GIFT: Toward Accurate and Efficient Federated Learning With Gradient-Instructed Frequency Tuning

Chen Chen, *Member, IEEE*, Hong Xu, *Senior Member, IEEE*, Wei Wang, *Member, IEEE*, Baochun Li, *Fellow, IEEE*, Bo Li, *Fellow, IEEE*, Li Chen, *Member, IEEE*, and Gong Zhang, *Member, IEEE*

***Abstract*— Federated learning (FL) enables distributed clients to collectively train a global model without revealing their private data, and for efficiency clients synchronize their gradients *periodically*. However, this can lead to the inaccuracy in model convergence due to inconsistent data distributions among clients. In this work, we find that there is a strong correlation between FL accuracy loss and the synchronization frequency, and seek to fine tune the synchronization frequency at training runtime to make FL accurate and also efficient. Specifically, aware that under the FL privacy requirement only gradients can be utilized for making frequency tuning decisions, we propose a novel metric called *gradient consistency*, which can effectively reflect the training status despite the instability of realistic FL scenarios. We further devise a feedback-driven algorithm called Gradient-Instructed Frequency Tuning (GIFT), which adaptively increases or decreases the synchronization frequency based on the gradient consistency metric. We have implemented GIFT in PyTorch, and large-scale evaluations show that it can improve FL accuracy by up to 10.7% with a time reduction of 58.1%.**

***Index Terms*— Federated learning, synchronization frequency, gradient statistics.**

Manuscript received 17 April 2022; revised 6 September 2022; accepted 30 November 2022. Date of publication 6 February 2023; date of current version 17 March 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62202300; in part by the Shanghai Pujiang Program under Grant 22PJ1404600; in part by RGC RIF Grant under Contract R6021-20; in part by RGC GRF Grants under Contract 16209120, Contract 16200221, Contract 16213120, Contract 16202121, and Contract 11209520; and in part by CUHK under Contract 4937007, Contract 4937008, Contract 5501329, and Contract 5501517. *(Corresponding author: Bo Li.)*

Chen Chen is with the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: chen-chen@sjtu.edu.cn).

Hong Xu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: hongxu@cuhk.edu.hk).

Wei Wang and Bo Li are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong (e-mail: weiwa@cse.ust.hk; bli@cse.ust.hk).

Baochun Li is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: bli@ece.toronto.edu).

Li Chen is with the Zhongguancun Laboratory, Beijing 102206, China (e-mail: lichen@zgclab.edu.cn).

Gong Zhang is with the Theory Laboratory, Huawei Hong Kong Research Center, Hong Kong (e-mail: nicholas.zhang@huawei.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSAC.2023.3242721.

Digital Object Identifier 10.1109/JSAC.2023.3242721

I. Introduction

FEDERATED learning (FL) [1], [2] emerges as a popular paradigm that allows edge clients to collaboratively train models without sharing their local private data. In typical FL scenarios, the hardware resources on edge devices—especially the network bandwidth—are usually quite limited. To reduce the overall training cost, the *de facto* FL mechanism is FedAvg, under which each client trains with its local dataset for *multiple* iterations before performing a synchronization.

A well-known challenge for FL is that the local datasets on clients are *not identically and independently distributed* (i.e., being non-IID). Recent works have empirically shown that, FL with non-IID datasets would suffer remarkable accuracy loss [2], [3], [4], [5], which correlates with the synchronization frequency of FedAvg (also confirmed by us in §III). There is thus a trade-off in setting up the FL synchronization frequency: Less frequent synchronization can reduce the communication cost, but in the meantime would compromise the accuracy performance. To make FL accurate and also efficient, it is a promising technique to dynamically adjust the synchronization frequency during the training process.

Yet, it remains a largely-unexplored territory how to tune the FL synchronization frequency at runtime: For FedAvg and its cluster counterpart called local SGD [6], [7], [8], [9], existing practices either assume a fixed frequency (which may be either too high or too low as training proceeds), or consider IID data only. In particular, we note that the distinct FL characteristics impose three challenges for the design of an ideal frequency tuning strategy. First, privacy is a primary concern for FL, meaning that the frequency tuning decisions shall be made only with gradients, not requiring additional information like client loss or accuracy. Second, that frequency tuning strategy shall work smoothly when facing dynamic and massive client participations in realistic FL systems. Third, since a FL framework may be used for many machine learning models, the frequency tuning strategy shall be generally effective, not relying on any specific model characteristics (i.e., knowledge of the convexity or smoothness constant).

To quickly recap, our objective in this work is to design a *privacy-preserving*, *system-practical* and *model-generic* method to tune FL synchronization frequency at runtime.

To make that, we propose *Gradient-Instructed Frequency Tuning (GIFT)*—a feedback-driven frequency tuning algorithm with gradient-based feedback signal. GIFT is tailor-made respectively for the three challenges aforementioned.

To be privacy-preserving, GIFT perceives the training status based purely on gradients. To bridge the gap between training performance and gradient behavior, we conduct both theoretical and experimental analysis on the root cause of accuracy degradation when training models with non-IID data. We find that, when insufficient synchronization occurs with inconsistent data distribution, the global model parameters would stagnate prematurely at sub-optimal positions, where the model gradients from different clients conflict with each other, i.e., exhibiting a *bifurcating* trend. Therefore, through the level of such gradient bifurcation, we can gauge the instantaneous FL training status without privacy violation.

To be system-practical, the GIFT signal to quantify the gradient bifurcation level must work well affronting the system challenges in realistic FL scenarios. That is, it should be robust to dynamic client participation and mini-batch randomness, and shall also be resource-efficient despite a vast number of participating clients. We adopt a *smoothing* method to address the system and mini-batch instability, and adopt a *pooling* method to compress the computation and memory consumption when handling massive client participation. Atop those methods, we propose a novel metric called *gradient consistency*, which would decrease towards zero when the gradient bifurcation level gradually amplifies. It then works as a feedback signal to instruct frequency tuning in FL.

Finally, to be model-generic, instead of deriving a subtle formula linking the ideal synchronization frequency to the gradient consistency metric (which requires rigid but risky model property assumptions), in GIFT we adopt a feedback-driven frequency-tuning heuristic: Once gradient consistency stabilizes around zero (indicating premature training stagnation), we increase the synchronization frequency by a fixed factor; such a process repeats when the gradient consistency stabilizes again under the new frequency. We further extend GIFT to incorporate modest frequency relaxation in phases where frequent synchronization is not necessary (e.g. at the FL commencement).

We have implemented GIFT with PyTorch and evaluated its performance in a 100-node Amazon EC2 cluster emulating real-world FL setups. In addition to the privacy-preserving benefit, our evaluations confirm that GIFT is a practical and general algorithm that can substantially improve the FL performance on both accuracy and resource efficiency: It can improve the convergence accuracy of VGG-16 by 10.7% with a time reduction of 58.1% (after the same amount of training rounds); meanwhile, given a fixed accuracy target, GIFT can save the training time by 28.9% when compared with existing methods.

II. Background and Motivation

A. A Primer on Federated Learning

Machine learning models are increasingly trained with a vast amount of data samples under the SGD (Stochastic Gradient Descent) algorithm [10], [11], [12]. In many real-world scenarios, the training samples are privacy-sensitive and dispersed on distributed clients like IoT devices, cellphones [2], [13], [14], [15]. To train models without centralizing such private data, an increasingly popular technique is Federated Learning (FL) [1], [2], under which each client locally refines the model parameters and communicates the updates to the central server.

1) FedAvg: Compared to computing servers in production clusters, FL clients like IoT devices or cellphones suffer remarkable bandwidth limitations. To reduce the communication cost, *FedAvg* [1], [2] has become the *de facto* FL mechanism, which dictates each client to perform *multiple* (denoted by τ) local iterations before synchronizing their accumulated updates. As a distinct hyper-parameter in FL, τ, or equivalently the synchronization frequency, critically affects the model training performance. With less frequent synchronization, the communication overhead for processing a given data amount can be reduced. Yet, local datasets on different FL clients are usually not independently and identically distributed (i.e., being non-IID), and less frequent synchronization would on the other hand compromise the model convergence accuracy. This has been empirically observed in many existing works [2], [3], [4], [5]. Given this trade-off, it is of urgent need to make FL accurate and also efficient by properly setting up the synchronization frequency.

B. Sync-Frequency Setup: Prior Arts and Their Limitations

While there do exist some related works [6], [7], [8], [9], [16], [17], [18] on setting up the synchronization frequency for distributed model training, we find that they either work within an over-narrow solution space (e.g., static frequency) or base their solutions on unrealistic assumptions (e.g, IID data).

For FL scenarios, some research works [16], [17], [18] have proposed to properly setup the synchronization frequency for better resource efficiency. As a representative example, Wang et al. [17] proposed Adaptive Federated Learning (hereafter called by AFL by us), which estimates the best FL synchronization frequency that can minimize the training loss under a given resource budget. It derives a sophisticated formula representing the best frequency, which involves the instantaneous loss value, the concrete loss function characteristics (Lipschitz parameter, smoothness parameter, gradient divergence bound) and the data distribution of each worker. Yet, including AFL, those works implicitly assume a *fixed frequency* without any runtime dynamicity. This unnecessarily narrows down the solution space by excluding the dynamic frequency-tuning strategies, and may thus suffer suboptimal performance (As shown later in §V, a fixed frequency may be either too high or too low for different training phases).

