FedAT: A Communication-Efficient Federated Learning Method with Asynchronous Tiers under Non-IID Data

Zheng Chai [1] **Yujing Chen** [1] **Liang Zhao** [2] **Yue Cheng** [1] **Huzefa Rangwala** [1]

Abstract

Federated learning (FL) involves training a model over massive distributed devices, while keeping the training data localized. This form of collaborative learning exposes new tradeoffs among model convergence speed, model accuracy, balance across clients, and communication cost, with new challenges including: (1) straggler problem, where the clients lag due to data or (computing and network) resource heterogeneity, and (2) communication bottleneck, where a large number of clients communicate their local updates to a central server and bottleneck the server. Many existing FL methods focus on optimizing along only one dimension of the tradeoff space. Existing solutions use asynchronous model updating or tiering-based synchronous mechanisms to tackle the straggler problem. However, the asynchronous methods can easily create a network communication bottleneck, while tiering may introduce biases as tiering favors faster tiers with shorter response latencies. To address these issues, we present **FedAT**, a novel **Fed**erated learning method with **A**synchronous **T**iers under Non-i.i.d. data. FedAT synergistically combines synchronous intra-tier training and asynchronous cross-tier training. By bridging the synchronous and asynchronous training through tiering, FedAT minimizes the straggler effect with improved convergence speed and test accuracy. FedAT uses a straggler-aware, weighted aggregation heuristic to steer and balance the training for further accuracy improvement. FedAT compresses the uplink and downlink communications using an efficient, polyline-encoding-based compression algorithm, therefore minimizing the communication cost. Results show that FedAT improves the prediction performance by up to 21.09%, and reduces the communication cost by up to $8.5\times$, compared to state-of-the-art FL methods.

1 Introduction

The number of intelligent devices, such as smartphones and wearable devices, has been rapidly growing in the last few years. Many of these devices are equipped with various smart sensors and increasingly potent hardware that allow them to collect and process data at unprecedented scales. With advanced machine learning techniques and the growth in computation power of these devices, federated learning (FL) has emerged as a novel machine learning paradigm that aims to train a statistical model among a large number of edge device nodes , as opposed to traditional machine learning training at a centralized location (McMahan et al., 2017; Konečnỳ et al., 2016). FL has been used in many application domains such as predicting human activities (Chen et al., 2019c;b), learning sentiment (Smith et al., 2017) and language processing (Li et al., 2019a).

In a typical FL framework, a shared model is learned from a federation of distributed clients [1] with the coordination of a centralized server. Different clients in a FL deployment do not share data with each other due to security and privacy reasons (Tankard, 2016; O'herrin et al., 2004). Instead, each client trains a local model using its (decentralized) local data, and the centralized server, aggregates the learned gradients of the local models to train a global model.

FL often involves a large number of clients, which feature highly heterogeneous hardware resources (CPU, memory, and network resources) and Non-i.i.d. data. The resource and data heterogeneity presents unique challenges to FL algorithms. In addition, with large number of clients, how clients communicate with server becomes a crucial design choice. Most existing FL frameworks can be divided into two communication modes: (1) synchronous communication (e.g., Federated Averaging, or FedAvg (McMahan et al., 2017)), or (2) asynchronous communication (e.g., FedAsync (Xie et al., 2019)). When there are stragglers (i.e., slower clients) in the system, which is common especially at the scale of hundreds of clients, asynchronous approaches are more robust. However, most asynchronous implementations suffer communication bottleneck as all the clients can

[1]Department of Computer Science, George Mason University, Fairfax, Virginia, USA [2]Department of Computer Science, Emory University, Atlanta, Georgia, USA. Correspondence to: Zheng Chai <zchai2@gmu.edu>.

[1]We use "clients" and "devices" interchangeably in the paper.

asynchronously talk to the server, and clients are limited by their communication bandwidth. Therefore, in this paper, we focus on two significant challenges of federated learning:

Stragglers. Recent research efforts in synchronous FL assume: i) no resource or data heterogeneity (Reisizadeh et al., 2020; Sattler et al., 2019a), or ii) all the clients are available during the whole training process (Zhao et al., 2018; Nishio & Yonetani, 2019). However, in practice, clients may come (online) and go (offline) frequently, or lag due to resource constraints and/or data heterogeneity (i.e., stragglers). Existing, synchronous FL solutions (e.g., Federated Averaging, or FedAvg (McMahan et al., 2017)) synchronously aggregates model updates, where the server has to wait for the slowest clients, therefore, leading to significantly prolonged training time.

Communication Bottleneck. To solve the straggler problems in synchronous FL, asynchronous FL approaches (Chen et al., 2019b; Xie et al., 2019) were proposed, where the server can aggregate without waiting for the straggling clients. Unlike synchronous FL where only a portion of sampled clients communicate with the server in each training round, in an asynchronous FL setting, the server communicates with all the clients asynchronously; therefore, the server can easily become a communication bottleneck with tens of thousands of clients updating the model simultaneously.

To overcome the deficiencies described above, we design and implement FedAT, a novel communication-efficient FL approach that combines the best of both worlds – synchronous and asynchronous FL training – using a tiering mechanism. In FedAT, the clients involved in a FL deployment are partitioned into logical tiers based on their response latency (i.e., the time a client takes to finish a single round of training). All the logical tiers in FedAT participate in the global training simultaneously, with each tier proceeding at its own pace. Clients within a single tier update a model associated with that particular tier in a synchronous manner, while each tier, as a logical, coarse-grained, training entity, updates a global model asynchronously. Faster tiers, with shorter per-round response latency, drive the global model training to converge faster; slower tiers get involved in the global training by asynchronously sending the model updates to the server, so as to further improve the model's prediction performance.

Uniformly aggregating the asynchronously updated tier model into the global model may result in biased training (biased towards the faster tiers), as more performant tiers tend to update the global model more frequently than the slower tiers. To solve this issue, we propose a new weighted aggregation heuristic, which assign higher weight to slower tiers. Furthermore, to minimize the communication cost introduced by asynchronous training, FedAT compresses the model data transferred between the clients and the server using Encoded Polyline Algorithm. In a nutshell, FedAT synergizes the four components together, namely, the tiering scheme, asynchronous inter-tier model updating, the weighted aggregation method, and polyline encoding compression algorithm, to maximize both the convergence speed and the prediction performance while minimizing communication cost.

We make the following contributions in this paper:

- We design and implement FedAT, a novel tiered FL framework that updates local model parameters synchronously within tiers and update the global model asynchronously across tiers.
- We propose a new optimization objective with a weighted aggregation heuristic, which the FL server uses to speed up model convergence and improve the prediction performance by balancing the model parameters from different tiers.
- We provide rigorous theoretical analysis for our proposed method for both convex and non-convex objectives. We show that FedAT has provable convergence guarantee.
- We utilize a lossy compression technique, polyline encoding, to compress the transferred model data between the clients and the server to reduce the communication cost without affecting the model accuracy.
- We evaluate FedAT extensively on a 100-client cluster. Experimental results on three real-world federated datasets show that FedAT improves the prediction accuracy by up to 21.09%, exhibits significantly less accuracy variance during the training, and reduces the communication cost by up to $8.5\times$ compared with the baseline FL methods.

2 Related Work

2.1 Stragglers in Federated Learning

The main premise of FL has been collective learning using a network of common devices such as smartphones and tablets. Thus, the assumption made by FedAvg that all clients are available during the whole training process is not practical. Li et al. (Li et al., 2019b) suggest to select a smaller ratio of participated clients for each global iteration to alleviate the straggler's effect, while with more communications on model convergence. TiFL (Chai et al., 2020) is a tiered FL framework that uses a synchronous, intra-tier model updating scheme similar to that used in FedAvg. TiFL's favoring-faster-tier strategy may result in biased training and lower model accuracy. FedAT uses TiFL's tiering approach, but differs against TiFL in that FedAT combines the intra-tier synchronous training with cross-tier asynchronous training to effectively avoid biased training. Asynchronous FL frameworks (Xie et al., 2019; Chen et al., 2019a;b) that allow wait-free communication and computation have been proposed to tackle the stragglers problem. These asynchronous approaches, however, suffer from the communication bottleneck issue as they require more frequent communications between all the clients and the server.

