Towards Flexible Device Participation in Federated Learning

Yichen Ruan Carnegie Mellon University
Xiaoxi Zhang Sun Yat-Sen University
Shu-Che Liang Carnegie Mellon University
Carlee Joe-Wong Carnegie Mellon University

Abstract

Traditional federated learning algorithms impose strict requirements on the participation rates of devices, which limit the potential reach of federated learning. This paper extends the current learning paradigm to include devices that may become inactive, compute incomplete updates, and depart or arrive in the middle of training. We derive analytical results to illustrate how allowing more flexible device participation can affect the learning convergence when data is not independently and identically distributed (non-IID). We then propose a new federated aggregation scheme that converges even when devices may be inactive or return incomplete updates. We also study how the learning process can adapt to early departures or late arrivals, and analyze their impacts on the convergence.

1 Introduction

Federated learning is a cutting-edge learning framework that allows distributed devices to train a shared machine learning model cooperatively without sharing the raw data. In recent years, federated learning has exhibited remarkable performance in many applications such as next word suggestion, fault detection, and learning on private medical data (Li et al., 2020a). Generic federated learning involves a coordinator and a collection of devices. The training procedure consists of multiple rounds, each of which includes the following three steps: 1) *Synchronization*: the coordinator synchronizes the latest *global model* with all devices. 2) *Local updates*: each device trains a *local model* for a few *local epochs*, using samples from its *local dataset*. 3) *Aggregation*: the coordinator aggregates some, or all, of the local models to produce the next global model.

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

Our work focuses on cross-device federated learning (Kairouz et al., 2019), where participating entities are mostly mobile devices such as smart phones and tablets. These devices generally have limited computing and communication resources, e.g., due to battery limitations, and have different training data distributions, i.e., data is not independently and identically distributed (non-IID) among devices (Li et al., 2020b). To relieve the computation and communication burden, in the last step of the training procedure, the federated learning coordinator may only aggregate a subset of local models. However, only a few device selection policies ensure convergence in the non-IID setting, and the selection must be independent of the hardware status of devices (Li et al., 2020b). In other words, for the training to converge successfully, all selected devices must be able to train their local models and upload the results whenever they are selected. This is why the traditional federated learning paradigm requires participating devices to be dedicated to the training during the entire federated learning period, e.g., the popular *FedAvg* algorithm assumes mobile users will participate only when their phones are currently plugged-in, and have unlimited WI-FI access (McMahan et al., 2016).

Considering that federated learning typically takes thousands of communication rounds to converge, it is difficult to ensure that all devices will be available during the entire training in practice. Moreover, there are typically multiple apps running simultaneously on user devices, competing for already highly constrained hardware resources. As such, it cannot be guaranteed that devices will complete their assigned training tasks in every training round as expected. A similar challenge also arises in cloud based distributed learning due to the increasingly popular usage of preemptive cloud services, where the user process can be interrupted unexpectedly (Zhang et al., 2020).

While many methods have been proposed to mitigate the workload of individual devices, such as weight compression and federated dropout (Caldas et al., 2018)(Konečnỳ et al., 2016), they cannot completely remove the possibility that devices are unable to fulfill their training responsibilities, e.g., due to poor wireless

connectivity. Thus, in large scale federated learning, many resource-constrained devices have to be excluded from joining federated learning in the first place, which restricts the potential availability of training datasets, and weakens the applicability of federated learning. Furthermore, existing work does not specify how to react when confronting unexpected device behaviors, and also does not analyze the (negative) effects of such behaviors on the training progress.

In this paper, we relax these restrictions and allow devices to follow more flexible participation patterns. Specifically, the paper incorporates four situations that are not yet well discussed in the literature: 1) *Incompleteness*: devices might submit only partially completed work in a round. 2) *Inactivity*: furthermore, devices might not complete any updates, or respond to the coordinator at all. 3) *Early departures*: in the extreme case, existing devices might quit the training without finishing all training rounds. 4) *Late arrivals*: apart from existing devices, new devices might join after the training has already started.

The difference between inactivity and departure is that inactive devices will temporarily disconnect with the coordinator, but are expected to come back in the near future. In contrast, departing devices will inform the coordinator that they do not plan to rejoin the training. For example, if a user quits the app running federated learning, a message can be sent to the coordinator; the coordinator thus knows who is departing. In the meanwhile, although devices' arriving and departing seem symmetric, they affect the model training differently, and thus require distinct treatments. The key difference is that arriving devices offer extra information about the data distribution, which can be utilized to accelerate the training, while departing devices reduce our available knowledge, thus degrading the applicability of the trained model.

Our approach to improve the flexibility of device participation comprises the following components that supplement the existing *FedAvg* algorithm and handle the challenges brought by flexible device participation.

- **Debiasing for partial model updates**. *FedAvg* aggregates device updates as a weighted sum, with weights that are proportional to the sizes of the local datasets. This choice of aggregation coefficients yields an unbiased gradient as in the centralized setting only when all data points from all devices are equally likely to join the learning (Li et al., 2020b). However, it in general fails to guarantee convergence to the globally optimal point in the presence of partial aggregation from incomplete and inactive devices. We show that by *adapting the aggregation coefficients*, the bias can be reduced and the convergence to a global optimum can still be established. Furthermore, our analysis shows the bias originates from the heterogeneity in device participation, as well as from the degree to which local datasets are not IID.
- **Fast-rebooting for device arrivals**. Arriving devices interrupt the training by forcing the model to re-orient to the new device's data, thus slowing the convergence process. In this paper, we propose to rapidly reboot the training in the case of device arrivals by applying *extra updates* from the new devices. Intuitively, since an arriving device misses all previous epochs, the model training should emphasize more on its updates to compensate. We will rigorously prove this method indeed expedites learning convergence under certain conditions.
- **Redefining model applicability for device departures**. A model successfully trained by federated learning is expected to be applicable to the data from all participating devices. However, when a device withdraws itself from the learning, due to the lack of its future updates, we may no longer require the trained model to perform well on its data. It is then important to redefine the model's applicability. Namely, one can either keep the departing device as a part of the global learning objective, or exclude it to focus only on the remaining devices. The decision depends on which definition yields smaller training loss. We will show the key to this determination lies in the *remaining training time*.

