., $\nabla f_i(v, \varepsilon) = \nabla F_i(v)$, where $\varepsilon$ is randomly sampled from the local dataset $\mathcal{D}_i$. Then, given the stepsize $\eta$, the model update follows $x_{t+1}^{(i)} = x_t^{(i)} - \eta \nabla F_i(x_t^{(i)})$. Previous works [52], [53] show that this scheme can converge to a stable point when a client performs sufficient local updates. Under a heterogeneous data setting, the points vary among clients because each of them is determined by the local data distribution. Therefore, clients are biased from the global orientation, and the phenomenon is known as client deviation.\(^1\)

Existing works to overcome client deviation mainly focus on the variance reduction approach, i.e., SCAFFOLD [11]. It is somewhat similar to our proposed algorithm when $\lambda$ is set to 1. In this case, client $i$’s local reference direction $\nu^{(i)}$ is assumed to be equivalent to the vector from the current point to its local optimizer. As for the global reference orientation, $\nu$ overlaps with the gradient from the current point to the global minimizer. However, it is nearly impossible to coincide with the case, especially when applied with the gradient calibration technique. Generally speaking, using an obsolete gradient to predict the coming gradient is not reasonable because the aggregated direction presumably deviates from the expected one.

Therefore, we introduce a calibration rate $\lambda$ for the correction term. With this hyperparameter, a gradient can be adjusted and approximated to the global update. Empirical results in Fig. 2 intuitively present the effectiveness of $\lambda$. Generally speaking, a smaller $\lambda$ has a similar performance as FedAvg because the calibrated gradient is still biased to the local computed one. For a greater $\lambda$, the test accuracy goes down dramatically since the gradient is over-calibrated. As a result, a constant $\lambda$ cannot be too large or too small such that the calibration term is effective. Furthermore, in Fig. 2(b), we evaluate a case where $\lambda$ increases over time. Apparently, the strategy is impressive because it outperforms all other constant settings. The reason for the improvement is clear: at the beginning stage, the difference between two successive updates is significant because the model is far away from convergence. When the training comes to a stable point, the value of $\lambda$ should be 1 such that the gradient eliminates the deviation towards the local minimizer.

B. Estimating the Global Reference Orientation

While applying SCAFFOLD [11] to train a model, we notice that the model update is biased to the fastest node under step-asynchronous settings. Given a model $x$, some clients, e.g., client $i$, are close to a stable point such that the computed local reference orientation significantly deviates from the expected one, i.e., $\nabla F_i(x)$. Regarding that the clients (client $i$) with fewer local updates can better estimate the local orientation $\nabla F_i(x)$, the model prefers those with more local updates, which undermines the convergence property.

---

\(^1\)Client deviation is also known as client drift [11], [47].
SCAFFOLD requires all nodes to outperform another three potential approaches under \( \nabla F_{\text{FedAGr}} \) to all clients. To obtain an exact result of \( \nabla F_i^t(\hat{x}_i) \), each client \( i \in \{1, \ldots, M\} \) should provide an accurate estimation for \( \nabla F_i^t(\hat{x}_i) \), or the bias of the estimation \( \nu(i) \) can be eliminated by the sum, i.e., \( \sum_{i=1}^M \omega_i \nu(i) \). Therefore, there are two practical ways to estimate \( \nabla F_i^t(\hat{x}_i) \) for client \( i \), namely, (i) the first stochastic gradient, i.e., \( \nabla f_i(\hat{x}_i, \varepsilon) \), and (ii) the averaged stochastic gradient, i.e., \( \frac{1}{K} \sum_{k=0}^{K-1} \nabla f_i(\hat{x}_{i,k}; \varepsilon(i)) \) in Line 11 of Algorithm 1. Based on these two strategies, we design and empirically evaluate four different schemes to find a proper estimation for the global reference orientation: (Note: faster or slower nodes are classified by whether the number of local updates is greater than the average updates)

- **\textit{FedAGr}** requires faster nodes to transmit the first stochastic gradient while the rest push the average one; 
- **\textit{FedAGr_avg}** (a.k.a. SCAFFOLD) requires all nodes to transmit the average stochastic gradient; 
- **\textit{FedAGr_first}** requires all nodes to transmit the first stochastic gradient; 
- **\textit{FedAGr_reverse}** requires faster nodes to transmit the average stochastic gradient while the rest push the first one.