For cluster scenarios, a training method called local SGD (or periodical averaging) [6], [7], [8], [9] also allows workers to proceed for multiple local iterations before one global synchronization, and some works in this regard explored how to dynamically change the synchronization frequency. For example, AdaComm [7] formulated the best frequency that can minimize the training error after a given time budget; it further extended that static formula into a dynamic

strategy by dividing the training process into short intervals and estimating the best frequency in each interval. Nonetheless, those works did not consider non-IID data in their solution design, rendering their applicability in FL scenarios questionable.

1) Design Requirements: In particular, given the distinct FL characteristics, we notice that there exist three key requirements that any effective frequency tuning method under FL must comply with:

1) Privacy-Preserving Requirement. Data privacy is a primary concern for FL, and a key principle of FL is that only gradients (or equivalently the model parameters) can be collected from the clients [13], [14]; collecting additional information would increase the risk of privacy leakage. Prior works base their solutions mainly on mathematical formulas, and require client knowledge (e.g., loss values) as essential components in their formulas. The AFL work [17] even demands the information of local data distributions, which would severely impair the client privacy. To be privacy-preserving, we need to perceive the FL training status from gradients. In standard FL, model gradients are the default content collected from clients, and would be readily available at the FL server.

2) Practicality Requirement. A well-known system characteristic for FL is that clients may dynamically join or leave training at random time [13], [19]. Meanwhile, for commercial FL applications the number of clients may be quite large [20]. Existing methods ignored such stability and scalability challenges, and may suffer performance degradation as well as large maintenance cost.

3) Generality Requirement. A FL framework may serve various models without model-specific customization; hence, frequency tuning strategies in FL shall not rely on any model-specific characteristics. Yet, the aforementioned works build their solutions with unrealistic model assumptions. For example, AFL [17] derives its formulation by assuming convex loss functions, which does not hold for deep learning models; it also requires the smoothness constant, the Lipschitz-ness constant and the gradient variance bound, which are hard to obtain in reality. In fact, while such theoretical analysis can yield inspiring insights, given the analytical complexity of neural network models, it is however too risky to directly work out the exact FL synchronization frequency from them. To be model-generic, our solutions should be built not on end-to-end formulations but with a feedback control heuristic.

To summarize, in this work we seek to design a *privacy-preserving*, *system-practical* and *model-generic* frequency tuning strategy to make FL accurate and also efficient. The remaining part of this paper is organized as follows. In §III, we will analyze from the gradient perspective how FL performance is affected by synchronization frequency, and then describe a gradient pattern that can reflect the instantaneous training status. In §IV, we give the definition of our feedback signal, and elaborate our frequency tuning algorithm based on that signal. We further evaluate the effectiveness of our GIFT solution in §V. Finally, we introduce some additional related work in §VI and conclude in §VII.

TABLE I
SUMMARY OF MAIN NOTATIONS

N	Number of clients
D_i	Local dataset on client-i
$l(s,\omega)$	Loss value given sample s and parameter ω
$L_i(\omega)$	Local loss function on client-i
$L^\star(\omega)$	Global loss function
ω_k^i	Local model on client-i at iteration k
$\omega_k^\star$	Ideal model at iteration k if trained with IID data
$\omega^\star$	Converged model if trained with IID data
$\bar{\omega}^\star$	Converged model under realistic FL
τ	Synchronization frequency (sync once for τ iterations)
u_τ^i	Accumulated gradient over τ iterations on client-i
$u_\tau^\star$	Ideal value of u_τ^i if trained with IID data
e_τ^i	Local-error component of u_τ^i
β	Lipschitz constant in the smoothness assumption
C	Metric of gradient consistency
γ	Divisor to scale down τ
δ	Addend to increase τ
θ	EMA smoothing factor
$\tilde{P}_r$	EMA value of positive gradient component
$\tilde{N}_r$	EMA value of negative gradient component

III. SYNC-FREQUENCY AND FL PERFORMANCE: A GRADIENT POINT OF VIEW

In this section, we first theoretically and experimentally analyze the impact of synchronization frequency on FL performance, and then introduce an interesting phenomena called gradient bifurcation, which can help to reflect the instantaneous training status.

A. Impact of Synchronization Frequency

To properly setup the FL synchronization frequency, we first need to know the impact of a given frequency on the FL training performance.

While a series of research works [17], [18], [21], [22] have theoretically analyzed the FL convergence process, we find that they are not suitable for our needs. Those works in general follow a common methodology: first assume a constant bound for the gradient variance, and then derive a formula representing how fast the model parameters can converge. Nonetheless, fast convergence rate does not necessarily mean high accuracy; meanwhile, the gradient variance bound—as a highly idealized ground-truth knowledge—has little relationship with the instantaneous gradient behavior, thus yielding little help for understanding runtime FL status.

In our analysis, we primarily want to understand why there is an accuracy degradation when training models under FL with non-IID data: What is the key factors behind? And can we somehow probe the related training status with gradient-wise characteristics at runtime? We resort to the following theoretical analysis to find out the answers.

1) Symbol Description: In our FL setup, there are N clients each with a local dataset D_i ($i = 1, 2, \ldots, N$). Let $l(s,\omega)$ be the loss value when predicting sample s with parameter

ω, then the local loss function on client-i is $L^i(w) = \frac{1}{|D_i|}\sum_{s\in D_i} l(s,\omega)$ and the global loss function as the true optimization target is $L^\star(\omega) = \frac{1}{|\cup_{D_i}|}\sum_{s\in\cup_{D_i}} l(s,\omega)$. For simplicity we assume balanced data such that $L^\star(\omega) = \frac{1}{N}\sum_{i=1}^N L^i(\omega)$. Let ω_k^i be the local model on client-i after refined for k iterations from ω_0, and $\omega_k^\star$ be the ideal model when refined with a globally-shuffled IID dataset also for k iterations from ω_0. Table I has listed all the symbol notations adopted in this paper.

To facilitate our analysis of gradient behavior, we seek to decouple the local gradient into two components: a *global* component and a *local-error* component. To start with, we derive Lemma 1 based on the standard convexity[1] and smoothness assumptions.

Assumption 1 (Convexity): The loss function $L^\star(\omega)$ and local loss functions $L^i(\omega)$ ($i = 1, 2, \ldots, N$) are convex, i.e., $L(y) \geq L(x) + \nabla L(x)^T(y - x)$.

Assumption 2 (β-Smoothness): The loss function $L^\star(\omega)$ and each local loss function $L^i(\omega)$ ($i = 1, 2, \ldots, N$) is β-smooth, i.e., $\|\nabla L^i(x) - \nabla L^i(y)\| \leq \beta\|x - y\|$. It also means that $\nabla L^i(\omega)$ is β-Lipschitz, or equivalently $\|\nabla^2 L^i(\omega)\| \leq \beta$.

Lemma 1 (Local Gradient Composition): Let $u_\tau^i = \omega_\tau^i - \omega_0$ be the accumulated gradient[2] on client-i after τ iterations, then $u_\tau^i = u_\tau^\star + e_\tau^i$, where $u_\tau^\star$ is the ideal gradient attained with IID data representing the global component, and e_τ^i is the local-error component:

$$e_\tau^i = -\eta\sum_{k=0}^{\tau-1}[\nabla L^i(\omega_k^\star) - \nabla L^\star(\omega_k^\star) + \langle\nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle].$$

The proof of Lemma 1 can be found in Appendix. This lemma implies that the gradient error of a FL client is related to the gap between the local and global loss landscapes ($\nabla^2 L^i(\omega)$, i.e., the *hessian* matrix, or more straightforwardly, the *curvature*). Moreover, regarding the aggregated gradient after a synchronization round, with Lemma 1 we can further get the following Theorem:

Theorem 1 (Gradient Error After Synchronization): Let $\bar{u}_\tau = \frac{1}{N}\sum_{i=1}^N u_\tau^i$ and let $d(\tau) = \|\bar{u}_\tau - u_\tau^\star\|$ be aggregated gradient error after each client locally refines the parameter for τ iterations from ω_0, then

$$d(\tau) \geq (\tau - 1)\frac{\eta^2}{N}\|\sum_{i=1}^N\langle\nabla^2 L^i(\omega_0), \nabla L^i(\omega_0) - \nabla L^\star(\omega_0)\rangle\|.$$

[1]This convexity assumption does not invalidate our solution applicability for general models (which are mostly non-convex). Because our algorithm proposed later is based on qualitative properties (instead of quantitative derivations), and the qualitative behaviors around the local minima of non-convex models resemble that of convex ones.

[2]By *gradient*, we refer to the *accumulated update* over the entire round—with the learning rate η integrated in. In this sense, our analysis in this work is independent to the specific gradient generating scheme or learning rate scheme, and can thus be extended to other SGD variants like Adam [23] and AdaGrad [24]. Besides, for simplicity, in our analysis we ignore the impact of random sample selection within each mini-batch, and focus on the *expected* gradient of the local loss function. We will address mini-batch randomness later in §IV.

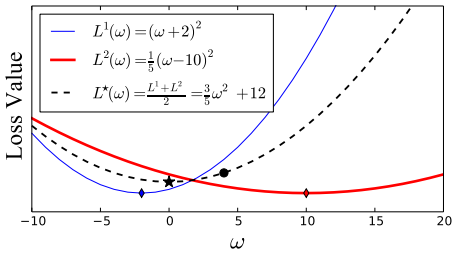


Fig. 1. During local iterations each FL client may already reach its local optimum (-2 and 10). With inaccurate gradient the parameter would be refined to 4 instead of the expected 0.

