GIFT: Towards Accurate and Efficient Federated Learning with Gradient-Instructed Frequency Tuning

Anonymous Authors[1]

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 efficient and also accurate. Specifically, we show that only the gradients can be utilized in frequency tuning decisions under the FL privacy requirement, and introduce a novel metric called *gradient consistency*, which can effectively reflect the training status despite the instability of realistic FL scenarios. We further propose a heuristic algorithm, Gradient-Instructed Frequency Tuning (GIFT), that multiplicatively increases the synchronization frequency once a FL process is diagnosed to stagnate. We have implemented GIFT in PyTorch, and large-scale evaluations show that it can improve FL accuracy by up to 10.7% with the same communication cost.

1. Introduction

Federated learning (FL) (Konečnỳ et al., 2016; McMahan et al., 2016) 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 constrained. 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.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Nonetheless, a well-known challenge for FL is that the local datasets on clients are not identically and independently distributed (i.e., being non-IID). Compared to centralized training with the datasets combined, FedAvg with non-IID datasets suffers salient accuracy loss. When training a CNN model with 100 clients, we observe an accuracy loss of up to 77% (§2.2). With both theoretical and experimental explorations, we find that model training using FedAvg will stagnate prematurely with suboptimal parameters, a phenomena we call *premature stagnation*. Moreover, the accuracy loss under FedAvg has a strong correlation with the synchronization frequency: A higher accuracy would require a higher frequency. Therefore, there is a clear trade-off in setting the FedAvg synchronization frequency: a lower frequency reduces communication cost but compromises the accuracy performance. To make FL both accurate and efficient, we need to strategically increase the FedAvg synchronization frequency during the FL process.

The key question is, when and how to tune the synchronization frequency? While some related works (Wang & Joshi, 2019; Wang et al., 2019) also focus on frequency tuning, we find that they neglect the distinct characteristics of FL and thus fall short (§4). Specifically, we summarize three design requirements for frequency tuning in FL. First, under the privacy constraint, frequency tuning decisions shall be made purely based on *gradients*—the only information allowed to be collected from clients. That is, any other information, e.g., local data distributions, local loss values or accuracies, cannot be used. Second, since many clients may dynamically join or leave the FL process, our solution should work smoothly with excellent stability and scalability. Moreover, our frequency tuning method should be generally applicable regardless of the specific model or dataset distribution pattern. To quickly recap, our objective in this work is to design a *privacy-preserving*, *practical*, and *generic* frequency tuning method for FedAvg so that it performs well in both accuracy and efficiency.

Our first challenge is to find a gradient-based signal to indicate the instantaneous training status, which can be used to inform the synchronization frequency tuning. We find that the consistency of gradients from different clients is a good candidate. Our theoretical analysis shows that the gradient from each client is composed of a global component and a local-error one: the former is identical for all the clients, while the latter differs as it is determined by the local sample distribution. When training under a given synchronization frequency, the global component slowly shrinks and the local one gradually dominates; finally the FL process enters a stagnation stage where gradients from different clients well counteract with each other. Therefore, by measuring the gradient consistency level, we can effectively gauge the model training status with privacy preserved.

Furthermore, considering that massive clients may dynamically participate in training with noisy samples, we must ensure that our metric is *practical*. That is, our metric should be robust to the systematic and statistical disturbance and be computation- and memory-efficient despite the vast client number. Using a series of smoothing and pooling techniques, we propose a novel metric called *Gradient Consistency*, which effectively reflects how fast the model parameters can move towards the true optimum.

Based on gradient consistency, we devise a simple yet effective frequency tuning algorithm called *Gradient-Instructed Frequency Tuning* (GIFT). While some related works (Wang & Joshi, 2019; Wang et al., 2019) choose to calculate the ideal frequency, they require rigorous assumptions (e.g. convexity and smoothness of the loss function or certain data distribution pattern) to work, which limits the applicability of the solution in cases with complex models or arbitrarily non-IID data. To be widely applicable, our GIFT algorithm is a simple and generic heuristic: once the gradient consistency stabilizes at a small value close to zero (indicating the occurrence of premature stagnation), we increase the synchronization frequency by a fixed scaler. We further extend the solution to allow the frequency to be slightly decreased when frequent synchronization is less 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. Our evaluation shows that GIFT can improve the model accuracy and resource efficiency substantially: for example, it can improve the convergence accuracy of VGG-16 by 10.7% with a time reduction of 58.1% (after a fixed number of synchronizations). Meanwhile, compared to existing frequency-tuning methods, GIFT saves the training time by 28.9% given the same accuracy target in addition to the privacy-preserving and practicality benefits.