2.2 Communication-efficient Federated Optimization

McMahan et al. propose a FL approach called FedAvg (McMahan et al., 2017), where instead of communicating after every iteration, each client performs multiple iterations of SGD to compute a weight update. By reducing the communication frequency, FedAvg reduces the communication cost and can work with partial client participation. In a follow-up work, Konečnỳ et al. (Konečnỳ et al., 2016) propose two approaches to reduce the uplink communication costs, i.e., structured updates and sketched updates, combined with probabilistic quantization. This approach, however, significantly slows down the convergence speed in terms of SGD iterations.

In the broader realm of communication-efficient distributed deep learning, a wide variety of methods has been proposed to reduce the amount of communication during the training process. Chen et al. (Chen et al., 2019c) propose a layerwise asynchronous update scheme that updates the parameters of the deep layers less frequently than those of the shallow layers. Mills et al. (Mills et al., 2019) adapt FedAvg to use a distributed form of Adam optimization and compress the uploaded parameters. Jeong et al. (Jeong et al., 2018) develop federated distillation that follows an online version of knowledge distillation to compress the model. Reisizadeh et al. (Reisizadeh et al., 2020) present a periodic averaging and quantization approach to reduce communication costs. However, these works only target uplink communication compression and are developed for synchronous frameworks that neglect the real-life scenario in which the straggler problem is a common phenomenon in learning on end devices. Besides the above mentioned server-worker topology, communication bottleneck via quantization and compression in distributed learning has also gained considerable attention in serverless approaches (Koloskova et al., 2019; Wang et al., 2019; Reisizadeh et al., 2019). While such techniques can be used to reduce communication cost in federated learning, the network topology without a server in distributed learning is fundamentally different.

Our proposed communication-efficient federated learning framework combines synchronous and asynchronous updates together to mitigate the challenge associated with stragglers and improve model convergence rate. We apply a weighted aggregation strategy on server to improve the model's prediction performance and compress both the uplink and downlink communications.

3 Background: Federated Learning

3.1 Preliminaries: Federated Learning and FedAvg

FL algorithms involve hundreds to millions of remote devices training locally on their device-generated data, and collectively train a global, shared model, under the coordination of a centralized server serving as an aggregator. In particular, the FL algorithm optimizes the following objective function:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{N} F_k(w), \tag{1}$$

where $F_k(w) \overset{def}{=} \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell_i(x_i, y_i; w)$, is the local empirical loss of client k, and $\ell_i(x_i, y_i; w)$ is the corresponding loss function for data sample $\{x_i, y_i\}$. K is the total number of devices. $\mathcal{D}_k$ for $k \in \{1, \ldots, K\}$ denotes data samples stored locally on device k. $n_k = |\mathcal{D}_k|$, is the number of data samples on device k; and $N = \sum_{k=1}^{K} |\mathcal{D}_k|$ is the total number of data samples stored on K devices. Assuming for any $k \neq k^{'}$, $\mathcal{D}_k \bigcap \mathcal{D}_{k^{'}} = \varnothing$.

The ultimate goal is to find a model w_* that minimizes the objective function:

$$w_* = \arg\min f(w). \tag{2}$$

FedAvg (McMahan et al., 2017) is a commonly used method to solve the optimization problem defined in Equation 2 in a non-convex setting with a synchronous update fashion. This method runs by randomly sampling a subset of clients with a certain probability at each round; each selected client performs E epochs of training locally on its own data using an optimizer such as stochastic gradient descent (SGD). This kind of local update methods enable flexible and efficient communication compared to traditional mini-batch methods (Yu et al., 2018; Wang & Joshi, 2018; Woodworth et al., 2018).

In a typical real-world scenario, the data stored across devices follow a **Non-i.i.d.** (non-independent and identical) distribution; that is, the training data distributed across the clients are often non-uniform, since the data of a given client is typically based on the usage of the particular edge device and will not be representative of the population distribution (McMahan et al., 2017). Although FedAvg can work with partial client participation at each training round, training on Non-i.i.d. data may lead each client towards its local optimal model as opposed to achieving a global optimal one.

In addition, slow clients (stragglers), which perform local training at a relatively slower speed (due to weaker computing resources and/or larger local data size), may have poor prediction performance due to less training, and thus may prevent the shared model from converge to a global optimal solution. Therefore, solving Equation 2 in this synchronous manner can implicitly introduce high variance in prediction performance given the existence of straggling clients. To better evaluate the robustness of FL training with stragglers, we define new metrics as a measure of the straggler tolerance level in a typical FL setup.

Definition 3.1. (*Robust training with straggling clients*). For two trained models w and w', we say that model w is more robust against straggling clients than model w', if (1) model w *converges faster* than model w'; (2) the test accuracy of model w for the K clients, $\{P_1, ..., P_K\}$, exhibits *lower variance* than that of model w' for the same set of K clients (where P_c represents the test accuracy of the model w over the testing data for client c), and (3) model w has better prediction performance than model w'.

As discussed in the previous sections, the existence of stragglers cause longer training times and prevent the model from converging to the optimal solution. The rational of using these three metrics, namely, convergence speed, accuracy variance, and prediction accuracy, as a means to quantify the robustness of a FL approach against stragglers, is that: (1) stragglers not only cause slow convergence speed, but also a loss in prediction performance, and (2) existing literature fail to comprehensively consider all these aspects (Chai et al., 2020; Chen et al., 2019c; Jeong et al., 2018; Xie et al., 2019; Chen et al., 2019a;b).

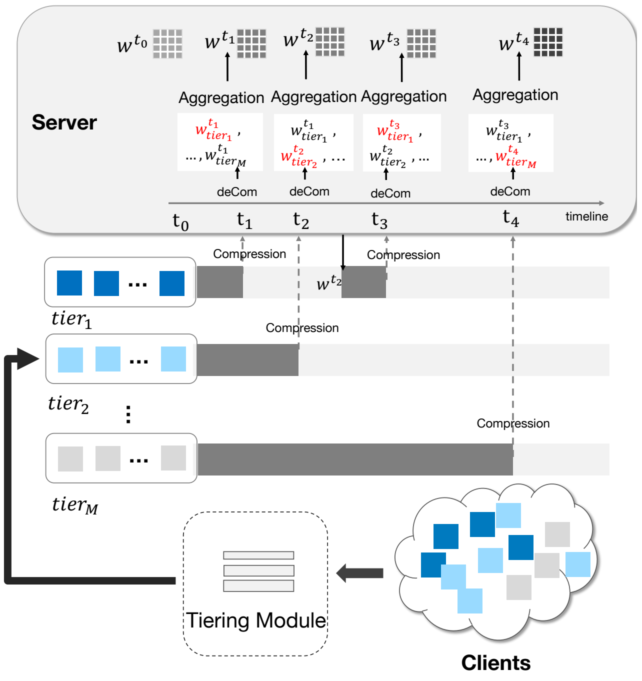


Figure 1. Overview of FedAT. $tier_1, ..., tier_M$ are M tiers, and $w^t_{tier_1}, ..., w^t_{tier_M}$ are their according weights, respectively. `deCom` denotes the decompression process of clients' models in a certain tier on the server.

4 FedAT: Federated Learning with Asynchronous Tiers

FedAT consists of three main components: (1) a centralized server for global model synchronization; (2) a group of clients that are logically partitioned into different performance tiers; and (3) a tiering module that profiles clients' training performance and performs client tiering based on the response latency of each client.

We next illustrate the FL training process of FedAT (as depicted in Figure 1 and listed in Algorithm 1). The tiering module profiles and partitions the involved clients into M tiers based on their response latencies: $\{tier_1, tier_2, ..., tier_M\}$, where $tier_1$ is the fastest tier and $tier_M$ is the slowest tier. The server maintains a list of M models, $\{w^t_{tier_1}, ..., w^t_{tier_M}\}$, one for each tier, reflecting the most updated view of per-tier local models, at a certain round t. Correspondingly, the server also maintains a global model w that gets asynchronously updated from M tiers.