Theorem 1 shows that the error of an aggregated gradient is determined by two factors: first by the heterogeneity among clients' loss *curvatures* ($\{\nabla^2 L^i(\omega_0)\}$), and second by the synchronization frequency. In particular, since by definition $\sum_{i=1}^N[\nabla L^i(\omega_0) - \nabla L^\star(\omega_0)] = 0$, the item $\|\sum_{i=1}^N\langle\nabla^2 L^i(\omega_0), \nabla L^i(\omega_0) - \nabla L^\star(\omega_0)\rangle\|$ can be viewed as assigning different weights to a set of numbers whose sum is zero. If the weights $\{\nabla^2 L^i(\omega_0)\}$ are homogeneous across different clients, e.g., when training with IID data, then the aggregated gradient has no error; otherwise the error would augment with the heterogeneity level of $\{\nabla^2 L^i(\omega_0)\}$.

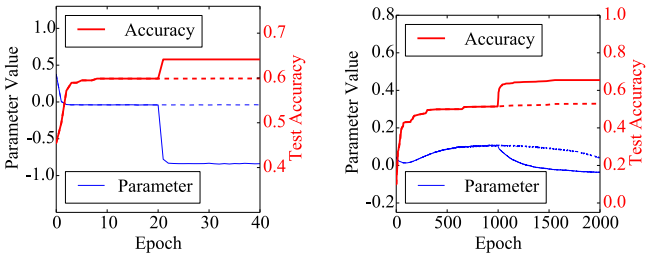
From the above theorem, we learn that the error would amplify with larger non-IID level or larger τ, consistent with the empirical results in existing works [2], [3], [4], [5]. Atop this per-round synchronization error, we can further derive Theorem 2 that depicts the model convergence status:

Theorem 2 (Suboptimal Convergence): Suppose FL process converges at parameter $\bar{\omega}^\star$, and $\omega^\star$ is the ideal parameter under IID dataset. Then

$$\|\bar{\omega}^\star - \omega^\star\| \geq \frac{(\tau-1)\eta}{\beta N}\|\sum_{i=1}^N\langle\nabla^2 L^i(\bar{\omega}^\star), \nabla L^i(\bar{\omega}^\star) - \nabla L^\star(\bar{\omega}^\star)\rangle\|.$$

This theorem shows that, for cases with heterogeneous $\{\nabla^2 L^i(\omega_0)\}$, the FL process would stagnate prematurely at a suboptimal position, with the error gap positively correlated to τ. Obviously, training around this position is a severe resource wastage with no accuracy benefit. To better understand the root cause of suboptimal stagnation from the gradient perspective, we resort to a toy example with quadratic loss functions.

Our example is shown in Fig. 1 where there are two clients with respective loss function $L^1(\omega) = (\omega + 2)^2$ and $L^2(\omega) = \frac{1}{5}(\omega - 10)^2$. The two loss functions exhibit different curvatures and have different local optima -2 and 10, emulating a non-IID setup. The global loss function to optimize is thus $L^\star(\omega) = \frac{1}{2}[L^1(\omega) + L^2(\omega)] = \frac{3}{5}\omega^2 + 12$, with the optimal parameter $\omega^\star$ be 0. During the local iterations, each client is essentially refining its parameter towards its local optimum; in the first iteration of a round, the local gradients if averaged can reflect the true optimal one, yet in later iterations, due to heterogeneous curvatures, gradients from different clients would decay in different rates, making the aggregated gradient less accurate. For example, if the synchronization is so late such that each client reaches its local optimum, the aggregated



(a) SVM (b) LeNet-5

Fig. 2. When training with two clients over non-IID data, by increasing the synchronization frequency (at epoch 20 for SVM and epoch 1000 for LeNet-5), the sampled global parameter can get closer to the true optimum, and the models can attain a higher accuracy. The dashed lines show the variables without frequency change.

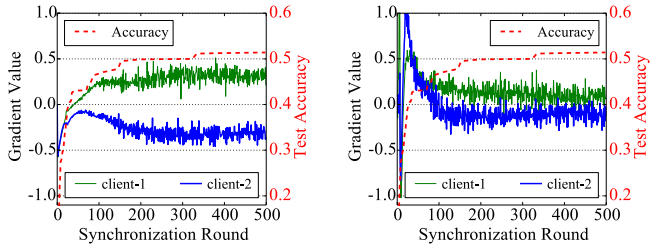


Fig. 3. When training LeNet-5 with 2 clients, the gradients of two randomly-chosen parameters gradually bifurcate.

gradient would point to the optimum average 4 instead of to the expected global optimum 0. The same phenomena would repeat in later rounds, meaning that the FL process stagnates at a suboptimal state. Thus, more frequent synchronization can reduce the negative impact of gradient curvature and make the aggregated gradient more accurate for the true global optimum, which can be further verified by testbed measurements.

2) Testbed Verification: To empirically verify the impact of synchronization frequency on FL training performance in a microscopic manner, we experiment with both convex (SVM) and non-convex (LeNet-5 [25]) models. The SVM model is trained upon 50,000 randomly-generated points composed of 2 classes, with two clients each holding only 1 class; the LeNet-5 model is trained upon CIFAR-10 dataset [26], also with 2 clients each holding only 5 classes. The initial τ is 500, and at epoch 20 (100) for SVM (LeNet-5) we change it to 1. In Fig. 2, for each model we depict the instantaneous value of a randomly-selected parameter as well as the model accuracy. As suggested in the figure, with higher-quality gradients after increasing the synchronization frequency, the parameters can be refined out of suboptimal stagnation and reach a better model accuracy.

Although in the above example a higher frequency can bring remarkable accuracy performance benefit, we cannot directly set $\tau = 1$ for FL due to the prohibitively large communication cost. To make FedAvg accurate and also efficient, we need to strategically adjust the synchronization frequency during the training process. Yet, how do we know the time to adjust the synchronization frequency under the privacy constraint[3]? Based on our previous analysis, we find an intriguing gradient pattern being a great fit for that purpose.

B. Phenomenon of Gradient Bifurcation

During the model training process, while an individual gradient may exhibit strong randomness, there actually exists a clear statistical pattern for gradients across different clients. In practice we have observed an interesting phenomena called *gradient bifurcation*, meaning that gradients from different FL clients are consistent in the beginning but conflicting later. We elaborate on this phenomena with theoretical explanations as well as testbed measurements.

1) Theoretical Insight: From Lemma 1, we learn that the local gradient (u_τ^i) can be decoupled into a global component ($u_\tau^\star$) and a local-error one (e_τ^i): The former is identical for all the clients, while the latter is related to the heterogeneity of clients' local loss surfaces. At FL commencement, the model parameters are usually far away from the optimum, thus $u_\tau^\star$ dominates u_τ^i and this yields a strong gradient consistency. In contrast, when the model parameter ω moves close to the optimal region, $u_\tau^\star$ would shrink (due to convexity) and it is the error component e_τ^i that dominates u_τ^i. Consequently, gradients from different clients would gradually *bifurcate*.

We further illustrate the gradient bifurcation phenomena also with Fig. 1. When the initial parameter is far away from the optima region (-2 to 10), say -100, the gradients from both clients would be consistently positive. With a larger curvature, gradient of $L^1(\omega)$ would decay faster than that of $L^2(\omega)$: When the parameter moves across -2, the gradient of client-1 would become negative, conflicting with that of client-2. Finally the two gradients well counteract with each other, and the model parameter stagnates.

2) Testbed Verification: We further verify gradient bifurcation with testbed measurements. We train the LeNet-5 model following the setup in Fig. 2b (two clients with non-IID data), and measure the instantaneous gradient values of two randomly-selected parameters on both clients. As depicted in Fig. 3, the gradients for both parameters would bifurcate after around round-100, the extent of which exhibits a clear correlation with the model convergence status (measured by accuracy): In the two figures, the training process stagnates when the bifurcation level is approximately maximized.

To summarize, gradient bifurcation can effectively signal the FL training status without compromising FL privacy. Yet, it remains unclear how to quantify the gradient bifurcation level and how to leverage it for tuning synchronization frequency. We will answer this question in the next section.

IV. Gradient-Instructed Frequency Tuning

In this section, we propose Gradient-Instructed Frequency Tuning (GIFT), a frequency tuning algorithm for FL that is privacy-preserving, system-practical and also model-generic. We first propose a metric to quantify gradient bifurcation level, and then devise a feedback-driven heuristic to adjust synchronization frequency at runtime.

[3]Validating accuracy or loss on the FL server is also inappropriate to work as feedback signals. Because frequent validations would incur much overhead. Besides, with the continuous evolution of realistic data, it is hard to maintain an up-to-date validating dataset on the FL server.

A. Quantifying Gradient Bifurcation Level

To quantify the extent of gradient bifurcation, we first propose an intuitive metric: $C = \frac{\sum_i u^i_\tau}{\sum_i u^i_\tau}$. This metric depicts the effective portion of the aggregated gradient that does help the model to move towards the true optimum. Obviously, C is 1 when all the gradients are of the same direction, and is 0 if they well counteract with each other, i.e., when suboptimal stagnation occurs. Nonetheless, this metric is not practical for a real-world FL setup. To be clear, we summarize the practicality challenges of FL as follows:

1) *Stability Challenge*. Since samples processed in a SGD iteration are chosen *randomly*, local gradients fluctuate drastically, as can be seen in Fig. 3. This statistical instability may inundate the expected bifurcation pattern of gradient. Besides, in real-world FL setup the clients are also unstable: FL clients may join or leave randomly during training, and the FL server usually collects gradients from only a portion of the clients whoever reporting the earliest, so as to avoid waiting for stragglers [13], [19], [20]. A practical metric must be robust to such *statistical* and *systematical* instability.
2) *Scalability Challenge*. Meanwhile, in realistic FL applications like GBoard [20], there may be hundreds or thousands of clients participating simultaneously. Therefore, our metric should scale well in computing or storage overhead.