Our contributions in this work are three-fold. First, we identify the correlation between synchronization frequency and model accuracy for FedAvg with both mathematical proofs and testbed measurements. Second, with a novel metric of Gradient Consistency we propose GIFT, a privacy-preserving, practical, and generic frequency-tuning heuristic that makes FL accurate and efficient. Third, we have implemented GIFT in PyTorch and evaluated it in large-scale testbed with realistic setups.

2. Research Background

2.1. A Primer on Federated Learning

Machine learning (ML) models, such as deep neural networks, are widely used for a range of applications to attain state-of-the-art performance (Krizhevsky et al., 2012; Collobert et al., 2011; Sutskever et al., 2014). However, in many real-world scenarios, training samples are privacy-sensitive and dispersed on distributed clients like IoT devices, cellphones, banks and hospitals (McMahan et al., 2016; Zhang et al., 2020). To train models without centralizing such private data, an increasingly popular technique is federated learning (FL) (Konečnỳ et al., 2016; McMahan et al., 2016), under which clients locally refine the model parameters with their private data and only communicate the model *updates* to the FL server. Compared with centralized model training, FL is confronted with two distinct performance challenges:

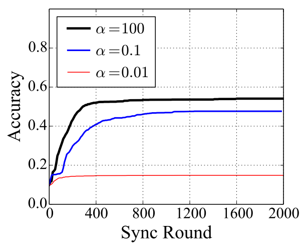
1) Privacy Constraint. To protect data privacy, the information exchanged between clients and the FL server can only be model *updates* (Bonawitz et al., 2019), and meanwhile samples cannot be shuffled across clients. Since the local samples of an IoT device or edge user are generated under particular location environment or user preference, the samples on different clients are usually *not* identically and independently distributed (i.e., *being non-IID*).

2) Inferior Resources. Compared with dedicated computing servers, FL clients like IoT devices or cellphones suffer from remarkable limitation in computation and communication resources. In particular, given the limited bandwidth between clients and the FL server, model update synchronization is often a severe performance bottleneck. Moreover, in realistic scenarios the clients may dynamically join or leave the FL process (Bonawitz et al., 2019).

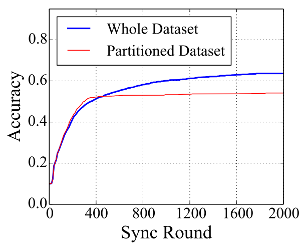
FedAvg To reduce the communication cost, the *FedAvg* algorithm (Konečnỳ et al., 2016; McMahan et al., 2016) has become the *de facto* FL mechanism, which dictates each client to perform *multiple* local iterations before synchronizing their updates. Yet, due to the non-IID data, FedAvg on the other hand would compromise the model convergence accuracy. Next we show the severity and omnipresence of this problem with testbed measurements emulating realistic data distribution patterns.

2.2. Measurements on FedAvg Accuracy Loss

There have been a series of research works (Zhao et al., 2018; Wang et al., 2019) revealing FedAvg accuracy loss under non-IID data setup, usually by partitioning the entire dataset according to sample label class in a non-overlapping manner. Yet, this setup is not realistic: samples



(a) Impact of non-IID Label Distribution (A smaller α from the Dirichlet distribution means a severer non-IID level.)



(b) Impact of Feature Insufficiency (Partitioned Dataset means to randomly partition the dataset over 100 clients.)

Figure 1: Test accuracy of LeNet-5 when trained under FedAvg with non-IID data. Non-IID label distribution causes severe accuracy loss, and insufficient feature diversity also matters.

on each client are in fact generated *independently*, and their label class compositions often overlap (although the concrete composition ratio may be different). Moreover, existing measurements treat "non-IID data" simply as "non-IID labels" and ignore the impact of sample features. Our experimental setup in this part will make up in both aspects.

Non-IID Evaluation in a Realistic Manner Instead of by partitioning the initial dataset, we independently draw (with replacement) each client's samples following the Dirichlet distribution (Yurochkin et al., 2019; Hsu et al., 2019), which controls the class composition via a concentration parameter α. With $\alpha \to \infty$, the client holds samples evenly from each class; with $\alpha \to 0$, the client holds samples only from one class. This setup can faithfully emulate the scenario that a user gradually accumulates local samples with certain preference. With this methodology, we train the LeNet-5 model (LeCun et al., 1998) on CIFAR-10 dataset (Krizhevsky & Hinton, 2009) with 100 clients (with a synchronization frequency of once-per-100-iterations; please refer to §6.1 for the detailed setup). As shown in Fig. 1a, there is a clear relationship between the final accuracy and data non-IID level. In particular, when $\alpha = 0.01$ (under which 91% clients host no more than two classes each), the accuracy is only 0.15, suffering a loss of 77% compared with that of centralized training (0.65).