Each tier performs synchronous update process, a fraction of clients (S) are selected randomly and compute the gradient of the loss on their local data, then send the compressed weights to the server for a synchronous and update the tier model on server.

Figure 1 shows an example of the intra-tier synchronous and cross-tier asynchronous training process. At time t_1, the clients in $tier_1$ quickly finish their local training, compress their trained models and send to the server. The server then performs the following steps: (1) decompresses the local models received from $tier_1$, (2) applies synchronous update to the received models of $tier_1$ and get $w^{t_1}_{tier_1}$ (highlighted in red color in Figure 1), and (3) aggregates the latest updates sent from all the tiers (including $tier_1$) using a weighted average aggregation method (see §4.1), to generate a new global model w^{t_1}.

At time t_2, the last client of $tier_2$ finishes its local training and sends the compressed model to the server. The server follows the same procedure as $tier_1$ to get a new global model w^{t_2}. Then the server sends the latest global model w^{t_2} to the next ready tier (in this example $tier_1$) and starts the next round. Note that a tier in FedAT directly interacts with the server to update the global model whenever it finishes a round of local training, thus forming an *asynchronous*, cross-tier process.

Since clients are partitioned into tiers based on their response latencies and the tiers asynchronously update the global mode, stragglers may not become a performance bottleneck that would otherwise slow down the global training progress. However, as the server interacts more frequently with the faster tiers than with the relatively slower ones, this would inevitably lead to biases towards the faster tiers. To address this issue, we introduce a new objective onto the server-side optimization, which uses a weighted aggregation strategy to more fairly balance the mode updating processes from different tiers.

4.1 Cross-Tier Weighted Aggregation

A straightforward idea to achieve unbiased, more balanced training is to assign relatively higher weights to slower tiers that update less frequently, so that the global model would not bias towards the faster tiers. To this end, FedAT uses a new cross-tier, weighted aggregation heuristic, which dynamically adjusts the relative weights assigned to each tier based on the number of times a tier has updated the global mode. The goal of the weighted aggregation heuristic is to help the global training converge faster.

Assuming there are M tiers, the number of updates from each tier till now is $T_{tier_1}, T_{tier_2}, ..., T_{tier_M}$, respectively, and the total number of updates from all the tiers till now is $T_{tier_1} + T_{tier_2} + ... + T_{tier_M} = T$, we define the objective function of FedAT as:

$$f(w) = \sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} f_{tier_m}(w), \tag{3}$$

where $\frac{T_{tier_{(M+1-m)}}}{T}$ is the relative weight of $tier_m$, and $\sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} = 1$. To understand the heuristic, a relatively slower tier with a tier number m would get assigned a relatively larger weight value, $\frac{T_{tier_{(M+1-m)}}}{T}$, as $M+1-m$ corresponds to a relatively faster tier, $tier_{(M+1-m)}$, whose historical updating frequency $T_{tier_{(M+1-m)}}$ is expected to be higher. In this way, FedAT can dynamically steer and balance the global model training, avoid potential bias towards a subset of faster tiers, and effectively improve the convergence rate.

$f_{tier_m}(w)$ is the weighted average of the models from the selected clients within $tier_m$. Assuming at round t, $tier_m$ happens to be in communication with the server, we have the update of $f_{tier_m}(w)$ as follow:

$$f_{tier_m}(w) = \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} F_k(w), \tag{4}$$

where $\mathcal{S}_t$, $|\mathcal{S}_t|$, and N_c denote a subset of randomly selected clients in $tier_m$, the number of selected clients in $tier_m$, and the total number of data samples in $\mathcal{S}_t$, respectively.

4.2 Training at Local Clients

As mentioned in §3.1, for training with Non-i.i.d. data, frequent local updates may potentially cause the local models to diverge due to the varying updating frequency of different tiers and the underlying data heterogeneity. We propose to add a constraint term to the local subproblem by restricting the local updates to be closer to the global model. Thus, instead of just minimizing the local function F_k , client k applies an update with the constraint using the following surrogate objective h_k:

$$h_k(w_k) = F_k(w_k) + \frac{\lambda}{2}||w_k - w||^2, \tag{5}$$

where w_k, w are the local model of client k and server model, respectively. Then we will update (4) as:

$$\begin{aligned} f_{tier_m}(w) &= \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} h_k(w_k) \\ &= \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} (F_k(w_k) + \frac{\lambda}{2}||w_k - w||^2). \end{aligned} \tag{6}$$

The constraint term addresses the issue of Non-i.i.d. by restricting the local updates to be closer to the global model. In the ideal situation, with $\lambda = 0$, and all clients share the same latency, thus we get one tier and FedAT becomes FedAvg. The approach of FedAT is detailed in Algorithm 1.

Algorithm 1: FedAT's Training Process

```
Input: $w_{tier_m}$, $t$, $T$ and $T_{tier_m}$. $w_{tier_m}$ denotes the
       weights of Tier $m$. $t$ represents the global
       round $t$. $T$ is the maximum global rounds.
       $T_{tier_m}$ is the number of updates of tier $m$
Server: Initialize $w_{tier_1}, w_{tier_2}...w_{tier_M}$ to $w^{t_0}$.
  Initialize $t, T_{tier_1}...T_{tier_M}$ to 0
for each tier $m \in M$ in parallel do
    while $t < T$ do
        $w$ = WeightedAverage($w_{tier_1}, w_{tier_2}...w_{tier_M}$)
        $\mathcal{S}_m$ = (random set of clients from tier $m$)
        for each client $k \in \mathcal{S}_m$ in parallel do
            $n_k = |\mathcal{D}_k|$
            $w_k^{t+1} = w_k^t - \eta \nabla h(w^t)$
        $N_c = \sum_{k=1}^{|\mathcal{S}_m|} n_k$
        $w_{tier_m} = \sum_{k=1}^{|\mathcal{S}_m|} \frac{n_k}{N_c} * w_k^{t+1}$
        $T_{tier_m} = T_{tier_m} + 1$
        $t = t + 1$
function
  WeightedAverage($w_{tier_1}, w_{tier_2}...w_{tier_M}$)
if $t == 0$ then
    return $w^{t_0}$
else
    return $\sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} * w_{tier_m}$
```

4.3 Compression, Marshalling, and Unmarshalling

Previous works on communication-efficient FL mentioned in §2 almost exclusively consider i.i.d. data distribution among the clients, which is not practical in real-world scenarios of FL, where the client data typically follows a Non-i.i.d. distribution (McMahan et al., 2017). As studied in (Sattler et al., 2019a), many compression methods (Bernstein et al., 2018; Sattler et al., 2019b) suffer from slow convergence rates in the Non-i.i.d. cases. Non-i.i.d. often introduces divergence of model weights collected from resource-heterogeneous clients. Due to such divergence, lossy compression methods such as quantization and dequantization (Zhang et al., 2020) may inevitably lead to huge errors and global performance reducing. Furthermore, as asynchronous FL methods aggregate more frequently than synchronous FL methods, highly frequent updates drastically amplify such divergence, and result in a poor global performance. Therefore, with frequent communications in asynchronous FL approaches, it is crucial to select a compression technique that can efficiently reduce the communication cost while effectively guaranteeing a fast convergence to the optimal solution.

To this end, we design a simple yet effective compression scheme based on Encoded Polyline Algorithm [2] (or polyline encoding). Polyline encoding is a lossy compression algorithm that converts a rounded binary value into a series of character codes for ASCII characters using the base64 encoding scheme. It can be configured to maintain a configurable precision by rounding the value to a specified decimal place. More importantly, it can achieve a high compression ratio of up to $3.5\times$ under the FL communication scenarios (we evaluate the effectiveness of compression in §6.3). FedAT compresses both the uplink and downlink communications. The process is as follow: (1) FedAT flattens (marshalling) the weights of each layer to get a list of decimal values. (2) Then, using polyline encoding, every decimal value in the list gets converted into a compressed ASCII string; along with the compressed weights, the dimensions of the weights of each layer is transmitted as well. (3) When the server/clients receive the compressed weights, a decompression process is performed, and then the decompressed weights are reshaped back (unmarshalling) to the original dimensions based on the dimension information received.