To tackle those challenges, we extend the previous metric definition with smoothing and pooling techniques.

1) Smoothing: To address statistical instability, we smooth the raw gradients with their historical values. To maintain low storage overhead, instead of using a window-based smoothing method, we calculate the gradients' *exponential moving average* (EMA). Let $\langle\cdot\rangle_\theta$ denote an exponential moving average with decay factor θ, and $u^i_{\tau,r}$ be the local gradient of client-i in r^{th} round, then we maintain $\tilde{u}^i_{\tau,r} = \langle u^i_{\tau,r}\rangle_\theta = \theta * \tilde{u}^i_{\tau,r-1} + (1-\theta) * u^i_{\tau,r}$.

2) Pooling: Note that maintaining $\tilde{u}^i_{\tau,r}$ for each client is memory-inefficient given the large client quantity, and is even infeasible due to participant instability. To tackle that problem, we further propose *bilateral gradient pooling*—the FL server only maintains two EMA gradients: one to collect the *positive* gradients from any client, and the other the *negative* ones. When the local gradients bifurcate, the two EMA gradients also bifurcate. This way, we can get a stable gradient statistics pattern despite the unstable client participation.

3) Gradient Consistency: Combining the above smoothing and pooling techniques, at round r, we define our customized metric, *gradient consistency*, as:

$$C = \frac{\tilde{P}_r + \tilde{N}_r}{\tilde{P}_r + \tilde{N}_r}, \quad \text{where} \begin{cases} \tilde{P}_r = \left\langle \sum_i \textbf{Relu}(u^i_{\tau,r}) \right\rangle_\theta \\ \tilde{N}_r = \left\langle \sum_i \textbf{-Relu}(\text{-}u^i_{\tau,r}) \right\rangle_\theta \end{cases}. \tag{1}$$

Here $\tilde{P}_r$ collects the EMA values of all the positive gradient components, and $\tilde{N}_r$ collects the EMA values of all the negative gradient components (we use the **Relu** operation for sign filtering). When the positive components and negative components well counteract with each other, C would be close to 0. Gradient consistency is thus a practical metric that can represent how fast (in terms of the useful share out of the aggregated gradient) the model is being refined to the optimum. While ideally it shall decrease to zero when model training stagnates, it is not so in practice because the impact of random mini-batches cannot be completely eliminated by smoothing; besides, for deep neural networks, there might be irregular landscapes like *flat minima* [27], where the gradient is not zero even at a local optimum. Therefore, we diagnose training stagnation not by absolute value of the gradient consistency but by its stabilizing behavior.

Algorithm 1 FL Workflow With GIFT

Require: τ, γ, δ, o ▷ τ: synchronization frequency; γ: τ scale down divisor; δ: τ scale up addend; o: observation window size to trigger τ scale up.

Client: $i = 1, 2, \ldots, N$**:**

Procedure ClientIterate(k)
  $k \leftarrow k + 1$ ▷ update iteration_id
  $\omega^i_k \leftarrow \omega^i_{k-1} + u^i_k$ ▷ u^i_k: local update in iteration k on client i
  if $k = k_s$ **then** ▷ k_s: next iteration_id for model synchronization
    $\omega^i_k, \tau \leftarrow$`FL_Server.aggregate`$(\omega^i_k)$
    ▷ synchronize global parameters and update the sync frequency
    $k_s \leftarrow k + \tau$

FL Server:

Procedure Aggregate($\omega^1_k, \omega^2_k, \ldots, \omega^N_k$)
  $r \leftarrow r + 1$ ▷ update round_id
  $\omega_r \leftarrow \frac{1}{n}\sum_{i=1}^n \omega^i_k$ ▷ update global model
  $u^i_r \leftarrow \omega^i_k - \omega_{r-1}, i = 1, 2, \ldots, N.$ ▷ get local accumulated update
  calculate C_r from $\{u^i_r\}$ based on Eq. 1
  if $C_r \geq C_{r-1}$ **then**
    $\tau_r \leftarrow \tau_{r-1}/\gamma$ ▷ multiplicatively increase sync frequency
  else if $C_{m+1} < C_m$ and $\tau_{m+1} = \tau_m \ \forall m \in \{r-1, \ldots, r-o\}$, **then**
    $\tau_r \leftarrow \tau_{r-1} + \delta$ ▷ additively decrease sync frequency
  Return ω_r, τ_r

B. Tuning Frequency in a Feedback-Driven Manner

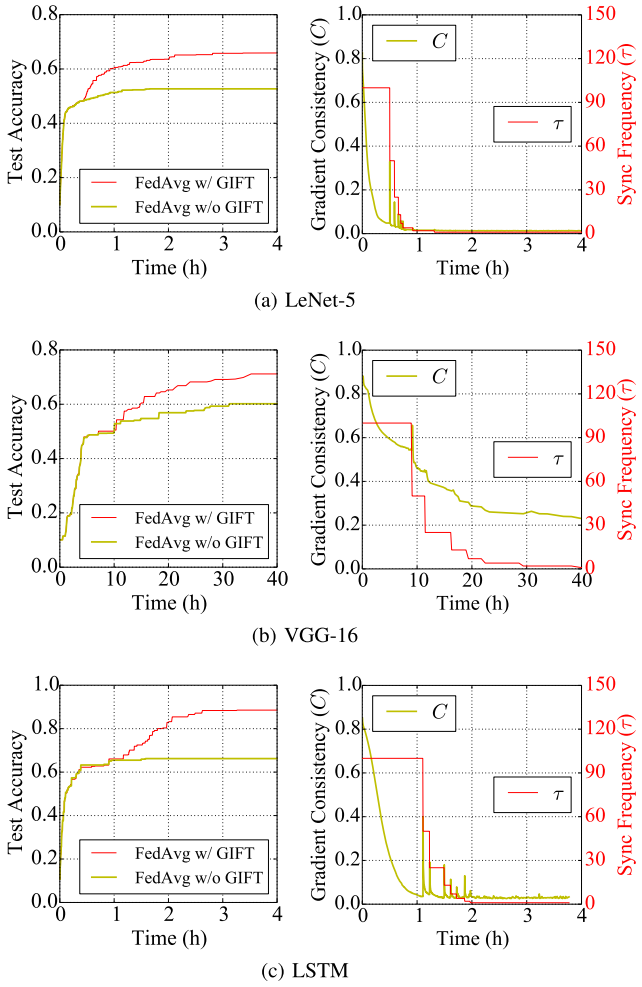
To be model-generic, instead of deriving a subtle formula representing the best synchronization frequency, we propose *Gradient-Instructed Frequency Tuning* (GIFT), a feedback-driven frequency tuning method based on the proposed gradient consistency metric. In GIFT, each time the gradient consistency stabilizes—a feedback signal suggesting that the model can no longer be effectively refined under the current frequency (τ)—we would scale down τ by a fixed divisor (e.g., 2). With such a higher frequency, it is expected that the

aggregated gradient in each round can be more accurate and the model can reach a higher accuracy.

Moreover, we also incorporate an extension that allows the synchronization frequency to be modestly decreased at the initial phase of the FL process. At FL commencement, the global component $u_\tau^\star$ dominates u_τ^i, rendering the gradient quite consistent (as shown in Fig. 3), and it is not necessary to conduct frequent synchronization. To exploit this optimization opportunity, we *tentatively* increase τ in a linear manner every a few rounds, until the gradient consistency metric signals that the FL process stagnates. Yet, inflating τ is essentially trading computation for communication efficiency, and to avoid the risk of over-compromised computation efficiency, frequency relaxation is not enabled by default.

Implementation: We have implemented GIFT with PyTorch, and the detailed workflow is shown in Alg. 1. In Procedure `ClientIterate`, lines 2-3 describe the regular local updating, and lines 4-6 mean that the client shall communicate with the FL Server every τ iterations to get the latest model ω as well as the updated synchronization frequency. In Procedure `Aggregate`, the FL Server first updates the global model with standard FedAvg (lines 8-9), and then calculates gradient consistency metric (lines 10-11). In particular, if gradient consistency no longer decreases, the synchronization frequency represented by τ is divided by γ (lines 12-13); if otherwise the gradient consistency metric keeps decreasing, τ is increased with the addend δ (lines 14-15).

Regarding the algorithm complexity, with the pooling technique we need to maintain the EMA values of $\tilde{P}_r$ and $\tilde{N}_r$, each has the same size as the model (denoted by M). Therefore, the space complexity is $\mathcal{O}(M)$ (with a very small coefficient). Similarly, since GIFT enforces identical operations for each gradient coordinate, the computation complexity to calculate gradient consistency C is also $\mathcal{O}(M)$.



(a) LeNet-5

(b) VGG-16

(c) LSTM

Fig. 4. FL performance with and without GIFT (accompanied by the frequency τ and gradient consistency C under GIFT).

V. Evaluation

In this section, we evaluate GIFT performance with testbed experiments. We first visually and quantitatively verify the effectiveness of GIFT in a 100-node cluster emulating realistic FL scenarios, and then justify the superiority of GIFT over existing practices. We also examined the behavior of GIFT under different levels of non-IID data, as well as the effectiveness of frequency relaxation. Finally, we conduct sensitivity analysis on the hyper-parameters involved.