Omnipresence of Non-IID Data due to Feature Insufficiency In the literature, non-IID data usually refers to non-IID labels, yet we find that the impact of non-IID problem is much more common than that. When many small datasets compose a large virtual dataset as in FL, there also exists the non-IID problem even if each partition has identical label composition. In fact, sample features are also part of the loss function; due to the shortage of feature diversity, a small partition may fail to represent the full dataset even when its class composition resembles the global one. To verify that, we evenly partition the CIFAR-10 dataset to 100 clients ensuring that they share the same class distribution. Compared to the case where each client holds the whole dataset, as shown in Fig. 1b, there is still an accuracy loss of 16%. Therefore, the non-IID problem is in fact omnipresent for FL irrespective of the label distribution.

Observing the severity and omnipresence of the negative impact of non-IID data on FedAvg, we ask: why there is such accuracy loss and how to mitigate it? In the following section, we explore the answers through both theoretical analysis and testbed measurements.

3. Motivative Exploration

In this section, we first build a mathematical model on the factors causing the accuracy loss under FedAvg, and then confirm our theoretical findings with a toy example as well as testbed measurements.

Symbol Description In our FL setup, there are N clients each with a local dataset D_i ($i = 1, 2, ..., N$). 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 (the true optimization target assuming full IID dataset) is $L^\star(\omega) = \frac{1}{|\cup_{D_i}|}\sum_{s\in\cup_{D_i}} l(s,\omega)$. Let ω_k^i be the local parameter on client-i 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 convexity and smoothness assumptions (described in appendix) we can derive the following lemma:

Lemma 1 (Two Components in Local Gradients)**.** *Let $u_\tau^i = \omega_\tau^i - \omega_0$ be the accumulated gradients*[1] *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 is in appendix. This lemma implies that the gradient error of each FL client is related to the gap between its local loss landscape (i.e., $\nabla L^i(\omega)$ and $\nabla^2 L^i(\omega)$) and the global one. Moreover, regarding the aggregated gradient after a synchronization round, with Lemma 1 we further have the following Theorem 1:

Theorem 1 (Error of Aggregated Gradient After a Round)**.** *Let $\bar{u}_\tau = \frac{1}{N}\sum_{i=1}^N u_\tau^i$ and $d(\tau) = \bar{u}_\tau - u_\tau^\star$ be the error of aggregated gradient 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] While the synchronized content in FL can be parameters instead of gradients, they are essentially equivalent as explained in (McMahan et al., 2016). Meanwhile, by *gradient*, we refer to the *accumulated update* in a round with 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 be extended to other SGD variants like Adam (Diederik P. Kingma, 2017) and AdaGrad (Duchi et al., 2011).

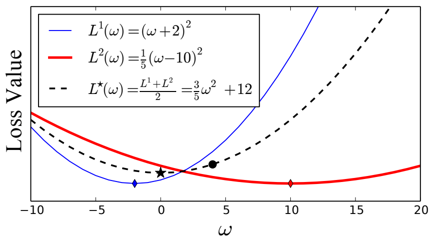


Figure 2: During the local iterations, each FL client may already reach its local optimum (−2 and 10), yielding an aggregated parameter of 4 instead of the true optimum 0. This procedure repeats in each round, indicating the occurrence of premature stagnation.

Theorem 1 shows that the error of aggregated gradient is determined by two factors: first by the inconsistency of clients' loss *curvatures* ($\nabla^2 L^i(\omega_0)$, which crucially affects $\| \sum_{i=1}^N \langle \nabla^2 L^i(\omega_0), \nabla L^i(\omega_0) - \nabla L^\star(\omega_0) \rangle \|$ given that $\sum_{i=1}^N [\nabla L^i(\omega_0) - \nabla L^\star(\omega_0)] = 0$), and second by the number of local iterations within a round (τ). Regarding the model convergence status, we derive Theorem 2:

Theorem 2 (Premature Stagnation)**.** *Suppose the FL process ultimately stagnates at parameter $\bar{\omega}^\star$, and $\omega^\star = \arg\min L^\star(\omega)$ is the ideal parameter under IID dataset. Then $\exists C > 0$ such that $\| \bar{\omega}^\star - \omega^\star \| \geq C(\tau - 1)$.*

This theorem shows that the FL process would stagnate with suboptimal parameters, a phenomena we call *premature stagnation*. To better elaborate premature stagnation, we first resort to a toy example with quadratic loss functions, and then experimentally show that frequency tuning can help to get out of premature stagnation.

A Toy Example for Premature Stagnation Our example is shown in Fig. 2 where there are two clients with different loss functions (with unequal curvatures): $L^1(\omega) = (\omega+2)^2$ and $L^2(\omega) = \frac{1}{5}(\omega-10)^2$. The ideal loss function with IID data is $L^\star(\omega) = \frac{1}{2}[L^1(\omega) + L^2(\omega)] = \frac{3}{5}\omega^2 + 12$, with the optimal parameter $\omega^\star$ be 0. Yet, during the local iterations each client is essentially refining its parameter towards the local optimum (−2 and 10, respectively), and a late synchronization may yield an aggregated parameter of 4. Such a process would repeat in the following rounds, meaning that FL stagnates at a suboptimal state.