5 Convergence Analysis

In this section, we show that FedAT converges to the global optimal solution for both strongly convex and non-convex functions on Non-i.i.d. data. First, we introduce some definitions and assumptions as follow.

[2]`https://developers.google.com/maps/documentation/utilities/polylinealgorithm`

Definition 5.1. (Smoothness) The function f has Lipschitz continuous gradients with constant $L > 0$ (in other words, f is *L-smooth*) if $\forall x_1, x_2$,

$$f(x_1) - f(x_2) \leq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{L}{2}||x_1 - x_2||^2. \quad (7)$$

Definition 5.2. (Strong convexity) The function f is μ-*strongly convex* with $\mu > 0$ if $\forall x_1, x_2$,

$$f(x_1) - f(x_2) \geq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{\mu}{2}||x_1 - x_2||^2. \quad (8)$$

Definition 5.3 has been made by the work (Li et al., 2018).

Definition 5.3. (γ-inexactness) For a function $h(w) = F(w) + \frac{\lambda}{2}||w - w_0||^2$, and $\gamma \in [0, 1]$. w^* is a γ-inexact solution for $\min_w h(w)$ if $||\nabla h(w^*)|| \leq \gamma ||\nabla h(w_0)||$, where $\nabla h(w) = \nabla F(w) + \lambda(w - w_0)$.

According to (Li et al., 2020), this definition aims to allow flexible performance of local clients in each communication round, such that each of the local objectives can be solved inexactly. The amount of local computation vs. communication can be tuned by adjusting the number of local iterations, i.e., more local iterations indicates more exact local solutions.

Further, we make the following assumptions on the objective functions:

Assumption 5.1. *The central objective $f(w)$ is bounded, i.e., $\min f(w) = f(w_*) > -\infty$*

Assumption 5.2. *The expected squared norm of stochastic gradients is uniformly bounded, i.e., there exists a scalar G, such that $\mathbb{E}||\nabla F(w^t)||^2 \leq G^2$, all $t = 0, ..., T-1$.*

Assumption 5.3. *With $\bar{g}_t(w^t)$ ($\bar{g}_t = \sum_{k=1}^{m} \frac{n_k}{N_c} \nabla h_k(w^t)$) as the averaged gradients from certain tier with m clients, there exists a scalar $\sigma > 0$ such that $\nabla f(w^t)^\top \mathbb{E}(\bar{g}_t(w^t)) \geq \sigma ||\nabla f(w^t)||^2$.*

Assumption 5.1 is easy to satisfy as there exists a minimum value for the central objective $f(w)$. The conditions in Assumption 5.2 on the variance of stochastic gradients is customary. While this is a much weaker assumption compared to the one that uniformly bounds the expected norm of the stochastic gradient. Assumption 5.3 ensure that the gradient of local tier $\bar{g}_t$ is an estimation of $\nabla f(w^t)$. And as $\sigma = 1$, we have $\bar{g}_t$ as the unbiased estimation of $\nabla f(w^t)$.

To convey our proof clearly, we first introduce and prove certain useful lemmas.

Lemma 5.1. *With Definition 5.3, the local functions $h(\cdot)$ are γ-inexact. For aggregated tier model $\bar{g}_t(w^t)$, we have*

$$\mathbb{E}||\bar{g}_t(w^t)||^2 \leq \gamma^2 G^2 c^2, \quad (9)$$

where c is the total number of clients within the given tier.

Lemma 5.2. *If $f(w)$ is μ-strongly convex, then with Definition 5.2, we have:*

$$2\mu(f(w^t) - f(w_*)) \leq ||\nabla f(w^t)||^2. \tag{10}$$

We show that the averaged model of local tier is bounded in Lemma 5.1. The detailed proof is in Appendix A.1. While the proof of Lemma 5.2 is supported by the literature (Nesterov, 2013; Bottou et al., 2018), we also provide a detailed proof in Appendix A.2.

Theorem 5.1. *Suppose that the central objective function $f(w)$ is L-smooth and μ-strongly convex. The local functions $h(\cdot)$ are γ-inexact. Let Assumption 5.2 and Assumption 5.3 hold. After T global updates on the server, FedAT converges to a global optimum w_*:*

$$\begin{aligned}&\mathbb{E}[f(w^T) - f(w_*)]\\ &= (1 - 2\mu B\eta\sigma)^T(f(w^0) - f(w_*)) + \frac{L}{2}\eta^2\gamma^2B^2G^2c^2,\end{aligned} \tag{11}$$

where c is the total number of clients within one tier, and $B = \frac{T_{tier(M+1-m)}}{T} \leq 1$.

We direct the reader to Appendix A.3 for a detailed proof. The convergence bound in Theorem 5.1 depends on local constrain μ, weighted parameter B and learning rate η. Note that B varies in each global iteration because the update number of each tier changes at every global iteration. A also varies due to the data heterogeneity of each tier.

Theorem 5.2. *Suppose that the central objective function $f(w)$ is L-smooth and non-convex. The local functions $h(\cdot)$ are γ-inexact. Let Assumption 5.1,Assumption 5.2 and Assumption 5.3 hold, then after T global updates we have:*

$$\begin{aligned}&\sum_{t=0}^{T-1} B\mathbb{E}[||\nabla f(w_t)||^2]\\ &\leq \frac{f(w^0) - f(w_*)}{\eta\sigma} + \frac{L}{2\sigma}T^2\eta\gamma^2BG^2c^2,\end{aligned} \tag{12}$$

where c is the total number of clients within one tier, and $B = \frac{T_{tier(M+1-m)}}{T} \leq 1$.

6 Evaluation

6.1 Experimental Setting

Federated Datasets. We evaluate FedAT using three different real-world federated datasets on both the convex and non-convex models described as follows:

- CIFAR-10: We train a convolutional neural network (CNN) on the 10-class CIFAR-10 (Krizhevsky et al., 2009) dataset. The network architecture includes three convolutional layers, each with 32, 64 and 64 filters, followed by two fully connected layers with units of 64 and 10. We partition the dataset into 100 clients and follow the same Non-i.i.d. setting of CIFAR-10 as (McMahan et al., 2017).
- Fashion-MNIST: We use a 10-class Fashion-MNIST dataset (Xiao et al., 2017) to train an image classification model with the same mode structure, number of clients and Non-i.i.d. setting as CIFAR-10. The input of the model is a flattened 784-dimensional (28 $\times$ 28) image, and the output is a class label between 0 and 9, with each label corresponding to one cloth type.
- Sentiment140: In order to evaluate the model performance under a convex setting, we train a logistic regression model on a text sentiment analysis task on tweets from the Sentiment140 (Go et al., 2009) dataset. Each twitter account corresponds to a client.