In Section 2, we review relevant literature. In Section 3, we give a convergence analysis that incorporates flexible device participation. Based on this analysis, we detail our contributions, as outlined above, in Section 4, and we experimentally verify our theoretical results in Section 5. Finally we conclude in Section 6.

2 Related Works

The celebrated federated learning algorithm named *FedAvg* runs the stochastic gradient descent (SGD) algorithm in parallel on each device in the system and periodically averages the updated parameters from a small set of end devices. However, its performance degrades when the local data is non-IID (Hsieh et al., 2019)(Zhao et al., 2018). A few recent works provide theoretical results for the non-IID data case. For instance, Li et al. (2020b) analyze the convergence of *FedAvg* on non-IID data and establish an $O(\frac{1}{T})$ convergence rate for strongly convex and smooth optimization problems, where T is the number of rounds of local SGD updates. These works either simplify the heterogeneity of the devices, e.g., ignoring cases where some devices may partially finish some aggregation rounds or quit forever during the training (Li et al., 2020b), or consider alternative objective functions for the SGD

algorithm to optimize (Li et al., 2018). Alternatively, some recent papers propose to combine federated learning with the multi-task learning paradigm (Smith et al., 2017)(Corinzia and Buhmann, 2019) where multiple models are trained simultaneously, but they also entail dedicated device participation throughout the training.

The *FedAvg* algorithm with non-IID data across devices has also been modified in specific edge computing scenarios to reduce the communication overhead (Liu et al.)(Sattler et al., 2019)(Bonawitz et al., 2019) or maintain a good training convergence under a resource budget constraint (Wang et al., 2019). However, these works do not consider the possibility that the edge devices can be unavailable during the training process or join at different times, which are the main challenges of this work. An online learning framework (Chen et al., 2019)(Han et al., 2020)(Damaskinos et al., 2020) is a possible way to enable flexible device participation in the federated learning scenario. For instance, Chen et al. (2019) propose an asynchronous federated learning algorithm to handle unbalanced data that arrives in an online fashion onto different devices. Although the asynchronous aggregation in their proposed algorithm can be naturally applied to randomly inactive devices, the authors do not analyze how their algorithm's convergence is affected by the device inactivity or incompleteness and the data heterogeneity.

In recent years, some attempts have been made to relax the strict training requirements on the participating devices. For example, Tu et al. (2020) study federated learning in a fog network topology with possible data sharing among devices; Yang et al. (2020) incorporate heterogeneity of devices into the design of the learning systems; Nishio and Yonetani (2019) propose a client selection policy that adapts to the change of devices' hardware status. However, these works do not show how the variations in the devices could affect the convergence of training, nor do they incorporate the heterogeneity of user data into the algorithm design.

In (Rizk et al., 2020) and (Wang et al., 2020), the authors reveal that incomplete devices can block the convergence, but they consider neither other dynamic participation patterns such as inactivity, arrivals and departures, nor probabilistic models for uncertain device participation. To relieve the impact of incomplete devices, these works propose similar strategies as our paper by reweighting the contribution of local models. However, they focus mostly on removing the additional bias term originating from heterogeneous device updates, without looking into how this bias is related to the participation frequency of devices and the divergence among them. They also do not compare the proposed methods with alternative extensions of *FedAvg*. In this work, we model the device participation as random variables and incorporate them into the convergence analysis, and we compare the convergence rates for three reasonable aggregation schemes.

3 Convergence Analysis

In this section, we establish a convergence bound for federated learning with flexible device participation patterns. Our analysis generalizes the standard *FedAvg* to incorporate arbitrary aggregation coefficients. In the aggregation step, all devices are counted even if they cannot finish all local epochs. The analysis considers a non-IID data distribution and heterogeneous devices, i.e., some devices can be more stable than the others. We first derive the convergence bound with incomplete and inactive devices in Sections 3.1 to 3.2, and then discuss arrivals and departures in Section 3.3.

3.1 Algorithm Description

Suppose there are N devices, where each device k defines a local objective function $F_k(w)$. Here w represents the parameters of the machine learning model to be optimized, and $F_k(w)$ may be defined as the average empirical loss over all data points at device k, as in typical federated learning frameworks (McMahan et al., 2016). The global objective is to minimize $F(w) = \sum_{k=1}^{N} p^k F_k(w)$, where $p^k = \frac{n_k}{n}$, n_k is the number of data points device k owns, and $n = \sum_{k=1}^{N} n_k$. Let w^* be the minimizer of F, and denote by F_k^* the minimum value of F_k. We quantify the degree to which data at each device k is distributed differently than that at other devices as $\Gamma_k = F_k(w^*) - F_k^*$ to capture that data distributions at different devices are non-IID, and let $\Gamma = \sum_{k=1}^{N} p^k \Gamma_k$ as in (Li et al., 2020b).

We consider discrete time steps $t = 0, 1, \ldots$. Model weights are synchronized when t is a multiple of E, i.e., each round consists of E time steps. Assume there are at most T rounds. For each round (say the τth round), the following three steps are executed:

- Synchronization: the coordinator broadcasts the latest global weight $w_{\tau E}^{\mathcal{G}}$ to all devices. Each device updates its local weight so that: $w_{\tau E}^k = w_{\tau E}^{\mathcal{G}}$
- Local updates: each device runs stochastic gradient descent (SGD) on F_k for $i = 0, \ldots, s_\tau^k - 1$:[1]

$$w_{\tau E+i+1}^k = w_{\tau E+i}^k - \eta_\tau g_{\tau E+i}^k \tag{1}$$

  Here η_τ is a staircase learning rate that decays with τ, $0 \le s_\tau^k \le E$ represents the number of local updates this device completes in this round, $g_t^k = \nabla F_k(w_t^k, \xi_t^k)$ is the stochastic gradient at device

[1] While some papers define local epochs and local updates separately, we use them interchangeably in this paper. Both refer to the times (1) is conducted in a global round.

k, and ξ_t^k is a mini-batch sampled from device k's local dataset. We also define $\bar{g}_t^k = \nabla F_k(w_t^k)$ as the full batch gradient at device k, hence $\bar{g}_t^k = \mathbb{E}_{\xi_t^k}[g_t^k]$.