Effect of Frequency Tuning As indicated by Theorem 2, an intuitive idea to combat premature stagnation is by increasing the synchronization frequency (i.e., reducing τ). We experimentally verify this with both convex (SVM) and non-convex (LeNet-5) models. The SVM model is trained upon 50,000 randomly-generated points of 2 classes, with two clients each holding only 1 class; the LeNet-5 model is trained upon CIFAR-10, also with 2 clients each holding only 5 classes. The initial τ is 500, and once premature stagnation occurs we change it to 1. In Fig. 3, for each model we depict the variation of a randomly-selected parameter as well as the instantaneous model accuracy. After increasing the synchronization frequency, with higher-quality gradients, the parameter can get out of stagnation and achieve better model accuracy.

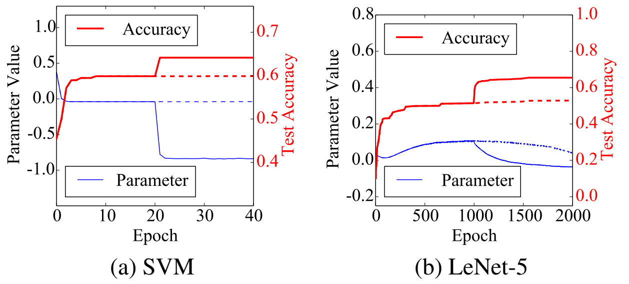


Figure 3: For FL with two clients over non-IID data, by increasing the synchronization frequency (at epoch 20/1000), the sampled global parameter can get closer to the true optimum, and the models can attain a higher accuracy. The dashed lines show the measured variables under the initial frequency for reference.

To summarize, in this section we find that a higher FL accuracy requires a higher synchronization frequency. Yet, a higher frequency would on the other hand sacrifice the communication efficiency. Therefore, to make FedAvg accurate and also efficient, we need to strategically increase the synchronization frequency during the training process. Next we survey the related works on that topic.

4. Prior Arts and Their Limitations

On addressing the statistical heterogeneity (i.e., non-IID data) in FL, existing solutions can be broadly categorized into three types: data complementing, optimization rectifying and frequency tuning.

Data Complementing Data complementing means to reduce data non-IID level by copying common samples to each client (Zhao et al., 2018), or by augmenting clients' local datasets with auxiliary samples generated from GAN models (Jeong et al., 2018). Yet these methods may incur large computation, communication or storage overheads.

Optimization Rectifying To mitigate the training divergence among clients with non-IID data, Li et al. (2018) and Jeong et al. (2018) proposed to add an extra regularizing term to the loss function, and another work (Li et al., 2019) proposed to add a specific momentum to the optimizer. Meanwhile, Huang et al. (2018) introduced a two-phase training mechanism: clients receive the average loss as the feedback, and only those with a loss value large enough can enter a second phase to refine the global model. Yet, these methods require substantial modifications to the initial FL process and, as suggested by Theorem 1, still suffer accuracy loss due to inconsistent loss curvatures.

Frequency Tuning As elaborated in §3, frequency tuning is a promising approach to make FedAvg accurate and also efficient, which is the research focus of this paper. In the literature, there have been a series of works (Wang &

Joshi, 2019; Wang et al., 2019; Haddadpour et al., 2019) on tuning the accuracy-efficiency trade-off by frequency control. Although with various assumptions (e.g. convex or non-convex, IID labels or non-IID labels) or optimizing targets (e.g. communication rounds or time cost), they do share a common methodology: First deriving a formula depicting the exact relationship between the synchronization frequency and the metric to optimize, and then work out the ideal frequency by solving that optimization problem.

However, we argue that such *formula-based* frequency-tuning methods are deficient in three aspects:

1) Privacy. As elaborated in §2, the privacy constraint of FL stipulates that only model updates can be transmitted, and other client-side information such as local loss or accuracy values can not be collected by the FL server. Nonetheless, the formulas in those works require collecting the local loss value of each client to calculate the ideal frequency. The AFL algorithm (Wang et al., 2019) even demands the knowledge of local data distributions, which severely violates FL's privacy constraint.

2) Practicality. A well-known system challenge for FL is that clients may dynamically join or leave the training process at random time (Bonawitz et al., 2019). Meanwhile, in realistic FL applications like GBoard (Yang et al., 2018), hundreds or thousands of clients may simultaneously participate in FL, incurring a scalability challenge. However, existing formula-based methods ignore such challenges and may suffer instable performance and large computing or storage overhead.