FL Methods. We compare FedAT against three synchronous and asynchronous FL methods:

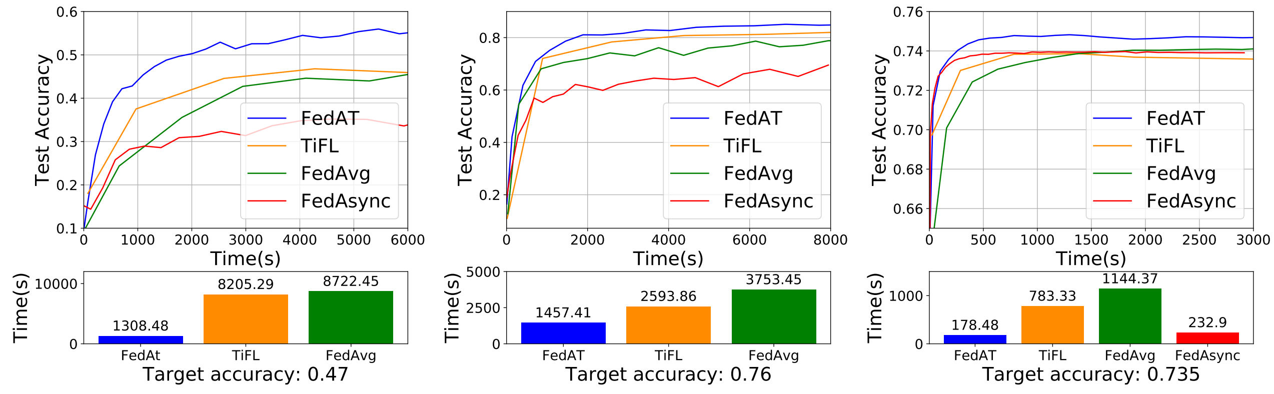
- FedAvg (McMahan et al., 2017): A baseline synchronous FL method proposed by McMahan et al.
- TiFL (Chai et al., 2020): A synchronous FL method that partitions training clients into different tiers based on their responding latency. TiFL's server uses the same aggregation method as that of FedAvg.
- FedAsync (Xie et al., 2019): A baseline asynchronous FL method using weighted averaging to update the server model.

Implementation and Setup. We have implemented FedAT and the comparison methods all in TensorFlow (Abadi et al., 2016). We simulate a FL setup using a 192-core Chameleon Cloud cluster, which consists of three bare-metal machines, each equipped with a 64-core Intel® Xeon® Gold 6242 CPU, and 192 GB main memory. We deploy the FL server exclusively on one machine, and all clients on the rest two, where each client gets assigned one CPU core. We evaluate 100 clients in most tests, unless otherwise specified. FedAT is configured to use `Precision 4` as the precision of the compressor (§6.3.2) throughout the evaluation.

Hyperparameters. We randomly split each client's local data into an 80% training set and a 20% testing set. For intra-tier synchronous training, we adopt the same sampling scheme as FedAvg: sampling clients (within a particular tier) randomly at each round. We use Adam (Kingma & Ba, 2014) as the local solver and set the local constrain parameter λ as 0.4. For each dataset, we tune the learning rate on FedAvg (*local epoch* $E = 3$, *batch size* $= 10$) and use the same learning rate for all the four FL methods on that dataset. We set the number of randomly selected clients as 10 for FedAvg, TiFL and FedAT on all datasets.

Table 1. Comparison of prediction performance and variance to baseline approaches. `#class` indicates the number of labels (i.e., classes) each client has. The `Accuracy` columns show the best prediction accuracy that each FL approach reaches after each model converges. The `Norm.Var.` columns show the average variance of test accuracy among all clients, normalized to that of FedAT. We show FedAT's absolute variance values (`Abs.Var.`). We show the absolute values for FedAT's accuracy variance. `impr.(a)` and `impr.(b)` are the accuracy improvement of FedAT compared with the best and worst baseline FL method, respectively. The best performance results are highlighted in bold font.

Dataset (#class)	**FedAvg**		**TiFL**		**FedAsync**		**FedAT**			
	Accuracy	Norm. Var.	Accuracy	Norm. Var.	Accuracy	Norm. Var.	Accuracy	Abs. Var.	impr.(a)	impr.(b)
CIFAR-10 (#2)	0.547	2	0.527	1.26	0.480	2	**0.591**	**0.0042**	7.44%	18.78%
CIFAR-10 (#4)	0.628	5.07	0.615	2.79	0.541	3.93	**0.633**	**0.0014**	0.79%	14.53%
CIFAR-10 (#6)	0.654	4.33	0.654	1.33	0.531	2.08	**0.673**	**0.0012**	2.82%	21.09%
CIFAR-10 (#8)	0.667	3.1	0.655	1.3	0.561	1.54	**0.681**	**0.0010**	2.05%	17.62%
CIFAR-10 (i.i.d.)	0.686	4.23	0.685	2.12	0.567	2.69	**0.701**	**0.00052**	2.13%	19.11%
Fashion-MNIST (#2)	0.842	1.86	0.859	1.29	0.795	2	**0.873**	**0.007**	1.60%	8.93%
Sentiment140	0.741	3.72	0.739	2.75	0.740	5.69	**0.748**	$\mathbf{2.67e^{-5}}$	0.93%	1.20%



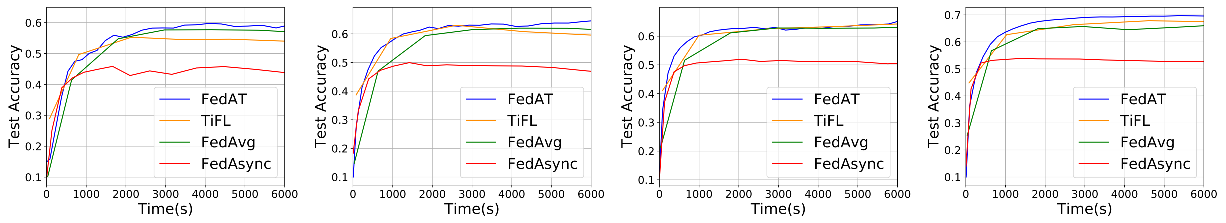
(a) CNN @ CIFAR10. (b) CNN @ Fashion-MNIST. (c) Logistic @ Sentiment140.

Figure 2. Performance comparison of different FL methods on 2-class Non-i.i.d. CIFAR10, Fashion-MNIST, and Sentiment140 datasets. **Above (test accuracy timeline curves):** The results are average-smoothed for every 40 global rounds. **Bottom (bar charts):** the time it takes for each evaluated FL methods to reach a target accuracy of $N\%$ as specified in the X-axis' labels. (Note that FedAsync is not able to reach the target accuracy for Cifar10 and FMnist, thus is omitted.)

Simulating Different Performance Tiers. We add random delays to the computations conducted by clients to simulate different levels of straggler effects that could be potentially caused by weaker computing powers and intermittent network connections in a real-world FL setup. We first evenly divide all the clients into 5 parts, then randomly assign delays of $0s$, $0 \sim 5s$, $6 \sim 10s$, $11 \sim 15s$ and $20 \sim 30s$ to the clients in each part at every round, respectively. Each part is called one *tier*. To guarantee fair comparison, each client, once selected, would follow a fixed, pseudo-random mini-batch schedule. The same strategy is applied to all the FL methods that we test (including FedAT's intra-tier synchronous training). Furthermore, to simulate unstable network connections, for all the tests that we run, we randomly select 10 "unstable" clients, which would drop out at any time during the training process. Once the client drops out, it will not come back and rejoin the training process again.

6.2 Prediction Performance

Table 1 presents the results of the prediction performance and the variance of the test accuracy on all the datasets. We report the best test accuracy after each training process converges within a global iteration budget. For the 2-class CIFAR10 dataset, FedAT outperforms the best baseline FL method, FedAvg, by 7.44%, and the worst baseline method, FedAsync, by 18.78%. Using the same tiering scheme as TiFL, FedAT achieves consistently higher accuracy than TiFL for all the experiments. This is because: (1) the local constraint forces local models to be closer to the server model, and (2) FedAT's new weighted aggregation heuristic can more effectively engage the straggling clients from the slower tiers, leading to better prediction performance (we evaluate the effectiveness of our weighted aggregation method in §6.4). FedAvg has the closest prediction performance as TiFL, because they both follow the same synchronous updating strategy. FedAsync, on the other hand, performs the worst, as it simply aggregates weights from one client at a round and has no effective way to deal with



(a) CIFAR10 Non-i.i.d. (4). (b) CIFAR10 Non-i.i.d. (6). (c) CIFAR10 Non-i.i.d. (8). (d) CIFAR10 i.i.d.

Figure 3. Convergence speed comparison on CIFAR10 over different level of Non-i.i.d-ness. The results are average-smoothed for every 40 global rounds.