- Aggregation: the coordinator aggregates the gradients and generates the next global weight as

$$\begin{aligned} w^{\mathcal{G}}_{(\tau+1)E} &= w^{\mathcal{G}}_{\tau E} + \sum_{k=1}^{N} p_\tau^k (w_{\tau E + s_\tau^k} - w^{\mathcal{G}}_{\tau E}) \\ &= w^{\mathcal{G}}_{\tau E} - \sum_{k=1}^{N} p_\tau^k \sum_{i=0}^{s_\tau^k} \eta_\tau g^k_{\tau E+i} \end{aligned} \tag{2}$$

We define that a device k is *inactive* in round τ if $s_\tau^k = 0$ (i.e., it completes no local updates), and say it is *incomplete* if $0 < s_\tau^k < E$. We treat each s_τ^k as a random variable that can follow an arbitrary distribution. Devices are *heterogeneous* if they have different distributions of s_τ^k, and otherwise they are *homogeneous*. We allow the aggregation coefficients p_τ^k to vary with τ. In Section 4, we will discuss different schemes of choosing p_τ^k and their impacts on the convergence.

As a special case, traditional *FedAvg* assumes all selected devices can complete all E local epochs, so that $s_\tau^k \equiv E$. Also, *FedAvg* with full device participation uses fixed aggregation coefficients $p_\tau^k \equiv p^k$, so that the right hand side of (2) can be written as $\sum_{k=1}^N p^k w^k_{\tau E}$, i.e., aggregating gradients is equivalent to aggregating the model parameters directly.

3.2 General Convergence Bound

The analysis relies on the following five assumptions. The first four are standard (Li et al., 2020b). The last assumption ensures bounded aggregation coefficients and is satisfied by all schemes discussed in Section 4. In Section 5, we experimentally show that our proposed learning algorithm performs well even when some assumptions (like strong convexity) are violated.

Assumption 3.1. *$F_1, \dots, F_N$ are all L-smooth, so that F is also L-smooth.*

Assumption 3.2. *$F_1, \dots, F_N$ are all μ-strongly convex, so that F is also μ-strongly convex.*

Assumption 3.3. *The variance of the stochastic gradients is bounded: $\mathbb{E}_\xi \|g_t^k - \bar{g}_t^k\|^2 \le \sigma_k^2$, $\forall k, t$.*

Assumption 3.4. *The expected squared norm of the stochastic gradients at each local device is uniformly bounded: $\mathbb{E}_\xi \|g_t^k\|^2 \le G^2$ for all k and t.*

Assumption 3.5. *There exists an upper bound $\theta > 0$ for the aggregation coefficient: $p_\tau^k / p^k \le \theta, \forall k$.*

Assume the following expectations exist and do not vary with time: $\mathbb{E}[p_\tau^k]$, $\mathbb{E}[p_\tau^k s_\tau^k]$, $\mathbb{E}[(p_\tau^k)^2 s_\tau^k]$, $\mathbb{E}[(\sum_{k=1}^N p_\tau^k - 2)_+ (\sum_{k=1}^N p_\tau^k s_\tau^k)]$ for all rounds τ and devices k, and assume $\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k] \neq 0$. Intuitively, this last assumption ensures that some updates are aggregated in each round, otherwise this round can be simply omitted.

Generally, p_τ^k's are functions of s_τ^k, and these expectations can be estimated from device histories. Let $z_\tau \in \{0, 1\}$ indicate the event that the ratio $\mathbb{E}[p_\tau^k s_\tau^k]/p^k$ does not take the same value for all k. We can obtain the following convergence bound for general p_τ^k:

Theorem 3.1. *By choosing the learning rate $\eta_\tau = \frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]} \frac{1}{\tau E + \gamma}$, we can obtain*

$$\mathbb{E}\|w^{\mathcal{G}}_{\tau E} - w^*\|^2 \le \frac{M_\tau D + V}{\tau E + \gamma} \tag{3}$$

Here we define $\gamma = \max\left\{ \frac{32E(1+\theta)L}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}, \frac{4E^2\theta}{\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]} \right\}$, $M_\tau = \sum_{t=0}^{\tau-1} \mathbb{E}[z_t]$, $D = \frac{64E \sum_{k=1}^N \mathbb{E}[p_\tau^k s_\tau^k] \Gamma_k}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}$, $V = \max\left\{ \gamma^2 \mathbb{E}\|w_0^{\mathcal{G}} - w^*\|^2, \left(\frac{16E}{\mu \mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]}\right)^2 \frac{\mathbb{E}[B_\tau]}{E} \right\}$, $B_\tau = 2(2+\theta)L \sum_{k=1}^N p_\tau^k s_\tau^k \Gamma_k + \left(2 + \frac{\mu}{2(1+\theta)L}\right) E(E-1) G^2 \left(\sum_{k=1}^N p_\tau^k s_\tau^k + \theta (\sum_{k=1}^N p_\tau^k - 2)_+ \sum_{k=1}^N p_\tau^k s_\tau^k \right) + 2EG^2 \sum_{k=1}^N \frac{(p_\tau^k)^2}{p^k} s_\tau^k + \sum_{k=1}^N (p_\tau^k)^2 s_\tau^k \sigma_k^2$

Theorem 3.1 shows that the convergence rate is affected by the aggregation coefficients p_τ^k's as they determine M_τ, D, and V. From (3), $w^{\mathcal{G}}_{\tau E}$ will eventually converge to a globally optimal solution only if M_τ increases sub-linearly with τ. In the original full-participation *FedAvg*, $p_\tau^k s_\tau^k \equiv p^k E$, so $z_\tau \equiv 0$ and $M_\tau \equiv 0$ as per the definitions. Thus, full-participation *FedAvg* converges according to (3), which is consistent with (Li et al., 2020b). However, when considering flexible device participation, M_τ may increase with τ, which can cause *FedAvg* to converge to an arbitrary suboptimal point. The magnitude of M_τ is determined by the degree of heterogeneity in the device participation, and D is bounded by the non-IID metric Γ_k of local datasets. If M_τ increases linearly with τ (e.g., due to device departures), the model will converge to a suboptimal point with the loss bounded by $\frac{D}{E}$. As we will see in Section 4.1, by smartly choosing the aggregation coefficients p_τ^k, the increase of M_τ can be controlled and a convergence to the global optimum can still be established.