Fig. 3 presents the results of different strategies. As we can see, without step asynchronism, these four schemes do not have considerable differences. However, with step asynchronism, \textit{FedAGr} outperforms another three potential approaches under both convex and non-convex objectives. This is why Line 14 of Algorithm 1 is introduced. To further reduce the communication overhead, the algorithm solely requests the faster nodes to upload the first stochastic gradient, while the rest can be computed via

\[
\frac{1}{\eta K_i} (\hat{x}_i - \hat{x}_{i,k}) - \lambda (\nu - \nu(i)) \text{ if } \nu(i) \text{ is preserved on the server.}
\]

\[
\nu = \sum_{i,K_i \leq K} \frac{\omega_i}{K} \sum_{k=0}^{K_i-1} g_{i,k} + \sum_{i,K_i > K} \omega_i g_{i,0}
\]

Recursion Function. According to Line 9 in Algorithm 1, for client \( i \), the recursion between two successive local updates can be presented as:

\[
x_{i,k+1} = x_{i,k} - \eta \left[ g_{i,k} + \lambda \left( \nu - \nu(i) \right) \right]
\]

Then, based on the equation above, i.e., (5), for client \( i \) with the local updates of \( K_i \), \( x_{i,K_i} - \hat{x}_i \) can be formulated in mathematical expression as:

\[
x_{i,K_i} - \hat{x}_i = \sum_{k=0}^{K_i-1} \left( x_{i,k+1} - x_{i,k} \right)
\]

\[
= -\eta \sum_{k=0}^{K_i-1} g_{i,k} - \eta \lambda K_i \left( \nu - \nu(i) \right)
\]

Finally, according to the definition in (1), the recursion function between two successive global updates is the weighted average of all clients’ models, which is written as:

\[
\hat{x}_{i+1} - \hat{x}_i = \sum_{i=1}^M \omega_i x_{i,K_i} - \hat{x}_i
\]

\[
= -\eta \sum_{i=1}^M \sum_{k=0}^{K_i-1} \omega_i g_{i,k} - \eta \lambda K \nu + \eta \lambda \sum_{i=1}^M \omega_i K_i \nu(i)
\]

V. THEORETICAL CONVERGENCE ANALYSIS

In this section, we analyze the convergence property of \textit{FedAGr} under both non-convex objectives and strongly-convex objectives for solving Problem (1). The details of the mathematical proof are provided in the supplementary materials with step-by-step explanations, available online.

A. Mathematical Expression for Algorithm 1

In Section IV, we describe the details in Algorithm 1. Below represents how to derive the recursive function step by step.

Local Reference Orientation. To ensure every local update can calibrate to the expected one, we should use the averaged local update such that after multiple local updates, the acquired model does not deviate from the expected orientation. Therefore, the local reference orientation is defined as:

\[
\nu(i) = \begin{cases} 
\frac{1}{K} \sum_{k=0}^{K-1} g_{i,k} & K_i \leq K \\
\nu_{i,0} & \text{Otherwise}
\end{cases}
\]

Global Reference Orientation. SCAFFOLD [11] presents a remarkable performance with the aggregation of \( \nu(i) \) for all \( i \in \{1, \ldots, M\} \). However, the approach presumably does not work due to step asynchronism, where local reference orientations deviated from the expected direction are dramatically various among clients. To avoid this issue, we let the faster node with more number of local updates transfer the initial gradient while others send the local reference orientation to the server, which can be formally written as:

\[
\nu = \sum_{i,K_i \leq K} \frac{\omega_i}{K} \sum_{k=0}^{K_i-1} g_{i,k} + \sum_{i,K_i > K} \omega_i g_{i,0}
\]