Table 2. The total amounts of data (MB) transferred between clients and the server in order to achieve the target accuracy on all datasets with 2-class Non-i.i.d. case. − means that the FL method is not able to achieve the accuracy target within the iteration budget. The best results are highlighted in bold font.

Method	CIFAR10 (acc. = 0.50)	FMNIST (acc. = 0.79)	Sentiment140 (acc. = 0.73)
FedAvg	1828.54 / 1827.96	1048.25 / 1048.56	16.71 / 16.78
TiFL	2140.71 / 2139.19	1041.98 / 1041.53	17.20 / 17.25
FedAsync	− / −	9895.53 / 9898.54	82.27 / 82.24
FedAT	**1675.82 / 1671.43**	**1041.54 / 1041.52**	**16.41 / 16.39**

stragglers. The performance difference can also be clearly noticed from the convergence timeline graphs shown in Figure 2. FedAT converges faster towards the optimal solution than all other three compared methods on both the non-convex and convex objectives.

6.2.1 *Impact of Non-i.i.d. Level*

The models' convergence behaviors are sensitive to the degree of Non-i.i.d. of the data distribution across clients. Table 1 shows that, for the CIFAR10 dataset, the test accuracy increases as the degree of Non-i.i.d. decreases (i.e., the number of classes per client increases); accordingly, the variance of the test accuracy decreases as the degree of Non-i.i.d. decreases (i.e., the data is more evenly shuffled and each client covers all the classes). Figure 2(a) (the timeline charts above) and 3 together show a sensitivity analysis of the convergence rate as a function of the Non-i.i.d. (from two classes per client, to 8 classes per client, to the i.i.d. case), on the CIFAR10 dataset. We observe that FedAT outperforms all the other three FL methods with higher prediction performance across all different Non-i.i.d. levels. The most distinct performance gap between FedAT and the other FL methods can be observed in the 2-class Non-i.i.d. case, where each individual client holds only 2 classes of data. Notably, FedAT improves the prediction performance by as much as 8.04% compared to FedAvg.

6.2.2 *Robustness to Stragglers*

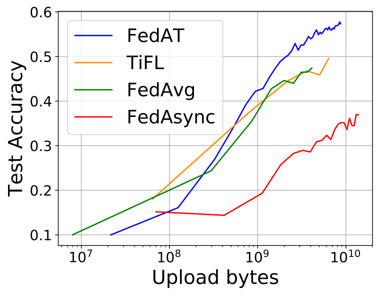
As defined in in Definition 3.1, the robustness of a FL method against stragglers can be quantified using the variance on the prediction performance and the convergence speed. Table 1 shows that FedAT has consistently the lowest accuracy variance across all experiments. FedAvg observes significantly higher accuracy variance, which are 1.86-5.07$\times$ higher than that of FedAT. This is due to the compound effect of both synchronous training and stragglers – synchronous training determines that during each round only a subset of clients can get involved to contribute to the global training, while the straggling clients are more likely to have a less accurate model when they next get selected (since they receive less training) by the server for training, thus causing huge accuracy fluctuation of the global model.

The bar charts in Figure 2 presents a comparison of the training time it takes for each FL method to achieve a target test accuracy. For example, as shown in Figure 2(a) (bar chart at bottom), to reach an accuracy of 47% for the CIFAR10 CNN model, TiFL and FedAvg spend $5.27\times$ and $5.67\times$ longer time than FedAT. Fashion-MNIST show a similar trend. For Sentiment140, FedAvg, TiFL and FedAsync take $1.3\times$,$4.38\times$ and $6.41\times$ longer time than FedAT, respectively.

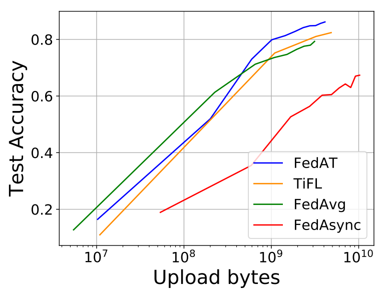
6.3 Communication Efficiency

6.3.1 *Communication Cost*

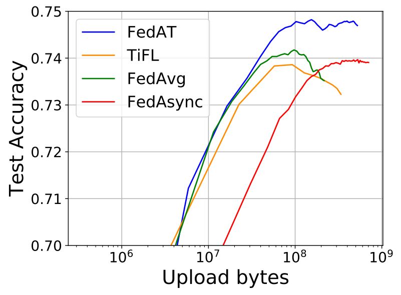
We next evaluate the network communication cost of FedAT in terms of the amounts of data transferred via network. Table 2 shows the amounts of data transferred between the clients and the server (i.e., counting both model uploading and downloading) in order to achieve the target accuracy. FedAsync incurs the highest communication cost – about $9.5\times$ of FedAT, or is not able to reach the target prediction performance. This confirms that severe communication bottleneck problem exists in asynchronous FL methods, where the server simply communicates with all the clients. FedAvg and TiFL have similar communication cost as they both use the same synchronous updating mechanism. FedAT incurs the lowest communication cost with compression technique and the proposed weighted aggregation on server.



(a) CIFAR10.

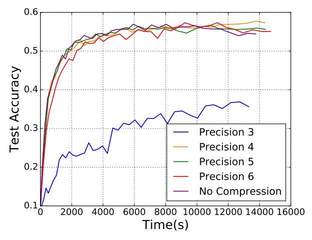


(b) Fashion-MNIST.

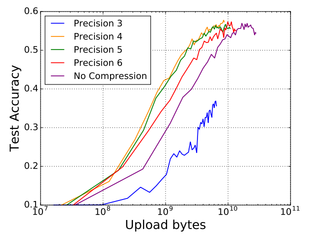


(c) Sentiment140.

Figure 4. Test accuracy as a function of the cumulative amounts of data uploaded from the clients to the server for 2-class Non-i.i.d. datasets. The performance curves are average-smoothed for every 40 global rounds. The X-axis is in log-scale.



(a) Test accuracy.



(b) Uploaded data vs. accuracy.

Figure 5. Impact of FedAT's compression precision on the prediction performance and the communication cost, for the CIFAR10 Non-i.i.d. 2-class dataset. All results are plotted with the average of every 40 global rounds.

Figure 4 further compares the uploaded bytes (from clients to the server) needed to reach a certain test accuracy. To achieve a relatively higher accuracy, FedAT needs fewer bytes than all the other three FL methods. More importantly, to achieve the same prediction performance for the CIFAR10 2-class Non-i.i.d. dataset, FedAT requires up to 1.28× less data uploaded to the server, again demonstrating the efficiency and effectiveness of the model compression method used by FedAT.

6.3.2 *The Accuracy vs. Communication-Cost Tradeoff*

Next, we explore the accuracy vs. communication-cost tradeoff by varying the precision of FedAT's compressor. `Precision 3` (i.e., a precision of three decimal places) leads to the worst prediction performance, as shown in Figure 5. This is because compressing the model by keeping only three digits after the decimal loses much information that is needed to converge the model; as a result, more training rounds, and more data communication, are needed in order to achieve a desirable accuracy. `Precision 4` is robust enough to strike a balance between the prediction performance and communication efficiency. `Precision 4` approaches the optimal accuracy achieved when no compression is used (Figure 5(a)), while effectively reducing the amount of data uploaded by 36.41% and 67.3% (given the same target accuracy of 50%) compared to `Precision 6` and `No Compression`, respectively (Figure 5(b)). FedAT achieves a compression ratio (i.e., the ratio of the data size after compression and before compression) up to 3.5× overall. FedAT is configured to use `Precision 4` as the default compression configuration in all the other experiments.

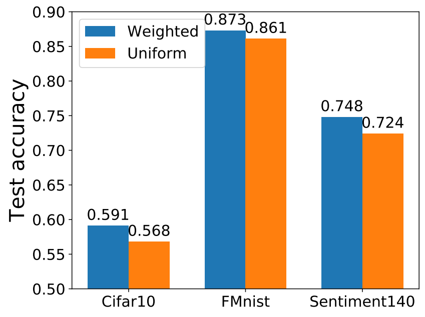


Figure 6. Comparison of FedAT's weighted aggregation heuristic vs. a uniform baseline approach that assign uniform weights when aggregating the models from different tiers.