While we only show results for s_τ^k whose distributions are static with time, Theorem 3.1 can be easily extended to time-varying distributed s_τ^k by replacing the corresponding expectations of $p_\tau^k s_\tau^k$ and $(p_\tau^k)^2 s_\tau^k$ with their minimum or maximum expectations over τ.

3.3 Shifts in the Global Objective

Recall the global objective is $F(w) = \sum_{k \in \mathcal{C}} p^k F_k(w)$, i.e., an average of local objectives for participating devices $\mathcal{C}$. A well trained model w^* is expected to perform well on all data points generated by devices in $\mathcal{C}$. In the presence of departing and arriving devices,

$\mathcal{C}$ may shrink or expand dynamically during the training. The global objective thus varies accordingly. For example, after admitting an incoming device l with n_l data points: $\tilde{\mathcal{C}} \leftarrow \mathcal{C} + \{l\}$, the global objective becomes $\tilde{F}(w) = \tilde{p}^l F_l(w) + \sum_{k\in\mathcal{C}} \tilde{p}^k F_k(w)$, where $\tilde{p}^k = \frac{n_k}{n_\mathcal{C}+n_l}$. The model $\tilde{w}^*$ fully trained with this objective is then applicable to the new data from device l. We formally define *objective shift* as the process of changing the global objective, and the applicability of the trained model, by adding or removing devices from $\mathcal{C}$.

The following theorem bounds the offset between the global optima due to the objective shift. As we can intuitively expect, the difference reduces when the data becomes more IID ($\Gamma_l \to 0$), and when the departing/arriving device owns fewer data points ($n_l \to 0$):

Theorem 3.2. *Suppose a device l arrives/departs, and let n be the total number of data points originally. Consider the objective shift $F \to \tilde{F}$, $w^* \to \tilde{w}^*$. Let $\tilde{\Gamma}_k = F_k(\tilde{w}^*) - F_k^*$ quantify the degree of non-IID with respect to the new objective. Then in the arrival case*

$$\|w^* - \tilde{w}^*\| \le \frac{2\sqrt{2L}}{\mu} \frac{n_l}{n+n_l} \sqrt{\Gamma_l} \tag{4}$$

and in the departure case

$$\|w^* - \tilde{w}^*\| \le \frac{2\sqrt{2L}}{\mu} \frac{n_l}{n} \sqrt{\tilde{\Gamma}_l} \tag{5}$$

Objective shift is mandatory when a new device (say device l) arrives: Unless $F_l \equiv F$ (which is highly unlikely), incorporating updates from l will always move $F(w)$ away from F^*. The best strategy without objective shift is then not to aggregate updates from l, and thus not to admit l into the learning process in the first place. In contrast, objective shift is optional when devices depart: we can keep the original objective F even if we will no longer receive updates from a departing device, if doing so yields smaller training loss.

Suppose an objective shift occurs at τ_0. The remainder of the training is then equivalent to starting over from $w^{\mathcal{G}}_{\tau_0 E}$ but converging towards the new objective $\tilde{w}^*$. Combining Theorems 3.1 and 3.2, we can obtain the following convergence bound after the objective shifts:

Corollary 3.2.1. *Assume the objective shifts at τ_0 with $\mathbb{E}\|w^{\mathcal{G}}_{\tau_0 E} - w^*\|^2 \le \Delta_{\tau_0}$. By increasing the learning rate back to $\eta_\tau = \frac{16E}{\mu\mathbb{E}[\sum_{k=1}^N p_\tau^k s_\tau^k]} \frac{1}{(\tau-\tau_0)E+\gamma}$ for $\tau > \tau_0$, the convergence to the new objective can be bounded by*

$$\mathbb{E}\|w^{\mathcal{G}}_{\tau E} - \tilde{w}^*\|^2 \le \frac{\tilde{M}_\tau \tilde{D} + \tilde{V}}{(\tau-\tau_0)E + \tilde{\gamma}} \tag{6}$$

Here $\tilde{M}_\tau, \tilde{D}, \tilde{V}, \tilde{\gamma}$ are defined analogously to M_τ, D, V, γ but they respectively include/exclude the arriving/departing device. The first term in $\tilde{V}$ equals $\tilde{\gamma}^2(\sqrt{\Delta_{\tau_0}} + \|w^* - \tilde{w}^*\|)^2 = O\left(\frac{V}{\tau_0 E+\gamma} + \Gamma_l\right)$.

The increase of the learning rate after the objective shift is necessary. Intuitively, if the shift happens at a large time τ_0 when $w^{\mathcal{G}}_{\tau_0 E}$ is close to the old optimal w^* and η_{τ_0} is close to zero, the learning rate used in Theorem 3.1 will be too small to steer the model to the new optimum, since $\|w^{\mathcal{G}}_{\tau_0 E} - \tilde{w}^*\| \approx \|w^* - \tilde{w}^*\|$.

Comparing (3) and (6), an objective shift yields an one-time increase in the loss, which forces us to take actions when confronting departures and arrivals. In the case of device departure, it is possible that retaining the old objective can result in a smaller training loss compared to doing a shift. In this situation, the trained model is still applicable to data of the departing device. In the arrival case, though objective shift is mandatory, we can still accelerate the training by a "fast-reboot", applying extra gradient updates from the arriving device.

We will discuss in Section 4.2 the fast-reboot method for the arrival case, and in Section 4.3 the decision of model applicability for the departure case.

4 Main Results

Based on the convergence analysis in Section 3, in this section, we present corollaries that can guide operators in reacting to flexible device participation.

4.1 Debiasing on Incomplete Aggregation

According to Theorem 3.1, the convergence bound is controlled by the expectation of p_τ^k and its functions. Below we discuss three plausible schemes of choosing p_τ^k, and compare their convergence rates in Table 1.

- Scheme A: Only aggregate parameters from devices that complete all E local epochs, with aggregation coefficient $p_\tau^k = \frac{Np^k}{K_\tau} q_\tau^k$, where K_τ is the number of complete devices, $q_\tau^k \in \{0,1\}$ denotes if client k is complete. If $K_\tau = 0$, this round is discarded.
- Scheme B: Allow clients to upload incomplete work (with $s_\tau^k < E$ updates), with fixed aggregation coefficient $p_\tau^k = p^k$.
- Scheme C: Accept incomplete works as in Scheme B, with adaptive $p_\tau^k = \frac{E}{s_\tau^k} p^k$, or $p_\tau^k = 0$ if $s_\tau^k = 0$.