...
B. Non-Convex Objectives

Theorem 2 (Non-convex objectives). Considering the same $x_1$ and $x_2$ as Theorem 1, under Assumption 1, 3, and 4, by setting
\[ \eta = O\left(\frac{M}{TK}\right), \]
the convergence rate of Algorithm 1 with step asynchronism for non-convex objectives is

\[
\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\|\nabla F(\tilde{x}_t)\|_2^2 \\
\leq O\left(\frac{F(x_0)}{\lambda KMT}\right) + O\left(\frac{\sigma^2 L}{\lambda K^2 T} \sum_{i=1}^{M} \omega_i K_i^2\right) \\
+ O\left(\frac{L^2 \sigma^2 M}{\lambda K^2 T} \sum_{i=1}^{M} \omega_i K_i^2\right) \\
+ O\left(\frac{L^2 \sigma^2 \lambda K^2}{K^2 T} \sum_{i=1}^{M} \omega_i K_i^2 \left(K_i \sum_{j=1}^{M} \frac{\omega_j}{K_j} + 1\right)\right). \tag{6}
\]

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

Corollary 2.1. By setting $\omega_1 = \ldots = \omega_M = 1/M$ and $\lambda = O(1)$, the following inequality holds under Theorem 2:

\[
\min_{t \in \{1, \ldots, T\}} \mathbb{E}\|\nabla F(\tilde{x}_t)\|_2^2 \leq O\left(\frac{1}{\sqrt{KT^2}}\right). \tag{7}
\]

Remark. [10] states that FedNova can achieve the convergence rate same as (7), but there exists an explicit condition that $\sum_{i=1}^{M} (K_i/M K_i)$ is a constant when $\omega_1 = \ldots = \omega_M = 1/M$. Let us consider an extreme case that the slow nodes locally update once, i.e., $K_i = 1$ for all $i \in \{1, \ldots, M-1\}$ while Client $M$ can run for a very large number of times. This case is possible, for instance, a system consists of multiple Raspberry Pi and a single Nvidia GTX 3080Ti GPU, the computational difference between which can be up to a thousandfold. Under such situation, the aforementioned term should be bounded by $O(K_i)$ instead of $O(1)$ and therefore, the convergence rate for FedNova should be $O(\sqrt{K/i}/MT)$. In comparison with (7), Fedagra supports the homogeneous environment.

C. Strongly-Convex Objectives

Theorem 3 (Strongly-Convex Objectives). Considering the same $x_1$ and $x_2$ as Theorem 1, under Assumption 1, 2, and 3, by setting $\lambda = 1$, $\eta = O(1/\mu LTK) \leq 1/LK$, the convergence rate of Algorithm 1 with step asynchronism for strongly-convex objectives is

\[
\mathbb{E}[F(\tilde{x}_T)] - F(x_\ast) \\
\leq O\left(\frac{\mu \|x_1 - x_\ast\|_2^2 \exp\left(-\frac{\mu T}{L}\right)}{\mu T} + \frac{H + P}{\mu^2 T^2}\right), \tag{8}
\]

where

\[
H = \frac{\sigma^2}{K^2} \sum_{i=1}^{M} \omega_i^2 \left(K_i + \frac{(K_i - K_j)^2}{K_j}\right),
\]

\[
P = \frac{L^2 \sigma^2}{\mu} \left(\sum_{i=1}^{M} \omega_i K_i^3\right) \sum_{j=1}^{M} \frac{\omega_j^2}{K_j^2}.
\]

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

Corollary 3.1. By setting $\omega_1 = \ldots = \omega_M = 1/M$, the following inequality holds under Theorem 3

\[
\mathbb{E}[F(\tilde{x}_T)] - F(x_\ast) \leq O\left(\frac{\sigma^2}{\mu^2 T}\right). \tag{9}
\]

Remark. Compared to FedNova [10] that has convergence theory only for non-convex objectives, we have established the rigorous convergence theory for our method Fedagra on strongly-convex objectives. Compared with Theorem 1, Fedagra not only converges to the optimal parameters, but also obtains a better convergence rate as $O(1/K) \leq O(1/K_{\min})$.