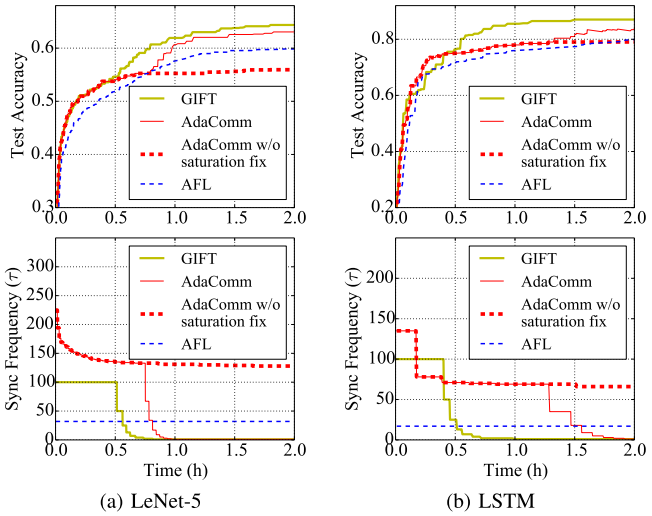
TABLE II
Time and Accuracy After 1000 Rounds

Model	Scheme	Time (h)	Accuracy (%)
LeNet-5	FedAvg	1.56	52.1
	GIFT	0.64	56.8
VGG-16	FedAvg	81.9	60.8
	GIFT	34.3	67.3
LSTM	FedAvg	3.14	66.2
	GIFT	1.62	74.8

A. Experimental Setup

1) Hardware Platform: We emulate real-world FL scenarios with 100 `m5.large` instances on Amazon EC2, each with 2 vCPU cores and 8GB RAM (similar with that of a smart phone). The client bandwidth is configured to be 5Mbps with the wondershaper [28] tool. The FL server is a `c5.9xlarge` instance with 10Gbps bandwidth.

2) Training Setup: Models trained in our evaluation are LeNet-5 [25], VGG-16 [29] and a LSTM network (containing 2 recurrent layers with a hidden size of 64). LeNet-5 and VGG-16 are trained on the CIFAR-10 dataset [26] and the LSTM network is trained on the KeyWord Spotting (KWS) dataset—a subset of the Speech Commands dataset [30] including 10 key words. To be realistic, instead of partitioning the initial dataset after label sorting, we let the samples on each client independently follow a Dirichlet distribution [3], [5], which controls label class composition via a concentration parameter α. In our experiments we set α to 1, emulating a modest non-IID level. The learning rates are set to 0.01 (LeNet-5), 0.1 (VGG-16) and 0.05 (KWS), with weight decay of 0.01, 0.0005 and 0.01. The initial τ is set to 100. In GIFT, the EMA smoothing factor θ is 0.9, and τ is divided by 2 once gradient consistency no longer decreases.



(a) LeNet-5 (b) LSTM

Fig. 5. Comparison between GIFT and existing methods.

B. GIFT Evaluation Under a Realistic FL Setup

We first evaluate the overall performance improvement of GIFT over FedAvg in a realistic FL setup with dynamic client participation at a relatively large scale. Our cluster contains 100 clients: We let each client delay for a random period before reporting its gradient, and in each round the FL server only collects 40% gradients reported the earliest.

In Fig. 4, we show the overall performance of GIFT when training the three models, including the instantaneous gradient consistency (C) and synchronization frequency (τ). As analyzed in §III-B, C does follow a decreasing pattern.[4] Moreover, each time τ is halved under GIFT, in the next moment there is a salient burst of C, indicating that a higher frequency can reduce gradient error and make the aggregated model closer to the true optimum.

We further make quantitative performance comparison between GIFT and standard FedAvg in Table II, which lists the time cost and accuracy attained when training each model for 1000 rounds. By adaptively tuning the synchronization frequency to ensure continuous model improvement, GIFT enables each model to achieve a higher accuracy[5] with less time consumption. For example, GIFT can reduce LeNet-5 training time by 58.9% while improving the accuracy by 9%. Additionally, thanks to our pooling technique in §IV, on the FL server we did not observe noticeable CPU and memory overhead after enabling GIFT.

C. Comparison With Existing Methods

We implement AFL [17] and AdaComm [7], two typical methods discussed in §II-B, also in PyTorch. In our evaluation, we obtain the ground-truth knowledge (e.g., Lipschitz parameter, smoothness parameter, gradient divergence bound) required by AFL via a trial run, and in AdaComm the initial frequency is selected via grid search. In particular, to address loss saturation due to plateaus or noises, AdaComm adopts a saturation fix that halves τ once the loss value no longer decreases; to evaluate the true effectiveness of the formula-based solution, we incorporate a pruned version of AdaComm without that saturation fix. Additionally, since those methods do not consider system challenges of FL, for fair comparison we switch to a 20-node cluster with full client participation.

Fig. 5 shows the instantaneous accuracy and frequency when training LeNet-5 and LSTM under different schemes. It shows that, apart from the advantages of privacy preserving and practicality, GIFT can also attain a better accuracy performance over existing methods. Given a LeNet-5 accuracy target of 0.6, time consumption under GIFT is 28.9% less than AdaComm, and 61.2% less than AFL. We also note that AdaComm without saturation fix fails to behave well for both models, confirming the weakness of existing methods as discussed in §II-B.

Regarding the reasons behind, for AFL, the fixed frequency calculated is too small to be communication-efficient in the beginning, and is on the other hand too large to attain high accuracy in the end. For AdaComm, since its formulation is based on IID label distribution, the formula on τ is actually inaccurate for realistic FL setup. Moreover, due to loss plateaus problem (i.e., accuracy improved but the loss value not so), the training loss is sometimes not an appropriate indicator of the training status.[6]

D. Effect of Frequency Relaxation

We further evaluate the effectiveness of frequency relaxation extension. To be specific, we additively increase τ by 5 once C keeps decreasing for 10 consecutive rounds, and Fig. 7 depicts the testing accuracy (against communication rounds) when training LeNet-5 and LSTM with the 20-node cluster. For both models, frequency relaxation can yield a prompter accuracy improvement especially in the early stage. For example, after training LeNet-5 for 500 rounds, it can achieve a test accuracy of 0.55, 5.1% better than that without frequency relaxation. Note that such benefit would be larger for cases with a smaller τ initialization, as verified by the LSTM training results in Fig. 7b where τ_0 is set to 10.

E. GIFT Behavior Under Different Levels of Non-IID Data

So far, we are operating with a modest level of non-IID data controlled by the hyper-parameter $\alpha = 1$ (§V-A); yet, how would GIFT behave if the data distribution is otherwise IID or extremely non-IID? To further reveal that, we resort to micro-benchmark experiments with different levels of non-IID data. In the IID case, each worker has a full copy the entire dataset; in the non-IID case, we separate the whole dataset across 5 clients—each hosting 2 label classes with no

[4]For VGG-16 in Fig. 4b, C stagnates at a large value, because VGG-16 is a large, over-parameterized model [31] with salient flat minima behavior [27].

[5]We accept near-optimal accuracy instead of pursuing for the optimal accuracy because the latter requires training with $\tau = 1$ for many ($\sim 10^5$) rounds, which is cost-prohibitive even with GIFT.

[6]It is well-known that the training loss in SGD may get stuck on plateaus landscape or fluctuate due to mini-batch randomness [7]. Once that occurs (i.e., the loss no longer decreases), AdaComm would degrade to naive frequency scaling (i.e., having τ halved), which works as a boundary-case remediation. Nonetheless, from Fig. 5 we find that the accuracy improvement of AdaComm largely comes from such a boundary-case remediation.

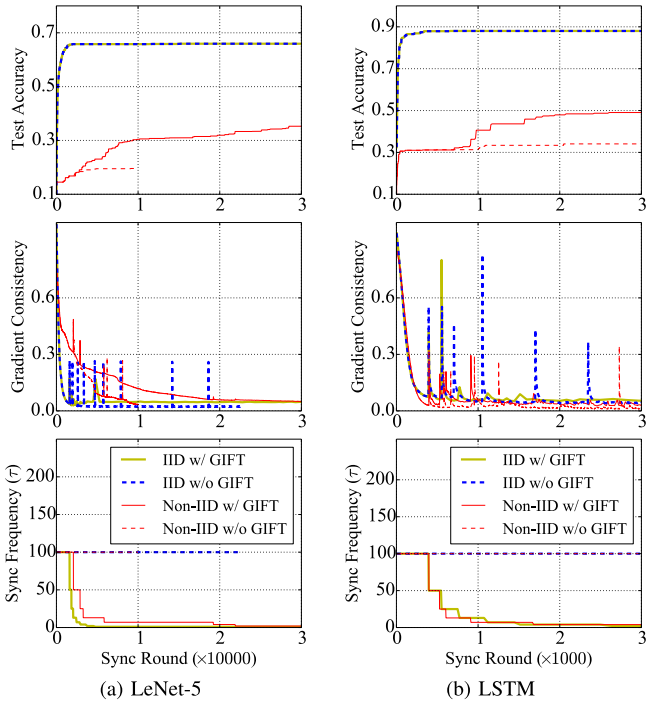


Fig. 6. GIFT Behavior under different levels of non-IID data.

overlap. In each case we record the instantaneous test accuracy, gradient consistency and sync-frequency under GIFT as well as the test accuracy under vanilla FedAvg.