Schemes A and B are natural extensions of *FedAvg*. Scheme C assigns a greater aggregation coefficient to devices that complete fewer local epochs. Though this idea seems counter-intuitive, as fewer local updates might lead to less optimal parameters (cf. Table 1), it turns out to be the only scheme that guarantees convergence when device participation is heterogeneous.

Corollary 4.0.1. *Let K_τ be the number of devices that run all E epochs, I_τ indicate the appearance of any inactive devices in round τ, and write $\bar{\sigma}_N^2 \equiv \sum_k^N (p^k \sigma_k)^2$.*

Table 1 gives the convergence rates of Schemes A, B, C when device updates may be incomplete and inactive.

Table 1: Convergence rates with incomplete and inactive devices. The bound for Scheme A assumes there is at least one complete device ($K_\tau \neq 0$), and those for Schemes B, C assume s_τ^k is not trivially zero ($\mathbb{E}[s_\tau^k] \neq 0$). While the three schemes have similar performance in the homogeneous setting, Schemes A and B fail to converge to the global optimum even assuming all devices are active. Scheme C works if inactive devices do not occur in every round ($\sum_t I_t < O(\tau)$).

	Homogeneous	Heterogeneous
A	$O\left(\frac{\mathbb{E}[\frac{N^2}{K_\tau}]+\bar{\sigma}_N^2+\Gamma}{\tau}\right)$	$\leq \frac{D}{E}$
B	$O\left(\frac{\bar{\sigma}_N^2+\Gamma}{\tau\mathbb{E}[s_\tau]}\right)$	$\leq \frac{D}{E}$
C	$O\left(\frac{\bar{\sigma}_N^2+\Gamma}{\tau(\mathbb{E}[1/s_\tau])^{-1}}\right)$	$O\left(\frac{\sum_{t=0}^{\tau-1} I_t D+\sum_k^N (p^k\sigma_k)^2\mathbb{E}\left[\frac{1}{s_\tau^k}\right]+\Gamma}{\tau}\right)$

The reason for enlarging the aggregation coefficients in Scheme C can be understood by observing from (2) that increasing p_τ^k is equivalent to increasing the learning rate of device k. Thus, by assigning devices that run fewer epochs a greater aggregation coefficient, these devices effectively run further in each local step, compensating for the additional epochs other devices completed. As shown in Figure 1, Scheme C ensures an unbiased gradient after aggregation, while Schemes A and B will favor devices that run more epochs. Ideally, allowing devices to adapt learning rates by themselves would effectively lead to the same result. However, when a device is running local updates, it may not yet know or be able to estimate the number of local epochs it will complete. In contrast, centralized intervention can make accurate adjustments a posterior.

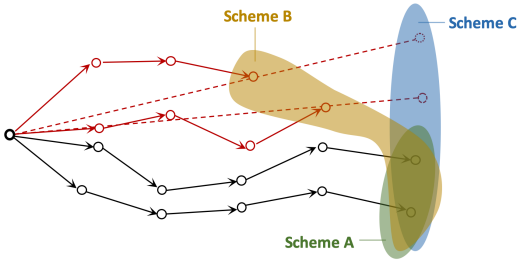

Figure 1: Snapshot of one aggregation round. The bottom two devices completed all $E = 5$ local epochs, while the top two completed only 3 and 4 epochs. Scheme C enlarges the incomplete gradients by respectively 5/3 and 5/4 and produces unbiased aggregation results. Aggregations with Schemes A and B are biased towards devices that run more epochs.

Table 1 also reveals how the following system and statistical factors affect the convergence asymptotically:

- The non-IID metric Γ is the major obstacle of convergence in the homogeneous case. In the heterogeneous setting, the D term (which grows with Γ) dominates the training loss. It controls the maximum non-diminishing loss D/E of Scheme A and B, and decelerates the training of Scheme C in the presence of inactive devices.
- Devices' activeness s_τ^k and K_τ contribute inversely to the training loss: The more devices participate, the faster the loss decays. When inactivity occurs frequently, Scheme C cannot converge either. E.g., if a device never responds to the coordinator (so $I_t \equiv 1$), its training loss can never converge to zero.
- The variance $\bar{\sigma}_N, \sigma_k$ in the stochastic gradient descent algorithm slows down the training as expected.

4.2 Fast-rebooting on Arrivals

Intuitively, when a device l arrives, $\tilde{w}^*$ will be "dragged" towards its local optimum w_l^*. The gradients from device l may thus encode more information about the new optimum $\tilde{w}^*$ compared to those from the other devices. Thus, by adding an extra update $-\delta^l \nabla F_l(w^{\mathcal{G}}), \delta^l > 0$ to the gradient aggregation, it is likely that w can move closer to $\tilde{w}^*$, allowing the training to fast-reboot from the point of arrivals. However, as shown in Figure 2, this intuition may not hold: it is also possible that $-\delta^l \nabla F_l(w^{\mathcal{G}})$ ends up driving $w^{\mathcal{G}}$ away from $\tilde{w}^*$. In fact, the success of this method is determined by the distance $b = \|w^{\mathcal{G}} - w^*\|$. When b is small, it is highly likely the extra update can rapidly reboot the training. We formalize this statement in Corollary 4.0.2.

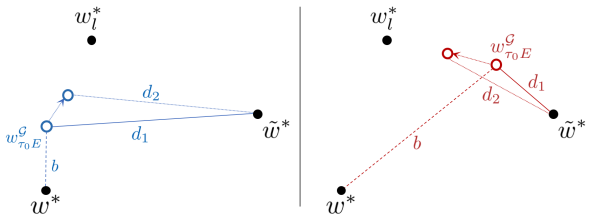

Figure 2: Left: when the distance to the old optimum $b = \|w^{\mathcal{G}} - w^*\|$ is small, applying an extra update to $w^{\mathcal{G}}$ following the direction $-\nabla F_l(w^{\mathcal{G}})$ moves it closer to $\tilde{w}^*$ ($d_2 < d_1$). Right: for a large b, the extra update may on the contrary enlarge this distance ($d_2 > d_1$).