VI. EMPIRICAL EVALUATION

In this section, we conduct extensive experiments to evaluate the performance of Fedagra in the real cases that are widely accepted by the existing studies. To further obtain an intuitive understanding of the numerical results, Fedagra competes against other up-to-date benchmarks that are comparable under various settings. The code is implemented with PyTorch and available at https://github.com/HarliWu/Fedagra.

A. Setup

Datasets. We leverage Fashion-MNIST [9] to run the preliminary experiments in the previous sections. This dataset comprises 60000 $28 \times 28$ grey-scale training images and 10000 test images, which can be categorized into ten classes related to the clothes type. In this section, we utilize two more datasets: a9a2 and CIFAR-10 [54]. As a binary classification task, a9a consists of 32561 training samples and 16281 test samples, and each sample possesses 123 features. CIFAR-10 is a 10-category image classification task, constituting 60000 $32 \times 32$ color images divided into the training and test set with the size of 50000 and 10000, respectively.

Models. For the assessment of convex objectives, we train a logistic regression (LR) model using a9a. In addition, we investigate the performance under non-convex objectives through an image classification task CIFAR-10 [54] with AlexNet [55] and VGG-19 [56], deep neural networks with total parameters of 7.21 M and 20.55 M, respectively. As for Fashion-MNIST,
2-layer CNN and LR are utilized to evaluate the performance under non-convex and convex objectives, respectively. Based on the dataset used, the details for 2-layer CNN, AlexNet and VGG-19 are separately described in Table III, Table IV and Table V.

Data Heterogeneity. As for the non-i.i.d. settings, we adopt two different partitioned ways. The first one that we split the dataset across the clients follows the Dirichlet distribution with parameter 0.3, denoted as DP1. This approach is suitable for both datasets. The other method disjoint the dataset via sharding, and thus each client holds 5 classes. We let such a method be DP2 and ensure clients carry the same volume of data. It is worth noting that this partition is only compatible with CIFAR-10 because there exists among workers follow the Gaussian distribution. Then, considering the number of local updates varies among clients and follows the normal distribution with predefined mean and variance. And the number of local updates may change over time for each client.

Implementation and Hyperparameter Settings. The experiments are conducted with an MPI-supported cluster with the configurations of 100 GB RAM, 25 CPU cores, and 1 Nvidia P100 GPU. Based on the resource, we utilize 20 cores to act as clients and a single core as the federated server. Besides, the batch sizes throughout our experiments are set as 25 and 20 for CIFAR-10 and a9a, respectively. We choose FedAvg [6], FedNova [10], SCAFFOLD [11] and FedProx [48] as benchmarks and present the effectiveness of our proposed approach FedGrac. For a fair comparison, we compare these algorithms with the results when they achieve the best performance under the constant learning rates \{0.01, 0.008, 0.005\} and \{0.005, 0.001, 0.0005\} for AlexNet/VGG-19 and LR, respectively. And other required hyperparameters are also carefully picked from a set, such as the coefficient of the regularization term for FedProx in \{1, 0.1, 0.01\}. We specified other unmentioned but necessary settings in the captions of the figures and the tables.

B. Numerical Results

Performance under Various combinations for learning rate and calibration rate. As learning rate \(\eta\) and calibration rate \(\lambda\) need tuning in FedGrac, we first explore how to set both hyperparameters scientifically. Fig. 4 depicts the test accuracy under various relations between \(\eta\) and \(\lambda\). As we observe, the differences regarding the convexity are quite significant, e.g., AlexNet in Fig. 4(a) and LR in Fig. 4(b), while the computation heterogeneity has minor influence on the selection of hyperparameters under the same model, e.g., AlexNet in Fig. 4(a) and (c). Based on the acquired results, we discuss how to set the hyperparameters for FedGrac under convex or non-convex objectives.