From Fig. 6, we first learn that data distribution remarkably affects the model training performance. When training LeNet and LSTM with the extremely non-IID setup, there is an accuracy loss of nearly 50%, and the training time required towards model convergence is also hugely inflated—consistent with previous measurements in [2], [3], [4], [5]. Regarding the benefit of GIFT, for the IID case the GIFT performance is similar with that under vanilla FedAvg (as implied by 2), and for the non-IID case GIFT can achieve salient accuracy improvement. Regarding the *gradient consistency* metric, we note that for the IID case there is also a decaying trend. This is because in SGD the randomness incurred by mini-batch sampling gradually dominates the local gradients; with the pooling technique (§IV-A), the calculated gradient consistency would also keep decreasing. Regarding the frequency-tuning actions, we find that the frequency decaying paces in both IID and non-IID cases are mostly similar. In general, GIFT can work smoothly without particular requirements on the data IID level.

F. Sensitivity Analysis

In Fig. 8 we further evaluate the impact of GIFT sync-interval divisor, i.e., γ in Alg. 1. We change it—from the default value 2—respectively to 3 and 10, and evaluate the GIFT performance in the 20-node cluster. As shown in Fig. 8, GIFT behavior is generally stable with different γ: The synchronization interval τ is decreased to 1 at a similar time, and the accuracy performance is also similar. This suggests that our GIFT algorithm is robust to the γ hyper-parameter.

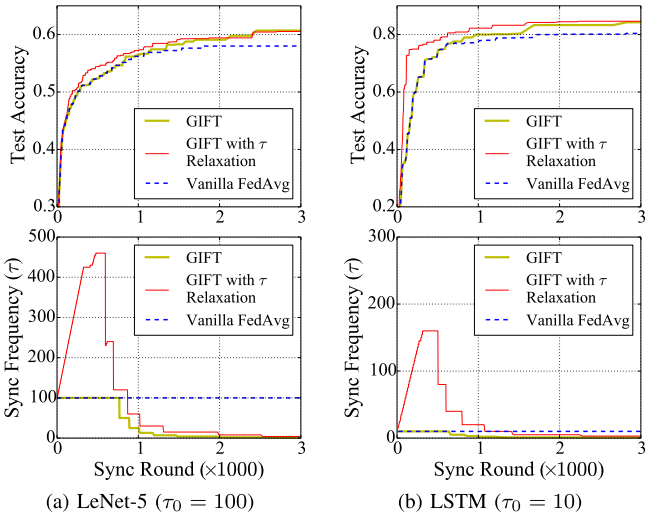


Fig. 7. Frequency relaxation can yield faster accuracy increase especially in the earlier stage.

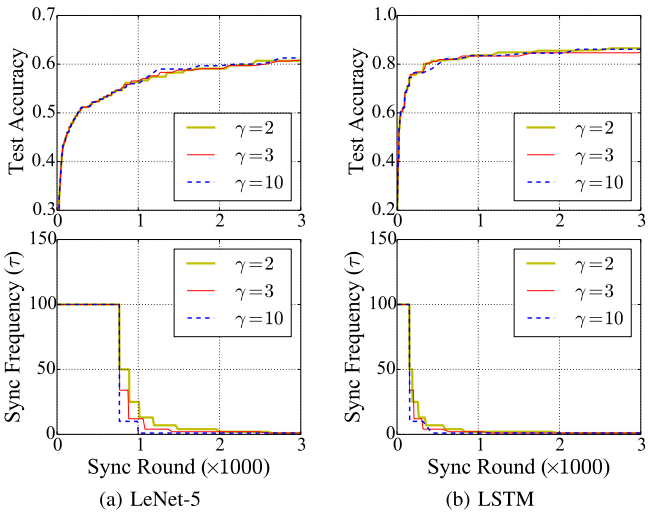
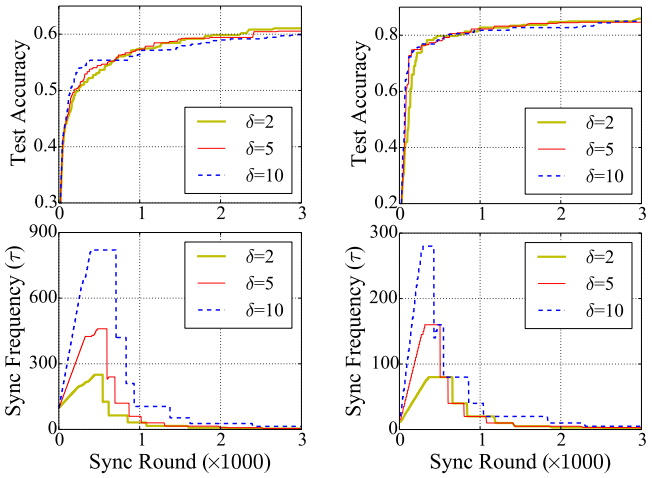


Fig. 8. GIFT Performance with different γ values (γ is used to divide τ when increasing synchronization frequency).

Meanwhile, recall that when enabling frequency relaxation (as in §V-D), the default addend δ to increase τ is set to 5; here we also evaluate its sensitivity with Fig. 9. For each of the experiments in Fig. 7, we change δ respectively to 2 and 10 and repeat the same training process. As shown in Fig. 9, a larger δ can yield a faster accuracy improvement in the early stage, although that speedup would be caught up by others in the later stage when a smaller τ becomes more beneficial. In general, the performance of GIFT is also stable with respect to different δ values.

VI. Additional Related Work and Discussion

Recall that in §II-B we have introduced a series of frequency-setup methods for FL; to improve FL accuracy or efficiency, in the literature there have been some other interesting directions not involving frequency-tuning. In this section, we will briefly review those parallel directions and further discuss the privacy-preserving property of GIFT.



(a) LeNet-5 (b) LSTM

Fig. 9. GIFT Performance with different δ values (δ is used to increase τ when relaxing synchronization frequency).

A. Improving FL Accuracy

To mitigate accuracy loss due to non-IID data in FL, two typical methodologies other than frequency tuning are *data complementing* and *optimization rectifying*. Data complementing means to reduce data non-IID level by copying a few common samples to each client [4], or by augmenting clients' local datasets with auxiliary samples generated from GAN (Generative Adversarial Network) models [32]. Yet these methods may incur large computation and communication overheads. Optimization rectifying means to add an extra regularizing term to the loss function [32], [33], or to add a specific momentum to the optimizer [34]. These methods are not transparent to models and require expert knowledge to set up. Overall speaking, our GIFT method outperforms those methods by being light-weight and also transparent to end users, not requiring any modifications to the user-specified loss function or optimizers.

B. Improving Communication Efficiency

To reduce communication cost of distributed training, *quantization* and *sparsification* are two well-known methodologies in addition to frequency tuning. Quantization reduces bandwidth consumption by transmitting low-precision updates, balancing the trade-off between accuracy and gradient compressing level [1], [35], [36]. Sparsification aims to synchronizing only the important gradients under a given criterion like normalized magnitude [37], [38], [39], [40]. These methods have proven effective in reducing the communication amount, yet they ignored the non-IID problem in FL, and may thus suffer accuracy loss when applied in realistic FL scenarios.

C. Privacy-Preserving Characteristics of GIFT

Privacy is not an absolutely-defined conception. By privacy-preserving, in this work we mean that GIFT does not compromise the accuracy level of vanilla FL, better than existing practices based on loss values. Nonetheless, it has been shown that certain user information can still be recovered from gradients [41]. To solve this problem, differential privacy (DP) [42], [43] is often adopted in FL to protect raw gradients by injecting Gaussian noises to them. We note that applying DP does not compromise the effectiveness of GIFT, because the noises added by DP exhibit a similar impact with that of mini-batch sample randomness, which is already solved with the smoothing and pooling techniques (§IV).

VII. Conclusion

In this work, to attain better model accuracy and resource efficiency in FL, we have proposed GIFT, a privacy-preserving, system-practical and model-generic scheme to adaptively tune the synchronization frequency at runtime. GIFT gauges the model training status with a novel metric called gradient consistency, based on which it then multiplicatively increases or linearly decreases the synchronization frequency. Prototype evaluations in realistic FL setup have demonstrated the superiority of GIFT in both accuracy and efficiency over existing methods.

Appendix

Lemma 2: Let ω_k^i be the local parameter on client-i ($i = 1, 2, \ldots, N$) after refined for k iterations from ω_0, and $\omega_k^\star$ be the ideal parameter when refined with IID dataset also for k iterations from ω_0. Under the Assumptions 1 and 2, let $u_\tau^i = \omega_\tau^i - \omega_0$ be the accumulated gradients on client-i after τ iterations, then $u_\tau^i = u_\tau^\star + e_\tau^i$, where $u_\tau^\star$ is the ideal gradient with IID data and e_τ^i is the local error component:

$$e_\tau^i = -\eta \sum_{k=0}^{\tau-1} [\nabla L^i(\omega_k^\star) - \nabla L^\star(\omega_k^\star) + \langle \nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star \rangle].$$

Proof: Since $u_\tau^\star$ is the ideal gradient with IID data, i.e., $u_\tau^\star = -\eta \sum_{k=0}^{\tau-1} \nabla L^\star(\omega_k^\star)$, and let $\Delta_k^i = \nabla L^i(\omega_k^i) - \nabla L^\star(\omega_k^\star)$ be the gradient error in the k^{th} iteration, we have

$$\begin{aligned} e_\tau^i &= u_\tau^i - u_\tau^\star = -\eta \sum_{k=0}^{\tau-1} [\nabla L^i(\omega_k^i) - \nabla L^\star(\omega_k^\star)] \\ &= -\eta \sum_{k=0}^{\tau-1} \Delta_k^i. \end{aligned} \tag{2}$$