6.4 Effectiveness of Weighted Aggregation

Finally, we validate the effectiveness of our weighted aggregation heuristic. Weighted aggregation assigns more weight to the tiers that participate in the global training less frequently to prevent training bias towards the faster tiers. As shown in Figure 6, the weighted aggregation heuristic improves the best test accuracy by 1.39% to 4.05%, compared to the baseline case, for the three datasets, demonstrating the effectiveness of the proposed approach.

7 Conclusion

We have presented FedAT, a new FL method that maximizes the prediction performance and minimizes the communication cost using a tiered, hybrid synchronous-asynchronous training model. FedAT cohesively synthesizes the following modules: (1) a tiering strategy to handle stragglers; (2) an asynchronous scheme to update the global model among tiers for enhanced prediction performance; (3) a novel, weighted aggregation heuristic that the FL server uses to balance the model parameters from heterogeneous, straggling tiers; and (4) a polyline-encoding-based compression algorithm to minimize the communication cost. We have provided rigorous theoretical analysis for our proposed method for two general classes of convex and non-convex losses. We show that FedAT has provable performance guarantee. Our empirical evaluation has validated our theoretical analysis, and demonstrates that FedAT achieves the highest prediction performance, converges the fastest, and is communication-efficient, compared to state-of-the-art FL methods.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283, 2016.

Bernstein, J., Wang, Y.-X., Azizzadenesheli, K., and Anandkumar, A. signsgd: Compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

Chai, Z., Ali, A., Zawad, S., Truex, S., Anwar, A., Baracaldo, N., Zhou, Y., Ludwig, H., Yan, F., and Cheng, Y. Tifl: A tier-based federated learning system. In *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (HPDC)*, pp. 125–136, 2020. ISBN 9781450370523.

Chen, M., Mao, B., and Ma, T. Efficient and robust asynchronous federated learning with stragglers. In *Submitted to International Conference on Learning Representations*, 2019a.

Chen, Y., Ning, Y., and Rangwala, H. Asynchronous online federated learning for edge devices. *arXiv preprint arXiv:1911.02134*, 2019b.

Chen, Y., Sun, X., and Jin, Y. Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation. *IEEE Transactions on Neural Networks and Learning Systems*, 2019c.

Go, A., Bhayani, R., and Huang, L. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009, 2009.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

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.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Koloskova, A., Stich, S. U., and Jaggi, M. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

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., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

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.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2019a.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. On the convergence of fedavg on non-iid data. In *International Conference on Learning Representations*, 2019b.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Mills, J., Hu, J., and Min, G. Communication-efficient federated learning for wireless edge intelligence in iot. *IEEE Internet of Things Journal*, 2019.

Nesterov, Y. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

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

O'herrin, J. K., Fost, N., and Kudsk, K. A. Health insurance portability accountability act (hipaa) regulations: effect on medical record research. *Annals of surgery*, 239(6): 772, 2004.

Reisizadeh, A., Mokhtari, A., Hassani, H., and Pedarsani, R. An exact quantized decentralized gradient descent algorithm. *IEEE Transactions on Signal Processing*, 67 (19):4934–4947, 2019.

Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., and Pedarsani, R. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2021–2031, 2020.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems*, 2019a.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019b.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, pp. 4424–4434, 2017.

Tankard, C. What the gdpr means for businesses. *Network Security*, 2016(6):5–8, 2016.

Wang, J. and Joshi, G. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

Wang, J., Sahu, A. K., Yang, Z., Joshi, G., and Kar, S. Matcha: Speeding up decentralized sgd via matching decomposition sampling. *arXiv preprint arXiv:1905.09435*, 2019.

Woodworth, B. E., Wang, J., Smith, A., McMahan, B., and Srebro, N. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. In *Advances in neural information processing systems*, pp. 8496–8506, 2018.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xie, C., Koyejo, S., and Gupta, I. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

Yu, H., Yang, S., and Zhu, S. Parallel restarted sgd for non-convex optimization with faster convergence and less communication. *arXiv preprint arXiv:1807.06629*, 2(4): 7, 2018.

Zhang, C., Li, S., Xia, J., Wang, W., Yan, F., and Liu, Y. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pp. 493–506. USENIX Association, July 2020. ISBN 978-1-939133-14-4. URL `https://www.usenix.org/conference/atc20/presentation/zhang-chengliang`.

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.

A THEORETICAL ANALYSIS OF FEDAT

We analyze FedAT in the setting of both convex and non-convex situations in this section.

Recall that w^t is the model parameters of the server maintained at the t-th round. Let $\bar{g}_t(w^t) = \sum_{k=1}^{c} \frac{n_k}{N_c} \nabla h_k(w^t)$, in which, N_c is the total number of data samples across all c clients at tier m. Therefore, $w^{t+1} = w^t - \frac{T_{tier_{(M+1-m)}}}{T} \eta \bar{g}_t(w^t)$.

A.1 Proof of Lemma 5.1

To prove Theorem 5.1, we first introduce two Lemmas.

Proof. Using the notion of γ-inexactness for each local objective. We have

$$\nabla h_k(w^t) = F_k(w^t) + \lambda(w^t - w_0) \tag{13}$$

$$||\nabla h_k(w^t)|| \leq \gamma ||\nabla F_k(w^t)|| \tag{14}$$

With $\bar{g}_t(w^t) = \sum_{k=1}^{c} \frac{n_k}{N_c} \nabla h_k(w^t)$, we can get

$$\begin{aligned} ||\bar{g}_t(w^t)||^2 &= \frac{1}{N_c^2} ||n_1 \nabla h_1(w^t) + n_2 \nabla h_2(w^t) +, ..., + n_c \nabla h_c(w^t)||^2 \\ &\leq \frac{1}{N_c^2} \cdot N_c^2 ||\nabla h_1(w^t) + \nabla h_2(w^t) +, ..., + \nabla h_c(w^t)||^2 \\ &\leq m^2 ||\nabla h_{k^*}(w^t)||^2 \quad (k^* = \arg\max_k \nabla h_k(w^t)) \\ &\leq m^2 \gamma^2 ||\nabla F_{k^*}(w^t)||^2 \quad (with\ Eq.(14)) \end{aligned} \tag{15}$$

Take expectation of both sides and with Assumption 5.2, we have

$$\begin{aligned} \mathbb{E}||\bar{g}_t(w^t)||^2 &\leq m^2 \gamma^2 \mathbb{E}||\nabla F_{k^*}(w^t)||^2 \\ &\leq \gamma^2 G^2 c^2 \end{aligned} \tag{16}$$

□

A.2 Proof of Lemma 5.2

Proof. $f(w)$ is *μ-strongly convex*, we can get:

$$f(w') - f(w^t) \geq \langle \nabla f(w^t), w' - w^t \rangle + \frac{\mu}{2} ||w' - w^t||^2, \tag{17}$$

Let us define $\Gamma(w')$ such that:

$$\Gamma(w') = f(w^t) + \langle \nabla f(w^t), w' - w^t \rangle + \frac{\mu}{2} ||w' - w^t||^2, \tag{18}$$