Corollary 4.0.2. *Assume $\nabla F(w)$ is continuous, and $0 < \|\nabla F(w)\|_2, \|\nabla^2 F(w)\|_2 \leq W$ for any w (The latter is the induced l_2 norm for matrices). Let $w' = w - \delta^l \nabla F_l(w)$, then there exists a $\delta^l > 0$ such that $\|w' - \tilde{w}^*\| < \|w - \tilde{w}^*\|$ if w satisfies*

$$\|w - w^*\| < \frac{\tilde{F}(w^*) - \tilde{F}(\tilde{w}^*)}{\left(\frac{2\sqrt{2L}}{\mu}\tilde{p}^l\sqrt{\Gamma_l} + 1\right)\tilde{p}^l W} \tag{7}$$

(7) defines a sphere around the original global optimum w^* within which the extra update helps fast-reboot. The radius of the sphere depends on the divergence between the new (arriving) and old data points. Generally, the longer the training has elapsed, the closer the global model is to w^*. Thus, the extra updating works best for devices that arrive late in the training.

When applied in practice, the extra updating can be conducted on-the-fly, by augmenting the aggregation coefficient of the arriving device so that $p_\tau^l = p^l + \delta^l$. Furthermore, the distance b can be estimated by the gradient norm with respect to the original objective.

As the name suggests, fast-reboot only accelerates the training for a certain duration after the device arrives. In fact, if there are no future interrupts, models with or without fast-rebooting eventually converge to the same global optimum. Nevertheless, fast-reboot is still beneficial if there is insufficient training time remaining (e.g., a device arrives near the end of the training).

4.3 Redefining Applicability on Departures

As is discussed in Section 3.3, when a device leaves, we need to redefine the applicability of the trained model. Namely, one can decide to either exclude this departing device and shift the objective, or keep including it and stick to the old objective. The decision depends on the time at which the device leaves. When including the device as a part of the global objective, from (3), since $M_\tau = \tau - \tau_0$ from then on, the training loss will always exceed a structural bias D/E. In contrast, if the device is excluded and the model is trained with a shifted global objective, there will be an immediate increase in the convergence bound as in Theorem 3.2. But afterwards, the bound will decrease and eventually the parameters will converge to the new global optimum.

Assume a device leaves at $\tau_0 < T$ and there are no subsequent arrivals/departures. Let $f_0(\tau)$ be the convergence bound if we include the device, and $f_1(\tau)$ be the bound if it is excluded. We can obtain $f_0(\tau) = \frac{(\tau-\tau_0)D+V}{\tau E+\gamma}, f_1(\tau) = \frac{\tilde{V}}{(\tau-\tau_0)E+\tilde{\gamma}}$. Here $\tilde{M}_\tau, \tilde{V}_\tau, \tilde{\gamma}$ are defined analogously to M_τ, V_τ, γ but they exclude the departing device. A device is excluded if by doing so, a smaller training loss can be obtained at the deadline T, which is summarized in the following corollary:

Corollary 4.0.3. *Excluding a device that departs at τ_0 leads to smaller training loss if*

$$\min_{\tau \geq \tau_0} f_0(\tau) \geq f_1(T) \tag{8}$$

Further assume $\tilde{\gamma} = \gamma$, and $\tilde{V}$ is dominated by its first term so that $\tilde{V} = \frac{V}{\tau_0 E+\gamma} + \Gamma_l$. (8) then becomes

$$T - \tau_0 \geq O\left(\sqrt{\Gamma_l \tau_0}\right) \tag{9}$$

From (9), when the remaining training time $T - \tau_0$ is at least $O(\sqrt{\Gamma_l \tau_0})$, applying the trained model to the departing device becomes less promising. It is thus better to exclude it and shift the objective. As we can expect, the bound grows with Γ_l, since the non-IID contribution from the departing device increases the initial $\tilde{V}$. As τ_0 increases, the learning rate without shift gets smaller, mitigating the increase of the training loss from departing devices.

5 Experiments

In this section, we experimentally evaluate Section 4's results. Due to the limitations on hardware resources, the training process is performed in computer simulations. To ensure the simulation is consistent with the real learning environment, we use real-world traces to represent the participation patterns of simulated devices. We present our experiment setup in Section 5.1, and verify our theory results in Sections 5.2 - 5.4.

5.1 Experiment Setup

We create various data traces to represent the heterogeneous participation patterns of local devices. We set up a simple federated learning experiment with five Raspberry PIs as workers, and a desktop server as the coordinator. Each PI has a training process that runs the original *FedAvg* algorithm, and a competitor process doing CPU-intensive work simultaneously. We manually tune the workload of the competitor process so that it takes up 0%, 30%, 50%, 70%, 90% of the PI's CPU resources, simulating different device configurations in federated learning. Under the five settings, for each round, we record the percentage of required epochs the PI ends up submitting before a preset, fixed deadline. Due to the default load-balancing behavior of the operating system's CPU scheduler, these traces do not contain zero epochs (i.e. inactive cases). To generate inactive device participation patterns, we create another set of three traces with respectively low, medium and high bandwidth. Devices can thus be inactive due to weak transmission. Table 2 shows the mean and standard deviation of the percentage of epochs completed for each trace. In the following experiments, each simulated device is randomly assigned a trace. For each aggregation round τ, it randomly samples from its trace to obtain the number of local epochs s_τ^k.

Three datasets are used in this paper: MNIST (LeCun et al., 1998), EMNIST (Cohen et al., 2017) and SYNTHETIC(α, β) (Li et al., 2018). We build a two-layer MLP model and a two-convolution-layer CNN model respectively for MNIST and EMNIST, both models are defined by McMahan et al. (2016). For SYNTHETIC(α, β), we use an ordinary logistic re-

Table 2: The means and standard deviations for the percentage of required local epochs actually submitted to the coordinator during the federated training. The first five traces do not contain inactive cases.