Both Fig. 4(a) and (c) illustrate the performance under AlexNet with and without computational heterogeneity. In both cases, most \(\lambda\)s achieve the highest accuracy at \(\eta = 0.05\), while some have the best performance at \(\eta = 0.01\). When the learning rate initializes with a value smaller or equal to 0.001, most AlexNets seem untrained after 100 rounds because they are less likely to escape a saddle point. Although some portfolios successfully get out of the minima, they still cannot outperform the aforementioned settings because they may (i) trap into a non-optimal stable point or (ii) need a longer period to reach the optimal solution. A constant \(\lambda\) that performs well in all learning rates does not exist. However, when we shrink the choice of learning rate between 0.01 and 0.05, \(\lambda = 0.05\) has a remarkable performance. In our experiments, the calibration rate is chosen from \{0.01, \ldots, 0.05\} depending on the algorithm’s performance.

Fig. 4(b) and (d) present the results under the convex objectives. Regardless of the step asynchronism, \(\lambda = 1\) always has remarkable performance for any learning rate. And it is noticeable that FedGrac can obtain the best performance when \(\lambda = 1\) and \(\eta = 0.005\). As for a \(\lambda \neq 1\), FedGrac can achieve better performance as the learning rate becomes smaller.

With such a phenomenon, we hypothesize that FedGrac cannot exactly reach the identical minimizer when \(\lambda \neq 1\) and approaches the expected point as the learning rate reduces.
Fig. 4. Comparison of various setting combinations for learning rate $\eta$ and calibration rate $\lambda$ using DP1 data distribution under AlexNet and LR after 100 communication rounds. The horizontal index indicates the value of $\lambda$ while the vertical index shows the value of $\eta$. The numeric in the box presents the averaged test accuracy of the last 10 rounds under the specific hyperparameter settings. The mean number of local updates is 500, and the variance with step asynchronism is 10000.

### TABLE VI

| Model   | Data Distribution | Target Accuracy | Variance | Mode | Number of communication rounds (%) |
|---------|-------------------|-----------------|----------|------|------------------------------------|
|         |                   |                 |          |      | FedAvg | FedNova | SCAFFOLD | FedProx |
| AlexNet | DP1               | 66%             | V = 0    | fixed | 113    | 123    | 127      | 113     | 145    |
|         |                   |                 | V = 100  | random | 106    | 130    | 147      | 114     | 144    |
|         |                   |                 | V = 1000 | random | 118    | 136    | 154      | 135     | 140    |
| AlexNet | DP2               | 70%             | V = 0    | fixed | 132    | 177    | 179      | 141     | 152    |
|         |                   |                 | V = 100  | random | 140    | 183    | 184      | 140     | 147    |
|         |                   |                 | V = 1000 | random | 138    | 195    | 195      | 137     | 141    |
| VGG-19  | DP2               | 80%             | V = 0    | fixed | 73     | 83     | 79       | 72      | 90     |
|         |                   |                 | V = 100  | fixed | 73     | 75     | 72       | 66      | 72     |
|         |                   |                 | V = 1000 | fixed | 77     | 85     | 74       | 78      | 102    |

Performance under various data distributions. Table VI validates our algorithm under different data heterogeneities, i.e., DP1 and DP2 under AlexNet. The target accuracy is determined by the best performance that these five algorithms can achieve when they run a constant number of updates. By comparing each algorithm under these two data distributions, DP2 is more challenging for FedAvg and FedNova because the algorithms generally require more communication rounds to achieve the target. Even worse, these two algorithms cannot achieve the goal within 200 rounds in some DP2 settings. As for the regularization-based approach (i.e., FedProx) and the variance reduction approaches (i.e., FedaGrac and SCAFFOLD), the task shifting does not cause a distinct influence in terms of the required communication rounds. As we see in both cases with computational differences, FedaGrac demonstrates the superiority over other benchmarks.