3) Generality. Although theoretically sound, the solution validity of those works largely relies on a series of rigorous assumptions and ground-truth knowledge. For example, AdaComm (Wang & Joshi, 2019) and AFL (Wang et al., 2019) derive their solutions with convexity assumptions not holding for deep neural networks; meanwhile, AdaComm and LUPA-SGD (Haddadpour et al., 2019) assume IID label distributions and AFL assumes a bounded data distribution divergence; moreover, AFL requires a series of ground-truth knowledge like the smoothness constant, the Lipschitz constant as well as the gradient variance bound, which are hard to obtain in reality. Dependence on such assumptions and ground-truth knowledge often impede the applicability of these methods in real-world FL scenarios; we argue that theoretical analysis is more appropriate for qualitative elaboration instead of for direct calculation.

Objective Motivated by the above discussions, our objective in this paper is to design a *privacy-preserving*, *practical* and *generic* frequency tuning algorithms for FedAvg that can make FL accurate and also resource-efficient. In the next section, we will elaborate our solution with the three design requirements simultaneously satisfied.

5. Gradient-Instructed Frequency Tuning

In this second, we propose Gradient-Instructed Frequency Tuning (GIFT), a privacy-preserving, practical and generic frequency tuning algorithm for efficient and accurate FL. We first show that gradient statistics is an appropriate angle to learn FL training status, and then propose a practical metric called Gradient Consistency, based on which we finally devise a generic heuristic for frequency tuning.

5.1. Learn Training Status From Gradient Statistics

Our first challenge is to find an effective signal that can reveal instantaneous training status under the FL privacy constraint. On the one hand, client-side information like local loss or accuracy values or local data distributions cannot be remotely collected, and on the other hand, in realistic scenarios with evolving trends, it is hard to maintain an up-to-date validating dataset on the FL server. Therefore, the only information we can utilize are the *gradients*.

Then, can we really learn the training status from gradients? Yes we can. While an individual gradient may exhibit strong randomness, we find that there actually exists a clear statistical pattern for gradients across different clients. In fact, we have observed an interesting phenomena called *gradient bifurcation*, meaning that the gradients from different FL clients are consistent in the beginning but conflicting later as training proceeds. We next elaborate this phenomena with theoretical explanations, toy examples as well as testbed measurements.

Gradient Bifurcation 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 component 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 where the parameter is far away from the optimum, the loss function landscapes (e.g., $\nabla^2 L(\omega)$) are similar across different clients (due to $L(\omega)$ smoothness or equivalently $\nabla L(\omega)$ Lipchitz-ness), meaning that $u_\tau^\star$ 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) while the heterogeneity of loss landscape (e.g., the curvature $\nabla^2 L(\omega)$) amplifies, meaning that it is the error component e_τ^i that dominates u_τ^i. Consequently, gradients from different clients would gradually *bifurcate* in a FL process.

The gradient bifurcation phenomena can be illustrated also with Fig. 2. 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; yet when the parameter moves across -2, the gradient of client-1 would become negative, conflicting with that of client-2.

We further verify gradient bifurcation with testbed mea-

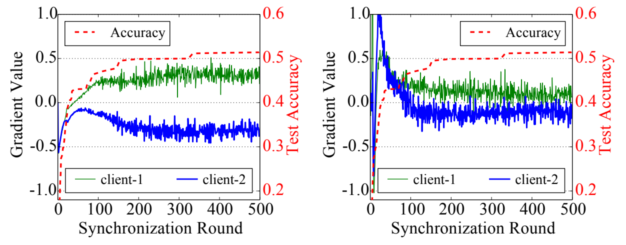


Figure 4: When training LeNet-5 with 2 clients under FedAvg, the gradients of two randomly-chosen parameters gradually bifurcate.

surements. We train the LeNet-5 model following the setup in Fig. 3b (two clients with non-IID data), and measure the instantaneous gradient values of two randomly-selected parameters on both clients. As depicted in Fig. 4, the gradients for both parameters would bifurcate after around round-100, the extent of which exhibits a clear correlation with the model convergence status (expressed in accuracy).

To summarize, gradient bifurcation can effectively signal the FL training status without privacy leakage. Yet, realistic FL may be much more complex than the testbed setup of Fig. 4, exhibiting distinct stability and scalability challenges. Next, we propose a practical metric to quantify the level of gradient bifurcation for real-world FL scenarios.

5.2. Gradient Consistency: A Practical Metric

To quantify the extent of gradient bifurcation, we propose an intuitive metric: $C = \frac{\|\sum_i u_\tau^i\|}{\sum_i \|u_\tau^i\|}$. 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 premature stagnation occurs. Nonetheless, this metric is not practical for real-wold FL. To be clear, we summarize the practicality challenges of FL as follows:

1) Stability Challenge. Since samples processed in each SGD iteration are chosen *randomly*, client gradients often fluctuate drastically, as can be seen in Fig. 4. Such *statistical instability* may inundate the desired gradient patterns. Besides, in real-world FL setup the active clients are also unstable: FL clients may join or leave the federation randomly during training, and meanwhile, the FL server usually collects gradients from only a portion of the clients whoever reporting the earliest, so as to avoid waiting for stragglers (Bonawitz et al., 2019). A practical metric must be robust to such statistical and system instability.