Name	$\mathcal{T}_0$	$\mathcal{T}_{30}$	$\mathcal{T}_{50}$	$\mathcal{T}_{70}$	$\mathcal{T}_{90}$	$\mathcal{T}_{hi}$	$\mathcal{T}_{mi}$	$\mathcal{T}_{lo}$
Mean	100	75.3	67.2	57.2	56.3	82.5	74.1	51.2
Stdev	0	14.8	11.3	11.7	14.8	23.3	22.3	18.3

gression model. All models use the vanilla SGD as local optimizers, with batch sizes of 10 for MNIST and EMNIST, and 20 for SYNTHETIC. When generating non-IID data, we sort the MNIST and EMNIST data by labels so that each device is assigned data from one label chosen uniformly at random. For SYNTHETIC(α, β), we vary the parameters α, β from 0 to 1. The larger α, β are, the less IID the dataset becomes. We use the staircase learning rate $\eta_\tau = \eta_0/\tau$ as adopted in our convergence analysis. The initial η_0 is 2e-3 for MNIST, 5e-4 for EMNIST, and 1 for SYNTHETIC(α, β). Unless otherwise noted, the number of samples at each device follows the Type-I Pareto distribution with the Pareto index of 0.5.

5.2 Comparison of Aggregation Schemes

We first examine the effects of the device heterogeneity and the non-IID data distributions on the convergence for each aggregation scheme. We conduct eight sets of experiments where we incrementally increase the number of participation traces to reflect the increasing heterogeneity in device participation. For SYNTHETIC, we use $\alpha = \beta = 0$ for the IID case, and $\alpha = \beta = 1$ for the non-IID case. We train on 100 devices for MNIST, 62 devices for EMNIST (by merge), and 50 devices for SYNTHETIC(α, β). Table 3 records the differences in the test accuracies between different aggregation schemes after 200 global epochs. The typical convergence process is depicted in Figure 3.

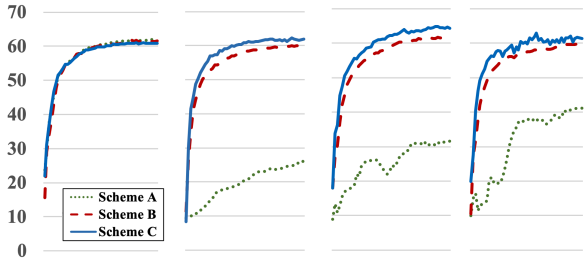

Figure 3: Test accuracy for non-IID EMNIST. Plots from left to right correspond to $|\mathcal{T}| = 1, 3, 5, 8$. (increasing device heterogeneity)

As we can see, Scheme C yields the best test accuracy on average. Compared to Schemes A and B, it achieves higher accuracy when devices get more heterogeneous and less IID. This is consistent with our loss bounds in Table 1, since Schemes A and B fail to converge to the global optimum in the heterogeneous case with non-IID data. On the other hand, Scheme A performs extremely badly with large $|\mathcal{T}|$. This is because the last few traces contain very few complete rounds, significantly increasing $\mathbb{E}[1/K_\tau]$. Noteworthily, Scheme C is no different from, or even worse than Scheme B in more homogeneous settings, this is consistent with Table 1 since $\frac{1}{\mathbb{E}[s_\tau]} \leq \mathbb{E}[\frac{1}{s_\tau}]$. When the traces contain inactive devices ($|\mathcal{T}| \geq 6$), Scheme C becomes less stable due to the variance introduced by I_t in Corollary 4.0.1.

Table 3: The % improvement in the test accuracies of Scheme B w.r.t. Schemes A(left numbers) and Scheme C w.r.t. Scheme B (right numbers). $|\mathcal{T}| = j$ represents using the first j traces in Table 2.

(a) MNIST Data

$\lvert\mathcal{T}\rvert$	1	2	3	4
IID	-0.6 | 0.3	1.9 | 0.1	5.6 | 0.1	8.3 | 0.7
NIID	0.2 | -0.3	9.5 | 1.8	19.3 | 1.6	33.8 | 3.3
$\lvert\mathcal{T}\rvert$	5	6	7	8
IID	10.4 | 2.6	14.0 | 2.2	5.8 | 1.9	11.8 | 2.4
NIID	33.2 | 3.2	28.7 | 6.2	36.0 | 3.6	43.4 | 6.9

(b) EMNIST Data

$\lvert\mathcal{T}\rvert$	1	2	3	4
IID	0.7 | -0.6	0.9 | 0.1	4.2 | 0.7	4.8 | 1.0
NIID	-0.1 | -0.7	17.0 | -2.0	34.2 | 1.8	37.9 | 4.8
$\lvert\mathcal{T}\rvert$	5	6	7	8
IID	6.9 | 1.1	6.6 | 1.2	4.0 | 1.5	7.6 | 1.2
NIID	30.2 | 2.5	22.5 | 3.0	25.3 | 2.2	18.6 | 1.8

(c) SYNTHETIC Data

$\lvert\mathcal{T}\rvert$	1	2	3	4
IID	-0.6 | 0.5	2.1 | 0.1	6.6 | 0.0	9.0 | 0.7
NIID	0.1 | -0.4	9.6 | 1.5	22.2 | 1.8	38.2 | 3.2
$\lvert\mathcal{T}\rvert$	5	6	7	8
IID	11.6 | 3.0	16.4 | 2.5	6.3 | 1.9	14.4 | 2.8
NIID	33.3 | 3.9	30.5 | 7.9	37.9 | 4.5	41.6 | 8.0

5.3 Effectiveness of Fast-Reboot

We now investigate the effectiveness of the fast-reboot method described in Section 4.2. The experiments involve $N-1$ existing devices, and the arriving device joins at τ_0. As is discussed in Section 4.2, the method makes no difference when data distribution is IID. We thus only consider non-IID cases. We set $N = 10$ for MNIST and EMNIST (balanced) and $N = 30$ for

SYNTHETIC(1, 1). To avoid the interference brought by inactive devices, for this experiment we only use the first five traces in Table 2, and we adopt Scheme C as the aggregation method. All devices are given the same number of samples for fair comparison.

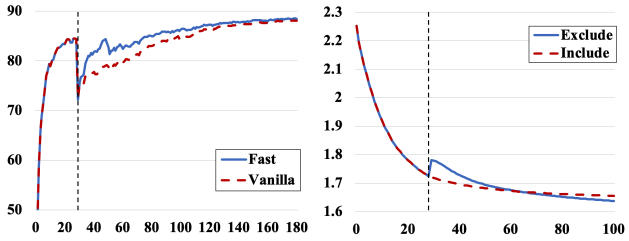

Figure 4: Evolution of the test accuracy (left) and loss (right) under device arrival (left, MNIST) and departure (right, SYNTHETIC) cases. The dashed vertical lines indicate the arriving (departing) time τ_0. After τ_0, except for the "include" option, models are tested with new datasets that include (exclude) holdout data from the arriving (departing) device.