$\Gamma(w')$ is a quadratic function of w', then it has minimal value when $\nabla \Gamma(w') = \nabla f(w^t) + \mu(w' - w^t) = 0$. Then the minimal value of $\Gamma(w')$ is obtained when $w' = w^t - \frac{\nabla f(w^t)}{\mu}$, which is:

$$\Gamma_{\text{min}} = f(w^t) - \frac{||\nabla f(w^t)||^2}{2\mu}, \tag{19}$$

For $f(w)$ is *μ-strongly convex*, we can complete the proof:

$$f(w_*) \geq \Gamma(w_*) \geq \Gamma_{\text{min}} = f(w^t) - \frac{||\nabla f(w^t)||^2}{2\mu}, \tag{20}$$

$$2\mu(f(w^t) - f(w_*)) \leq ||\nabla f(w^t)||^2. \tag{21}$$

□

A.3 Proof of Theorem 5.1

Proof. Now we start to prove the convergence of Theorem 5.1. With Definition 5.1 we can get:

$$\begin{aligned}
&f(w^{t+1}) - f(w^t)\\
&\leq \langle \nabla f(w^t), w^{t+1} - w^t \rangle + \frac{L}{2}||w^{t+1} - w^t||^2 \qquad (f(\cdot) \textit{ is L-smooth})\\
&= -\nabla f(w^t)^\top \frac{T_{tier_{(M+1-m)}}}{T}\eta \bar{g}_t(w^t) + \frac{L\eta^2}{2}\frac{T^2_{tier_{(M+1-m)}}}{T^2}||\bar{g}_t(w^t)||^2 \qquad (w^{t+1} = w^t - \frac{T_{tier_{(M+1-m)}}}{T}\eta \bar{g}_t(w^t))
\end{aligned} \tag{22}$$

Let $B = \frac{T_{tier_{(M+1-m)}}}{T}$. Then with Lemma 5.1, we can update Equation 22 as

$$\begin{aligned}
&\mathbb{E}[f(w^{t+1})] - f(w^t)\\
&\leq -\nabla f(w^t)^\top B\eta \mathbb{E}[\bar{g}_t(w^t)] + \frac{L}{2}\eta^2 B^2 \mathbb{E}||\bar{g}_t(w^t)||^2\\
&\leq -\nabla f(w^t)^\top B\eta \mathbb{E}[\bar{g}_t(w^t)] + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2
\end{aligned} \tag{23}$$

Then from Assumption 5.3, we have

$$\begin{aligned}
&\mathbb{E}[f(w^{t+1})] - f(w^t)\\
&\leq -B\eta\sigma||\nabla f(w^t)||^2 + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2
\end{aligned} \tag{24}$$

Then with Lemma 5.2, Equation (24) can be updated as

$$\begin{aligned}
&\mathbb{E}[f(w^{t+1})] - f(w^t)\\
&\leq -2\mu B\eta\sigma(f(w^t) - f(w_*)) + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2
\end{aligned} \tag{25}$$

By subtracting $f(w_*)$ from both sides and moving $f(w^t)$ from left to right, we get

$$\begin{aligned}
&\mathbb{E}[f(w^{t+1})] - f(w_*)\\
&\leq -2\mu B\eta\sigma(f(w^t) - f(w_*)) + (f(w_t) - f(w_*)) + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2\\
&= (1 - 2\mu B\eta\sigma)(f(w^t) - f(w_*)) + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2
\end{aligned} \tag{26}$$

Taking the whole expectations and rearranging (26), we obtain

$$\begin{aligned}
&\mathbb{E}[f(w^{t+1}) - f(w_*)]\\
&\leq (1 - 2\mu B\eta\sigma)\mathbb{E}[(f(w^t) - f(w_*))] + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2
\end{aligned} \tag{27}$$

subtracting $\frac{L\eta\gamma^2 BG^2 m^2}{4\mu\sigma}$ from both sides, we have

$$\begin{aligned}
&\mathbb{E}[f(w^{t+1}) - f(w_*)] - \frac{L\eta\gamma^2 BG^2 c^2}{4\mu\sigma}\\
&\leq (1 - 2\mu B\eta\sigma)(\mathbb{E}[(f(w^t) - f(w_*))] - \frac{L\eta\gamma^2 BG^2 c^2}{4\mu\sigma})
\end{aligned} \tag{28}$$

The left side of (28) is a geometric series with common ratio $1 - 2\mu B\eta\sigma$, when $t + 1 = T$, we get Equation (11), then we complete the proof. □

A.4 Proof of Theorem 5.2

Proof. Take expectation at both sides of Equation (24), we have

$$\begin{aligned}&\mathbb{E}[f(w^{t+1})] - \mathbb{E}[f(w^t)]\\ &\leq -B\eta\sigma\mathbb{E}[||\nabla f(w_t)||^2] + \frac{L}{2}\eta^2\gamma^2B^2G^2c^2\end{aligned} \tag{29}$$

Then sum Equation (29) at both sides over global iteration T. We have

$$\begin{aligned}&\mathbb{E}[f(w^{t+1})] - f(w^0)\\ &\leq \sum_{t=0}^{T-1} -B\eta\sigma\mathbb{E}[||\nabla f(w_t)||^2] + \frac{L}{2}T^2\eta^2\gamma^2B^2G^2c^2\end{aligned} \tag{30}$$

As $\min f(w^t) = f(w_*) \leq \mathbb{E}[f(w^{t+1})]$, then we have

$$\begin{aligned}f(w_*) \leq f(w^0) - &\sum_{t=0}^{T-1} B\eta\sigma\mathbb{E}[||\nabla f(w_t)||^2]\\ &+ \frac{L}{2}T^2\eta^2\gamma^2B^2G^2c^2\end{aligned} \tag{31}$$

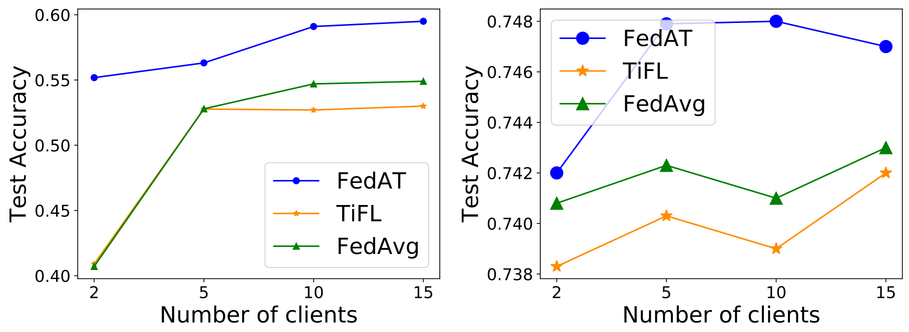
Rearrange (31) we can get

$$\begin{aligned}&\sum_{t=0}^{T-1} B\mathbb{E}[||\nabla f(w_t)||^2]\\ &\leq \frac{f(w^0) - f(w_*)}{B\eta\sigma} + \frac{L}{2\sigma}T^2\eta\gamma^2BG^2c^2\end{aligned} \tag{32}$$

Then we complete the proof. □

B Additional Experimental Results

B.1 Impact of Client Participation Level



(a) CIFAR10 Non-i.i.d.(2) (b) Sentiment140

Figure 7. Prediction accuracy on CIFAR-10 (left) and Sentiment140 (right) as number of client participation increase at each iteration. The comparison is performed among three federated methods which have the synchronous update component.

Figure 7 shows the prediction performance on CIFAR-10 (left) and Sentiment140 (right) trained with different degrees of client participation by FedAvg, TiFL and FedAT , respectively. In real-life situation, it is better to have as fewer clients participate in each global iteration for communication efficiency. However, in Figure 7, we notice that reducing the number of client participation has negative effects on all three federated methods. Partial participation may reduce the convergence speed of FedAT , while with the local constraint term, the optimization is still towarding the optimal solution. In FedAvg, if a nonrepresentative subset of clients is selected then the optimization process could deviate away from the minimum and and might even cause catastrophic forgetting (Goodfellow et al., 2013). Although TiFL possesses the same tiering strategy as FedAT, it has the close performance as FedAvg. As discussed informer sections, TiFL follows the same synchronous update scheme as FedAvg, tiering will affect the convergence speed but not the final prediction accuracy.

We can observe in Figure 7 that FedAT is robust in the non-i.i.d. situation, where the prediction accuracy slightly decreases with the number of client participation decreases. FedAT suffers much less from a reduced participation number than FedAvg and TiFL approaches. Even in the extreme case where only 2 out of 100 clients participate in every round of training, FedATstill achieves much higher accuracy than FedAvg and TiFL.