2) Scalability Challenge. In commercial FL applications the number of clients may be quite large. Our metric should scale well in computing or storage overhead.

To tackle those challenges, we incorporate smoothing and pooling techniques in our metric design.

Smoothing To address statistical instability, we smooth the raw gradients with their historical values. To maintain low storage overhead, instead of window-based smoothing method, we calculate their *exponential moving average* (EMA). That is, let $u_{\tau,r}^i$ be the local gradient of client-i in r^{th} round, we maintain $\tilde{u}_{\tau,r}^i = \text{E}_{\text{MA}}(u_{\tau,r}^i) = \beta * \tilde{u}_{\tau,r-1}^i + (1-\beta) * u_{\tau,r}^i$ instead of $u_{\tau,r}^i$.

Pooling Given the system instability and the large client quantity, it is nonetheless memory-inefficient and even infeasible to maintain $\tilde{u}_{\tau,r}^i$ for each client. Therefore, we further propose *bilateral gradient pooling*. That is, the FL server only maintains two EMAs: one to absorb the *positive* gradient values from any client, and the other the *negative* ones. This way, we can get a stable gradient statistics pattern despite the unstable client participation.

Combining the above smoothing and pooling techniques, we define our metric, *Gradient Consistency*, as:

$$C_r = \frac{\|\tilde{P}_r + \tilde{N}_r\|}{\|\tilde{P}_r\| + \|\tilde{N}_r\|}, \text{ where } \begin{cases} \tilde{P}_r = \text{E}_{\text{MA}}(\sum_i \text{Relu}(u_{\tau,r}^i)) \\ \tilde{N}_r = \text{E}_{\text{MA}}(\sum_i \text{-Relu}(\text{-}u_{\tau,r}^i)). \end{cases}$$

Gradient Consistency is therefore 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 true optimum. In particular, it indicates the occurrence of premature stagnation by stabilizing to a small value close to zero[2]. Next we elaborate how to tune the synchronization frequency with Gradient Consistency.

5.3. A Generic Frequency Tuning Algorithm

While it might be possible to set τ by building a formula on the ideal Gradient Consistency under a given setup, this would inevitably requires a series of rigorous assumptions and ground-truth knowledge as in the formula-based related works (§4), which however limits its applicability. Therefore, to make our solution generally applicable for various models or dataset distributions, we propose a simple heuristic: once Gradient Consistency suggests that premature stagnation occurs, we divide τ by a scaler (e.g., 2). With a higher frequency, it can be expected that the FL process can reach a higher accuracy before it saturates again.

Moreover, we also incorporate an extension that allows the synchronization frequency to be modestly decreased at the beginning of the FL process. As implied in Theorem 1, the gradient error is correlated to both τ and the heterogeneity of loss surface curvatures $\{\nabla^2 L^i(\omega)\}$. Since at the FL commencement $\{\nabla^2 L^i(\omega)\}$ are usually similar among clients, there is a relatively large tolerance for τ. For example, if in Fig. 2 the gradients from two clients are both positive, it is not that necessary to do frequent synchronization. To explore such optimization space, we *tenta-*

[2] When stabilizing, it may not reach zero exactly because gradient fluctuation can not be fundamentally eliminated with EMA.

tively increase τ every a few rounds, in a linear manner until premature stagnation occurs. Yet, since relaxing τ may hurt the computation efficiency (due to lower-quality gradients) and the initial τ is usually set large in practice, we enable frequency relaxing optionally only when optimizing the number of communication rounds.

Combining the Gradient Consistency metric with the above frequency tuning heuristic, we name our solution-kit as *Gradient-Instructed Frequency Tuning*, or GIFT. We have implemented GIFT in PyTorch, and next we evaluate its performance with prototype deployments.

6. Evaluation

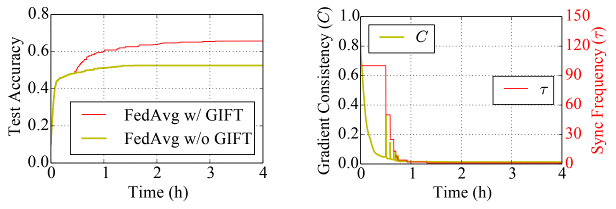
In this section, we evaluate GIFT with large-scale testbed experiments. We start with end-to-end comparisons between GIFT and standard FedAvg in a 100-node cluster emulating realistic FL setup, and then justify its superiority over existing formula-based methods. Finally we examine the effectiveness of the frequency relaxation extension.