Performance under various neural networks. In addition to exploring various data distributions based on Table VI, we investigate the performance of FedaGrac under different neural networks. As we notice, the approach in VGG-19 does not outperform all benchmarks in some computation heterogeneity cases. Specifically, it requires several more rounds than the best algorithm. An explanation for this phenomenon is that obtaining an 80%-accuracy VGG-19 on CIFAR-10 is not a difficult task. In contrast to getting an AlexNet with a test accuracy of 70%, the algorithms can adopt a greater learning rate to improve training efficiency. Since there are some restricted terms in FedaGrac, it is reasonable that our proposed algorithm cannot outperform the benchmarks. Meanwhile, it is common that some benchmarks cannot outperform FedAvg [57]. However, it is worth noting that, as presented in Fig. 5, the faster algorithm may not surpass the slower ones in terms of the final test accuracy.

Performance under various computational capabilities. While adopting Gaussian distribution to tune the computation heterogeneity, we should manually set both mean and variance. To explore whether these two hyperparameters influence the algorithms' performances, we conduct extensive experiments, and the relevant results are presented in Table VI and Fig. 5. Table VI evaluates the performance under the computational capabilities with a constant mean of 500 and different variances.

3The difference between the numbers of the communication rounds is less than 15%.
while Fig. 5 assesses the convergence tendency under a fixed variance of 10000 and diverse means.

Table VI presents the results given different variances with/without time-varying local updates. Our analysis is mainly based on AlexNet because it gives noticeable differences when the variances or the modes switch. Admittedly, these algorithms are not sensitive to whether the number of local updates is time-varying. However, it is worth noting that FedNova is vulnerable to time-varying settings in DP2. This is because a large learning rate may lead FedNova to a surrogate solution [47] such that FedNova has adopted a smaller learning rate to achieve the target accuracy. In contrast to time-varying local updates, the variance plays an important role in training efficiency. When the variance becomes larger, it is likely that the algorithms require more communication rounds. Nevertheless, a greater variance sometimes improves the training efficiency of those algorithms which mitigate the client-drift effect, i.e., FedaGrac, SCAFFOLD, and FedProx.

Fig. 5 illustrates the entire training progress, i.e., the test accuracy with respect to the training time and the communication rounds under both convex and non-convex objectives. Our proposed algorithm achieves competitive accuracy compared to other baselines, despite a slow start likely taking place because the calibration is yet to settle the client-drift effects properly in the beginning. Although FedAvg and FedNova require half communication overhead as our proposed algorithm does, they cannot keep dominant alongside the training. Use AlexNet as an example (Fig. 5(a) and (b)), and FedaGrac is capable of achieving the same performance with fewer rounds. In addition, it is interesting to see FedProx consuming more time to implement 200 rounds than FedAvg. A reasonable explanation for this phenomenon is that extra computation is required by the regularization terms. As the model gets larger, this effect becomes minor since the communication consumption asymptotically occupies most training time (compare between LR (Fig. 5(c)) and AlexNet (Fig. 5(a)) for this heuristic conclusion). As a convex objective, LR depicts the issue of objective inconsistency (the latter two plots in Fig. 5). The performances of FedAvg, FedNova, and FedProx are much worse than FedaGrac and SCAFFOLD. With the increasing mean and the unchanged variance, the deterioration gets mitigation but cannot eliminate. As for the comparison between SCAFFOLD and our proposed method, the latter possesses dominance nearly all the time.

VII. CONCLUSION

This paper introduces a new algorithm named FedaGrac to tackle the challenges of both statistical heterogeneity and computation heterogeneity in FL. By calibrating the local client deviations according to an estimated global orientation in each communication round, the negative effect of step asynchronism on model accuracy can be greatly mitigated, and the training process is remarkably accelerated. We establish the theoretical convergence rate of FedaGrac. The results imply that FedaGrac admits a faster convergence rate and has a better tolerance to computation heterogeneity than the state-of-the-art approaches. Extensive experiments are also conducted to validate the advantages of FedaGrac.

ACKNOWLEDGMENTS

The authors would like to thank Shiqi He for the useful discussion and the anonymous reviewers for their constructive comments.