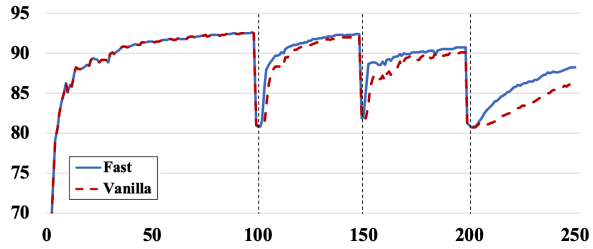

Figure 5: Test accuracy with and without fast-reboot for multiple arrivals for non-IID MNIST. The test dataset is updated every time a new device arrives to include its holdout data. The vertical dashed lines indicate the time the device arrives.

Table 4: The number of global epochs after the arriving time τ_0 until the test accuracy bounces back to that at $\tau_0 - 1$. Left: fast reboot. Right: vanilla reboot.

τ_0	10		30		50		70	
MNIST	4	4	22	27	55	63	59	66
EMNIST	4	3	11	12	14	19	21	24
SYNTHETIC	1	1	4	6	7	12	3	8

When the device arrives, we increase the learning rate to $\eta_0/(\tau - \tau_0)$. The aggregation coefficient of the arriving device l is boosted to $p^l_\tau = 3p^l$ initially, and decays to p^l by $O(\tau^{-2})$. Table 4 records the number of global epochs it takes to recover to the accuracy level before the arrival. Fast-reboot consistently achieves faster rebound, and works better for late arrivals as we expect. EMNIST-CNN enjoys less improvement from fast-reboot because CNN models converge more slowly than MLP and logistic regression models. Thus, at the moment new devices arrive, EMNIST models have not fully converged to the old optima, degrading the effectiveness of fast-reboot as per Corollary 4.0.2. The typical fast-reboot process is shown in Figure 4.

Next we study the situation when multiple devices arrive in a row. Figure 5 shows the training process for MNIST data. Every time a device arrives, we increase the learning rate as per Corollary 3.2.1. Initially, seven devices are in the training. After 100 global epochs, the remaining three devices arrive at 50 epoch intervals, without waiting for the model to fully converge. From Figure 5, the fast-reboot trick accelerates the convergence for every device arrival.

5.4 Model Applicability upon Departures

The right plot in Figure 4 shows the typical change of the test loss after the device departs. We use the same setting as in Section 5.3. As is predicted in Section 4.3, an objective shift ('exclude') initially increases the test loss. But eventually, the two curves cross and excluding the device becomes more beneficial.

Table 5 summarizes the number of global epochs it takes for the curves to cross with SYNTHETIC(α, β). As we can see, the values increase with τ_0 and the non-IID metric (α, β), confirming Corollary 4.0.3.

Table 5: The number of global epochs after the departing time τ_0 until the test losses coincide for including and excluding options. The rows correspond to three choices of parameters (α, β) in SYNTHETIC(α, β).

τ_0	10	15	20	25	30	35	40	45	50
(.1, .1)	2	5	3	3	9	3	10	26	40
(.5, .5)	1	3	9	14	13	7	12	36	34
(1., 1.)	10	9	27	18	34	17	28	62	77

6 Conclusion and Future Work

This paper extends the federated learning paradigm to incorporate more flexible device participation. The analysis shows that incomplete local device updates can be utilized by scaling the corresponding aggregation coefficients, and a mild degree of device inactivity will not impact the convergence. Further investigation reveals how the convergence relates to heterogeneity in both the data and the device participation. The paper also proposes techniques to fast-reboot the training after new devices arrive, and provides an analytical criterion on when to exclude a departing device. In the future work, we will analyze groups of arrivals or departures, and investigate the possibility for users to dynamically update their datasets during the training.

Acknowledgements

This research was partially supported by the CMU CyLab IoT Initiative, and NSF CNS-1909306, CNS-1751075.

References

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019.

Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

Yujing Chen, Yue Ning, and Huzefa Rangwala. Asynchronous online federated learning for edge devices. *arXiv preprint arXiv:1911.02134*, 2019.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

Luca Corinzia and Joachim M Buhmann. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

Georgios Damaskinos, Rachid Guerraoui, Anne-Marie Kermarrec, Vlad Nitu, Rhicheek Patra, and Francois Taiani. Fleet: Online federated learning via staleness awareness and performance prediction. *arXiv preprint arXiv:2006.07273*, 2020.

Pengchao Han, Shiqiang Wang, and Kin K Leung. Adaptive gradient sparsification for efficient federated learning: An online learning approach. *arXiv preprint arXiv:2001.04756*, 2020.

Kevin Hsieh, Amar Phanishayee, Onur Mutlu, and Phillip B Gibbons. The non-iid data quagmire of decentralized machine learning. *arXiv preprint arXiv:1910.00189*, 2019.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, page 106854, 2020a.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *In Proc. of ICLR*, 2020b.

Lumin Liu, Jun Zhang, S. H. Song, and Khaled B. Letaief. Client-edge-cloud hierarchical federated learning. `https://arxiv.org/abs/1905.06641`.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

Elsa Rizk, Stefan Vlaski, and Ali H Sayed. Dynamic federated learning. *arXiv preprint arXiv:2002.08782*, 2020.

Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 2019.

Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4424–4434, 2017.

Yuwei Tu, Yichen Ruan, Su Wang, Satyavrat Wagle, Christopher G Brinton, and Carlee Joe-Wang. Network-aware optimization of distributed learning for fog computing. *arXiv preprint arXiv:2004.08488*, 2020.

Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481*, 2020.

Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. `https://arxiv.org/abs/1804.05271`, 2019.

Chengxu Yang, QiPeng Wang, Mengwei Xu, Shangguang Wang, Kaigui Bian, and Xuanzhe Liu. Heterogeneity-aware federated learning. *arXiv preprint arXiv:2006.06983*, 2020.

Xiaoxi Zhang, Jianyu Wang, Gauri Joshi, and Carlee Joe-Wong. Machine learning on volatile instances. *arXiv preprint arXiv:2003.05649*, 2020.

Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning on non-iid data. `https://arxiv.org/abs/1806.00582`, 2018.