6.1. Experimental Setup

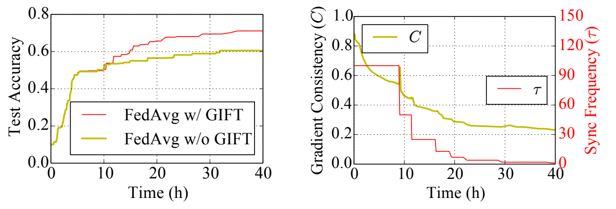
Hardware Setup We emulate real-world FL scenarios with 100 `m5.large` instances on Amazon EC2, each with 2 vCPU cores and 8GB RAM (similar with a smart phone). Meanwhile, resembling modern Internet condition[3], the bandwidth of each client is configured to be 25Mbps. The FL server is a `c5.9xlarge` instance with 10Gbps bandwidth. Moreover, to emulate dynamic user participation, 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.

Training Setup Models trained in our evaluation are LeNet-5 (LeCun et al., 1998), VGG-16 (Simonyan & Zisserman, 2014) 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 (Krizhevsky & Hinton, 2009) and the LSTM network is trained on the KeyWord Spotting (KWS) dataset—a subset of the Speech Commands dataset (Warden, 2018) including 10 key words. Similar to §2.2, the local dataset on each client is $\frac{1}{100}$ in size of the standard one, in which the sample labels follow a Dirichlet distribution with α set to 1 (a modest non-IID level). We set the learning rates to 0.01 (LeNet-5), 0.1 (VGG-16) and 0.05 (KWS), with a respective weight decay of 0.01, 0.0005 and 0.01. The default frequency is once-per-100-iterations (i.e., $\tau = 100$). The EMA smoothing factor β in GIFT is 0.9, and we halve τ once Gradient Consistency no longer decreases for two consecutive rounds.

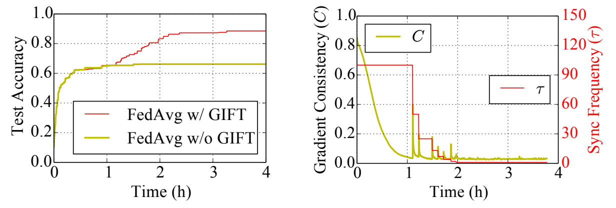
[3]https://www.atlasandboots.com/remote-work/countries-with-the-fastest-internet-in-the-world/, Accessed: 2021-02-04



(a) LeNet-5



(b) VGG-16



(c) LSTM

Figure 5: Test accuracy with and without GIFT, accompanied by the frequency τ and Gradient Consistency C under GIFT.

Table 1: Time Cost and Accuracy Attained 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

6.2. End-to-End Evaluation

Visual Observation In Fig. 5, we show the (best-ever) test accuracy when training the three models under FedAvg (with and without GIFT), accompanied by the instantaneous *Gradient Consistency* (C) and synchronization frequency (τ) of GIFT. It shows that there is a salient accuracy enhancement once τ is halved under GIFT, finally yielding a much better accuracy. Meanwhile, we notice that C does follow a decreasing pattern[4], consistent with our previous analysis in §5.1. Moreover, there is usually a short burst of C each time τ is halved, suggesting that a higher frequency can mitigate gradient conflicting level and help refine model parameters consistently towards the optimum.

[4]Interestingly, for VGG-16 the stagnating value of C in Fig. 5b is not close to 0. We find that this is because VGG-16 is a large, over-parameterized model (Neyshabur et al., 2018) and, due to irregular landscapes like flat minima (Hochreiter & Schmidhuber, 1997), some parameters may keep moving to a certain direction even after model converges. That said, it does not affect the effectiveness of GIFT because C still stagnates as expected.

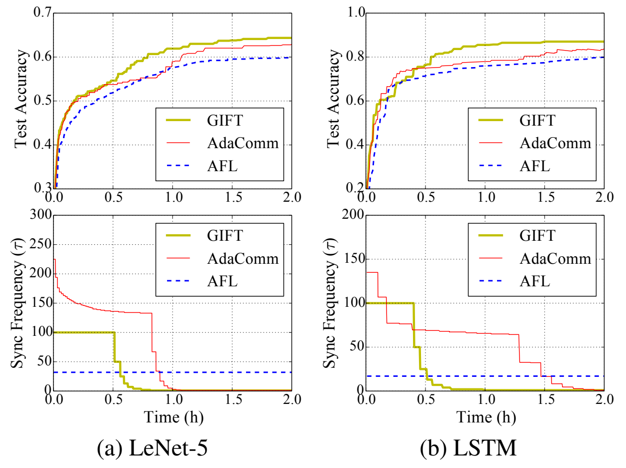


Figure 6: Comparison among GIFT and formula-based methods.