REFERENCES

[1] Y. Liu et al., “FedVision: An online visual object detection platform powered by federated learning,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 13172–13179.
[2] P. Yu and Y. Liu, “Federated object detection: Optimizing object detection model with federated learning,” in Proc. 3rd Int. Conf. Vis., Image Signal Process., 2019, pp. 1–6.
[3] F. Liu, X. Wu, S. Ge, W. Fan, and Y. Zou, “Federated learning for vision-and-language grounding problems,” in Proc. 3rd Int. Conf. Vis., Image Signal Process., 2020, pp. 11572–11579.
[4] X. Wu, Z. Liang, and J. Wang, “FedMed: A federated learning framework for language modeling,” Sensors, vol. 20, no. 14, 2020, Art. no. 4048.
[5] M. Chen et al., “Federated learning of n-gram language models,” 2019, arXiv:1910.03432.
[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Artif. Intell. Statist., 2017, pp. 1273–1282.
[7] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, “On the convergence of fedavg on non-iid data,” 2019, arXiv:1907.02189.
[8] Z. Chai et al., “Towards taming the resource and data heterogeneity in federated learning,” in Proc. USENIX Conf. Oper. Mach. Learn., 2019, pp. 19–21.
[9] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” 2017, arXiv:1708.07747.
[10] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, “Tackling the objective inconsistency problem in heterogeneous federated optimization,” in Proc. Adv. Neural Inf. Process. Syst., 2020, pp. 7611–7623.
Song Guo (Fellow, IEEE) is a full professor leading the Research Group of Networking and Mobile Computing with the Department of Computing, The Hong Kong Polytechnic University. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. Currently, he is the editor-in-chief of IEEE Open Journal of the Computer Society and chair of IEEE Communications Society (ComSoc) Space and Satellite Communications Technical Committee (SSCTC). His research interests include edge AI, Big Data, machine learning, mobile computing, and distributed systems. He has published many papers in top venues with wide impact in these areas and was recognized as a highly cited researcher (Clarivate Web of Science). He is the recipient of more than a dozen Recognition and Best Paper Awards from IEEE/ACM conferences, journals and technical committees. His research has been sponsored by RGC, NSFC, ITF, MOST, NRC, JSPS, MIC, JST, industry, etc. He has been named on editorial board of a number of prestigious international journals, including IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Emerging Topics in Computing, IEEE Transactions on Green Communications and Networking, IEEE Communications Magazine, and Wireless Networks. He has served as the chair of the organizing and technical committees of numerous international conferences. He is a fellow of the Canadian Academy of Engineering (FCAE) and ACM distinguished member.

Haozhao Wang received the PhD degree from the Huazhong University of Science and Technology, in December 2021. He is working as a PostDoc with the School of Computer Science and Technology, Huazhong University of Science and Technology. He has worked as a research assistant with the Department of Computing, Hong Kong Polytechnic University from June 2018 to June 2021. His research interests include distributed machine learning and federated learning.

Haobo Zhang received the BS degree from the University of Electronic Science and Technology of China. He is currently working toward the PhD degree with the Computer Science and Engineering department, Michigan State University. His research interest lies in privacy-preserving machine learning.

Zhihao Qu (Member, IEEE) received the BS and PhD degrees in computer science from Nanjing University, Nanjing, China, in 2009, and 2018, respectively. He is currently an Associate Professor with the School of Computer and Information, Hohai University. His research interests are mainly in the areas of wireless networks, edge computing, and distributed machine learning.

Jie Zhang (Member, IEEE) received the BE and ME degrees from the School of Computer Science, China University of Geosciences, Wuhan, in 2015 and 2018, respectively, and the PhD degree from Hong Kong Polytechnic University, Hong Kong, in 2022. She is currently a postdoctoral fellow with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. Her current research interest includes edge computing, machine learning, deep reinforcement learning, federated learning and computer vision. She has served as program committee member or reviewer for several prestigious conferences and journals such as CVPR, AAAI, INFOCOM, PerCom, IEEE Internet of Things Journal, IEEE Transactions on Cloud Computing, IEEE Transactions on Vehicular Technology, IEEE Open Journal of the Computer Society, etc.

Ziming Liu received the master’s degree in aerospace engineering from Beihang University, Beijing, China, in 2020. He is currently working toward the PhD degree with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China. His main research fields are federated learning and machine learning.