Let $v^i(\omega) = \nabla L^i(\omega) - \nabla L^\star(\omega)$, we have:

$$\Delta_k^i = v^i(\omega_k^i) + [\nabla L^\star(\omega_k^i) - \nabla L^\star(\omega_k^\star)]. \tag{3}$$

Regarding the first term $v^i(\omega_k^i)$, because Assumption 2 indicates that $L^\star(\omega)$ is upper-bounded by a quadratic function, and $\omega_k^i - \omega_k^\star$ scaled by η can be arbitrarily small, we can skip higher-order items and get:

$$v^i(\omega_k^i) = v^i(\omega_k^\star) + \langle \nabla v^i(\omega_k^\star), \omega_k^i - \omega_k^\star \rangle. \tag{4}$$

Similarly, regarding the second term in Eq. (3) we can get

$$\begin{aligned} \nabla L^\star(\omega_k^i) - \nabla L^\star(\omega_k^\star) &= \nabla L^\star(\omega_k^\star + \omega_k^i - \omega_k^\star) - \nabla L^\star(\omega_k^\star) \\ &= \langle \nabla^2 L^\star(\omega_k^\star), \omega_k^i - \omega_k^\star \rangle. \end{aligned} \tag{5}$$

With Eq. (4) and Eq. (5), we can transfer Eq. (3) to

$$\begin{aligned}\Delta_k^i &= v^i(\omega_k^\star) + \langle \nabla v^i(\omega_k^\star) + \nabla^2 L^\star(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle \\ &= v^i(\omega_k^\star) + \langle \nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle. \end{aligned} \tag{6}$$

Therefore, we can get

$$\begin{aligned} e_\tau^i &= -\eta \sum_{k=0}^{\tau-1} \Delta_k^i = -\eta \sum_{k=0}^{\tau-1} [v^i(\omega_k^\star) + \langle \nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle] \\ &= -\eta \sum_{k=0}^{\tau-1} [\nabla L^i(\omega_k^\star) - \nabla L^\star(\omega_k^\star) + \langle \nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle]. \end{aligned}$$

This completes our proof. □

Theorem 3: Under the Assumptions 1 and 2, let $\bar{u}_\tau = \frac{1}{N}\sum_{i=1}^N u_\tau^i$ *and let* $d(\tau) = \bar{u}_\tau - u_\tau^\star$ *be aggregated gradient error after each client locally refines the parameter for* τ *iterations from* ω_0*, then*

$$d(\tau) \geq (\tau - 1)\frac{\eta^2}{N} \sum_{i=1}^N \langle \nabla^2 L^i(\omega_0), \nabla L^i(\omega_0) - \nabla L^\star(\omega_0)\rangle.$$

Proof: Since $\frac{1}{N}\sum_{i=1}^N \nabla L^i(\omega_k^\star) = \nabla L^\star(\omega_k^\star)$, with Lemma 2 we can get:

$$d(\tau) = \frac{1}{N}\sum_{i=1}^N e_\tau^i = \frac{-\eta}{N} \sum_{k=0}^{\tau-1}\sum_{i=1}^N \langle \nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle. \tag{7}$$

Let Δ_k be the aggregated error in k^{th} ($k = 0, \ldots, \tau - 1$) iteration:

$$\Delta_k = \frac{-\eta}{N} \sum_{i=1}^N \langle \nabla^2 L^i(\omega_k^\star), \omega_k^i - \omega_k^\star\rangle \tag{8}$$

Next, we seek to prove $\Delta_{k+1} \geq \Delta_k$ with mathematical induction.

When $k = 0$, since $\omega_0^\star = \omega_0^i = \omega_0$, we have $\Delta_0 = 0$.

When $k = 1$, since $\omega_1^i = \omega_0 - \eta \nabla L^i(\omega_0)$ and $\omega_1^\star = \omega_0 - \eta \nabla L^\star(\omega_0)$, and also let $v^i(\omega) = \nabla L^i(\omega) - \nabla L^\star(\omega)$, we have

$$\begin{aligned}\Delta_1 &= \frac{-\eta}{N} \sum_{i=1}^N \langle \nabla^2 L^i(\omega_1^\star), \omega_1^i - \omega_1^\star\rangle \\ &= \frac{\eta^2}{N} \sum_{i=1}^N \langle \nabla^2 L^i(\omega_1^\star), v^i(\omega_0)\rangle. \end{aligned} \tag{9}$$

Since $\sum_i v^i(\omega_0) = 0$, Δ_1 can be viewed as the summation of conflicting values $\{v^i(\omega_0)\}$ assigned with respective weights $\{\nabla^2 L^i(\omega_1^\star)\}$. If with IID data such that $\forall i, \nabla^2 L^i(\omega) \equiv \nabla^2 L^\star(\omega)$, we then have $\Delta_1 = 0$; otherwise, if for non-IID data that usually yields heterogeneous $\{\nabla^2 L^i(\omega)\}$, we have $\Delta_1 \neq 0$, i.e., $\Delta_1 > \Delta_0 = 0$.

For clarity we focus on the case where there is only one dimension in ω. Given that $\{L_i(\omega)\}$ are convex and smooth (the global optimum is optimal in each dimension), the conclusion of single-dimension case can be extended to multi-dimensional cases.

Given that $\Delta_1 > 0$, without loss of generality we can assume that $\Delta_1 > 0$. With Eq. (8) the set of $\{i\}$ can be divided into two groups: $\{i^+\}$—those such that $\omega_1^i > \omega_1^\star$; and $\{i^-\}$—those such that $\omega_1^i < \omega_1^\star$. For simplicity, we assume there are only two clients: i^+ and i^-. Given $\Delta_1 > 0$, under Eq. (9) and with convexity we have:

$$\nabla^2 L^{i^-}(\omega_1^\star) > \nabla^2 L^{i^+}(\omega_1^\star) > 0. \tag{10}$$

Under mathematical induction, we assume that $\forall k \in \{0, 1, 2, \ldots, K-1\}$, $\Delta_{k+1} > \Delta_k \geq 0$, and seek to prove $\Delta_{K+1} > \Delta_K$. Based on the assumptions, we first have

$$\bar{\omega}_{K+1} - \omega_{K+1}^\star = \sum_{k=0}^K \Delta_k > \sum_{k=0}^{K-1} \Delta_k = \bar{\omega}_K - \omega_K^\star > 0 \tag{11}$$

Given that $\bar{\omega}_k = \frac{\omega_k^{i^+} + \omega_k^{i^-}}{2}$, Eq. (11) translates to

$$\begin{aligned}&(\omega_{K+1}^{i^+} - \omega_{K+1}^\star) - (\omega_{K+1}^\star - \omega_{K+1}^{i^-}) \\ &\qquad > (\omega_K^{i^+} - \omega_K^\star) - (\omega_K^\star - \omega_K^{i^-}) > 0. \end{aligned} \tag{12}$$

We next seek to prove $\Delta_{K+1} > \Delta_K$. Comparing Δ_{K+1} and Δ_K, we have

$$\begin{aligned}\Delta_{K+1} - \Delta_K = &\sum_{i\in\{i^+,i^-\}} -\frac{\eta}{N}\nabla^2 L^i(\omega_{K+1}^\star)(\omega_{K+1}^i - \omega_{K+1}^\star) \\ &- \sum_{i\in\{i^+,i^-\}} -\frac{\eta}{N}\nabla^2 L^i(\omega_K^\star)(\omega_K^i - \omega_K^\star) \end{aligned} \tag{13}$$

Under Assumption 2 (i.e., $\nabla^2 L^i(\omega_{K+1}^\star)$ is bounded by a constant) and with a small step size, we approximately have $\nabla^2 L^i(\omega_{K+1}^\star) = \nabla^2 L^i(\omega_K^\star) = \nabla^2 L^i(\omega_0^\star)$ (in fact $\nabla^2 L^i(\omega)$ is a constant if $L_i(\omega)$ are quadratic functions). We further have

$$\begin{aligned}&\Delta_{K+1} - \Delta_K \\ &= -\frac{\eta}{N} \sum_{i\in\{i^+,i^-\}} \nabla^2 L^i(\omega_0^\star)[(\omega_{K+1}^i - \omega_{K+1}^\star) - (\omega_K^i - \omega_k^\star)] \\ &= -\frac{\eta}{N}\nabla^2 L^{i^+}(\omega_0^\star)[(\omega_{K+1}^{i^+} - \omega_{K+1}^\star) - (\omega_K^{i^+} - \omega_K^\star)] \\ &\quad + \frac{\eta}{N}\nabla^2 L^{i^-}(\omega_0^\star)[(\omega_{K+1}^\star - \omega_{K+1}^{i^-}) - (\omega_K^\star - \omega_K^{i^-})]. \end{aligned} \tag{14}$$

Combining Eq. (10), Eq. (12) with Eq. (14), we can obtain $\Delta_{K+1} - \Delta_K > 0$, i.e., $\Delta_{K+1} > \Delta_K > 0$.