Quantitative Comparison We further make quantitative performance comparison between GIFT and standard FedAvg in Table 1, which lists the time cost and accuracy attained when training each model for 1000 rounds. Table 1 reveals that the performance benefit of GIFT lies in two perspectives. First, GIFT can help the FL process to achieve a higher accuracy by getting out of premature stagnation timely. Second, GIFT can also improve the computation efficiency by switching to a higher frequency when appropriate: Otherwise with an over-large τ the local parameters often saturate and then oscillate locally, which is a wastage of computing power. As a result, under GFT, each model can achieve a higher accuracy with less time consumption. For example, GFT can reduce LeNet-5 training time by 58.9% while improving the accuracy by 9%.

6.3. Comparison with Formula-based Methods

We implement AFL (Wang et al., 2019) and AdaComm (Wang & Joshi, 2019), two typical formula-based frequency tuning methods, also in PyTorch. AFL directly works out a fixed τ given the resource budget and a series of ground truth knowledge (like the Lipchitz constant and gradient variance bound); AdaComm divides the training process into short intervals and adjust τ for each interval based on variation of the instantaneous loss. In our evaluation, we obtain the ground-truth knowledge required by AFL via a trial run, and the initial frequency of AdaComm is selected via grid search. Moreover, since such formula-based methods do not consider system instability (e.g., partial client participation), for fair comparison we switch to a 50-node cluster with full client participation.

Fig. 6 shows the instantaneous accuracy and synchronization frequency when training LeNet-5 and LSTM under GIFT, AFL and AdaComm. It shows that GFT can yield the best model accuracy under a fixed time budget (2 hours). For example, given a LeNet-5 accuracy target of 0.6, the time consumption under GFT is 28.9% less than AdaComm, and 61.2% less than AFL. 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 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 loss not so), the training loss is sometimes not a good indicator of the training status[5].

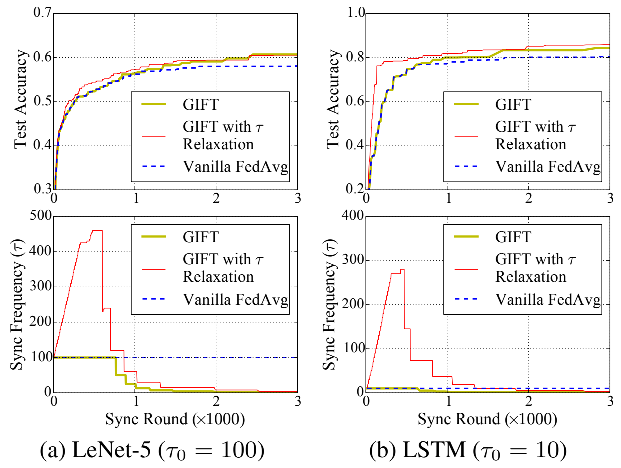


Figure 7: Frequency relaxation can yield faster accuracy increase.

6.4. Effect of Frequency Relaxation

We further evaluate the effectiveness of frequency relaxation in cases where only the communication cost is cared. To be specific, we 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 previous 50-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.

7. Conclusion

In this work, to attain better model accuracy and resource efficiency in FL, we have proposed GIFT, a privacy-preserving, practical and generally-applicable frequency tuning scheme. It 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 setups have demonstrated the superiority of GIFT in both accuracy and efficiency performance.

[5]Considering that the training loss may get stuck on plateaus or fluctuate due to noises, AdaComm degrades their method to naive frequency scaling once the loss no longer decreases. Yet we find in Fig. 6 that the accuracy achievements of AdaComm largely comes from such a boundary-case heuristic.

References

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. In *Proc. SysML*, 2019.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

Diederik P. Kingma, J. B. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Haddadpour, F., Kamani, M. M., Mahdavi, M., and Cadambe, V. R. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. In *NuerIPS*, 2019.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

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

Huang, L., Yin, Y., Fu, Z., Zhang, S., Deng, H., and Liu, D. Loadaboost: Loss-based adaboost federated machine learning on medical data. *arXiv preprint arXiv:1811.12629*, 2018.

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

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

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

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

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Li, C., Li, R., Zhou, P., Wang, H., Li, Y., Guo, S., and Li, K. Gradient scheduling with global momentum for non-iid data distributed asynchronous training. *arXiv preprint arXiv:1902.07848*, 2019.

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

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

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

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

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

Wang, J. and Joshi, G. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. In *SysML*, 2019.

Wang, S., Tuor, T., Salonidis, T., Leung, K. K., Makaya, C., He, T., and Chan, K. Adaptive federated learning in resource constrained edge computing systems. *Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

Warden, P. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Yang, T., Andrew, G., Eichner, H., Sun, H., Li, W., Kong, N., Ramage, D., and Beaufays, F. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

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

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

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., and Chandra, V. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.