Therefore for general case with $\tau > 1$ we have,

$$\begin{aligned} d(\tau) &= \sum_{k=0}^{\tau-1} \Delta_k > (\tau - 1)\Delta_1 \\ &= (\tau - 1)\frac{\eta^2}{N} \sum_{i=1}^N \langle \nabla^2 L^i(\omega_0), v^i(\omega_0)\rangle. \end{aligned}$$

Integrating the case of $\tau = 1$ we complete the proof. □

Theorem 4 (Suboptimal Convergence): Suppose FL process converges at a point $\bar{\omega}^\star$*, and* $\omega^\star$ *is the ideal parameter under IID dataset. Then*

$$\bar{\omega}^\star - \omega^\star \geq \frac{(\tau - 1)\eta}{\beta N} \sum_{i=1}^N \langle \nabla^2 L^i(\bar{\omega}^\star), \nabla L^i(\bar{\omega}^\star) - \nabla L^\star(\bar{\omega}^\star)\rangle.$$

Proof: Under Theorem 3, we treat $\bar{u}_\tau$, $u_\tau^\star$ and d as a function of ω_0, then we have $d_\tau(\omega) = \bar{u}_\tau(\omega) - u_\tau^\star(\omega)$.

When FL process stagnates at $\bar{\omega}^\star$, i.e., $\bar{u}_\tau(\bar{\omega}^\star) = 0$, we have

$$u_\tau^\star(\bar{\omega}^\star) \geq \frac{(\tau-1)\eta^2}{N}\sum_{i=1}^{N}\langle \nabla^2 L^i(\bar{\omega}^\star), \nabla L^i(\bar{\omega}^\star) - \nabla L^\star(\bar{\omega}^\star)\rangle. \tag{15}$$

Meanwhile, since $u_\tau^\star(\omega^\star) = 0$, with Assumption 2 we have:

$$\begin{aligned} u_\tau^\star(\bar{\omega}^\star) &= u_\tau^\star(\omega^\star + \bar{\omega}^\star - \omega^\star) \\ &= u_\tau^\star(\omega^\star) + \langle \nabla u_\tau^\star(\omega^\star), \bar{\omega}^\star - \omega^\star\rangle \\ &= \langle \nabla u_\tau^\star(\omega^\star), \bar{\omega}^\star - \omega^\star\rangle \leq \eta\beta\|\bar{\omega}^\star - \omega^\star\|. \end{aligned} \tag{16}$$

Combining Eq. (15) and Eq. (16) our proof completes. □

REFERENCES

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

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," 2016, *arXiv:1602.05629*.

[3] T.-M. H. Hsu, H. Qi, and M. Brown, "Measuring the effects of non-identical data distribution for federated visual classification," 2019, *arXiv:1909.06335*.

[4] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[5] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, T. N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *Proc. ICML*, 2019.

[6] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. R. Cadambe, "Local SGD with periodic averaging: Tighter analysis and adaptive synchronization," in *Proc. NIPS*, 2019.

[7] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proc. SysML*, 2019.

[8] P. Jiang and G. Agrawal, "Adaptive periodic averaging: A practical approach to reducing communication in distributed learning," 2020, *arXiv:2007.06134*.

[9] A. Khaled, K. Mishchenko, and P. Richtárik, "Tighter theory for local SGD on identical and heterogeneous data," in *Proc. AISTATS*, 2020.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[12] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014, *arXiv:1409.3215*.

[13] K. Bonawitz et al., "Towards federated learning at scale: System design," in *Proc. SysML*, 2019.

[14] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[15] C. Zhang, S. Li, J. Xia, W. Wang, F. Yan, and Y. Liu, "Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning," in *Proc. USENIX ATC*, 2020.

[16] S. Wang et al., "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE INFOCOM*, Apr. 2018, pp. 63–71.

[17] S. Wang et al., "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Mar. 2019.

[18] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *Proc. IEEE INFOCOM*, May 2021, pp. 1–10.

[19] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *Proc. USENIX OSDI*, 2021, pp. 1–18.

[20] T. Yang et al., "Applied federated learning: Improving Google keyboard query suggestions," 2018, *arXiv:1812.02903*.

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

[22] 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.*, vol. 33, 2020, pp. 7611–7623.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[24] 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. 1–39, 2011.

[25] 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.

[26] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[27] S. Hochreiter and J. Schmidhuber, "Flat minima," *Neural Comput.*, vol. 9, no. 1, pp. 1–42, 1997.

[28] (2020). *Wonder Shaper*. [Online]. Available: https://github.com/magnific0/wondershaper

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

[30] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018, *arXiv:1804.03209*.

[31] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "The role of over-parametrization in generalization of neural networks," in *Proc. ICLR*, 2018.

[32] E. Jeong, S. Oh, H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Communication-efficient on-device machine learning: Federated distillation and augmentation under non-IID private data," 2018, *arXiv:1811.11479*.

[33] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2018, *arXiv:1812.06127*.

[34] C. Li et al., "Gradient scheduling with global momentum for non-iid data distributed asynchronous training," 2019, *arXiv:1902.07848*.

[35] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs," in *Proc. INTERSPEECH*, 2014.

[36] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Proc. NIPS*, 2017.

[37] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in *Proc. Interspeech*, Sep. 2015.

[38] K. Hsieh et al., "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. USENIX NSDI*, 2017.

[39] N. Dryden, T. Moon, S. A. Jacobs, and B. Van Essen, "Communication quantization for data-parallel training of deep neural networks," in *Proc. 2nd Workshop Mach. Learn. HPC Environments (MLHPC)*, Nov. 2016, pp.'1–8.

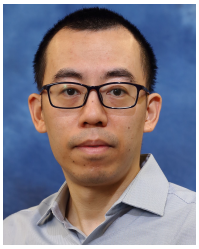
[40] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 954–964.

[41] Y. Aono et al., "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

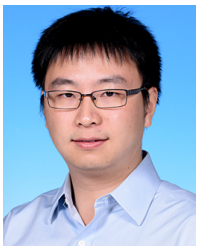
[42] M. A. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Proc. NIPS*, 2010.

[43] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015.

Chen Chen (Member, IEEE) received the B.Eng. degree from Tsinghua University, Beijing, China, in 2014, and the Ph.D. degree from the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, in 2018. He was a Researcher at the Theory Laboratory, Huawei Hong Kong Research Center. He is currently an Associate Professor with the John Hopcroft Center for Computer Science, Shanghai Jiao Tong University. His recent research interests include distributed deep learning, big data systems, and networking.



Hong Xu (Senior Member, IEEE) received the B.Eng. degree from The Chinese University of Hong Kong (CUHK), Hong Kong, in 2007, and the M.A.Sc. and Ph.D. degrees from the University of Toronto in 2009 and 2013, respectively. From 2013 to 2020, he was with the City University of Hong Kong. He is currently an Associate Professor with the Department of Computer Science and Engineering, CUHK. His research interests include computer networking and systems, particularly big data systems and data center networks. He is a Senior Member of ACM. He was a recipient of an Early Career Scheme Grant from the Hong Kong Research Grants Council in 2014. He received three best paper awards, including the IEEE ICNP 2015 Best Paper Award.



Wei Wang (Member, IEEE) received his B.Eng. and M.Eng. degrees from the Department of Electrical Engineering, Shanghai Jiao Tong University, China, in 2007 and 2010, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Toronto, Canada, in 2015. Since 2015, he has been with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology (HKUST), where he is currently an Associate Professor. He is also affiliated with the Big Data Institute, HKUST. His research interests include distributed systems, with focus on serverless computing, machine learning systems, and cloud resource management. He has published extensively in the premier conferences and journals of his fields. His research has won the Best Paper Runner Up Awards of IEEE ICDCS 2021 and USENIX ICAC 2013.

Baochun Li (Fellow, IEEE) received the Ph.D. degree from the Department of Computer Science, University of Illinois at Urbana–Champaign, Urbana, IL, USA, in 2000. Since then, he has been with the Department of Electrical and Computer Engineering, University of Toronto, where he is currently a Professor. He has been the Bell Canada Endowed Chair in computer engineering since August 2005. His research interests include large-scale distributed systems, machine learning, security, cloud computing, and wireless networks. He is a member of ACM.

Bo Li (Fellow, IEEE) received the B.Eng. degree (summa cum laude) in computer science from Tsinghua University, Beijing, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts at Amherst, Amherst, MA, USA.

He was the Cheung Kong Visiting Chair Professor at Shanghai Jiao Tong University from 2010 to 2016, the Chief Technical Advisor at ChinaCache Corporation (NASDAQ:CCIH), a Leading CDN Provider, and an Adjunct Researcher at Microsoft Research Asia (MSRA) from 1999 to 2006 and Microsoft Advanced Technology Center from 2007 to 2008. He is currently the Chair Professor with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology. He made pioneering contributions in multimedia communications and the internet video broadcast, in particular Coolstreaming system, which was credited as first large-scale peer-to-peer live video streaming system in the world. It attracted significant attention from both industry and academia and received the Test-of-Time Best Paper Award from IEEE INFOCOM in 2015. He was the Co-TPC Chair of IEEE INFOCOM in 2004. He is an editor or a guest editor of more than two dozen of IEEE and ACM journals and magazines.

Li Chen (Member, IEEE) received the B.E., M.Phil., and Ph.D. degrees from The Hong Kong University of Science and Technology (HKUST), Hong Kong, in 2011, 2013, and 2018, respectively. He was a Systems Researcher at the Huawei Theory Laboratory. He is currently working with the Zhongguancun Laboratory. His research interests include data center networks, distributed systems, vert computing, machine learning, and OAM.

Gong Zhang (Member, IEEE) is currently a Principal Researcher with Huawei 2012 Laboratories. He has more than 18 years of research experience in network communication and distributed systems, and has contributed more than 90 patents globally. He was in charge of Advance Network Technology Research Department in 2009. He has been a Team Leader for future internet and cooperative communication research since 2005 and led the System Group in data mining and machine learning since 2012. His recent research interests include network architecture and large-scale distributed systems.