FedAT: A High-Performance and Communication-Efficient Federated Learning System with Asynchronous Tiers

Zheng Chai
George Mason University
Fairfax, VA, USA
zchai2@gmu.edu

Yujing Chen
George Mason University
Fairfax, VA, USA
ychen37@gmu.edu

Ali Anwar
IBM Research - Almaden
San Jose, CA, USA
ali.anwar2@ibm.com

Liang Zhao
Emory University
Atlanta, GA, USA
liang.zhao@emory.edu

Yue Cheng
George Mason University
Fairfax, VA, USA
yuecheng@gmu.edu

Huzefa Rangwala
George Mason University
Fairfax, VA, USA
rangwala@gmu.edu

Abstract

Federated learning (FL) involves training a model over massive distributed devices, while keeping the training data localized and private. 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 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 single dimension of the tradeoff space. Existing solutions use asynchronous model updating or tiering-based, synchronous mechanisms to tackle the straggler problem. However, asynchronous methods can easily create a communication bottleneck, while tiering may introduce biases that favor faster tiers with shorter response latencies.

To address these issues, we present **FedAT**, a novel **Fed**erated learning system with **A**synchronous **T**iers under Non-i.i.d. training 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 across clients for further accuracy improvement. FedAT compresses uplink and downlink communications using an efficient, polyline-encoding-based compression algorithm, which minimizes 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×, compared to state-of-the-art FL methods.

SC '21, November 14–19, 2021, St. Louis, MO, USA

ACM ISBN 978-1-4503-8442-1/21/11…$15.00
https://doi.org/10.1145/3458817.3476211

CCS Concepts

• **Computing methodologies** → **Multi-agent systems**; **Neural networks**; **Distributed algorithms**; **Cooperation and coordination**.

Keywords

federated learning, asynchronous distributed learning, communication efficiency, tiering, weighted aggregation

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 (clients[1]), as opposed to traditional machine learning training at a centralized location [21, 30]. FL has been used in many application domains, including predicting human activities [9, 10], learning sentiment [41], language processing [15, 25, 49], and enterprise infrastructures [29].

In a typical FL framework, a shared model is learned from a federation of distributed clients with the coordination of a server, and clients do not share data with each other due to security and privacy reasons [35, 43]. Each client trains a local model using its (decentralized) local data.

FL often involves a large number of clients, which feature highly heterogeneous hardware resources (CPU, memory, and network resources) and **Non-i.i.d.** (non-independent and identical) data; that is, the training data distributed across the clients is often non-uniform, since the data generated by a given client is typically based on the usage of that particular edge device and would not be representative of the overall population distribution [26, 30, 39, 53]. The resource and data heterogeneity present 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

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

modes: (1) synchronous communication (e.g., Federated Averaging, or FedAvg [30]), or (2) asynchronous communication (e.g., FedAsync [48]). 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 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 [36, 39], or ii) all the clients are available during the whole training process [34, 53]. However, in practice, clients may come (online) and go (offline) frequently, or lag due to resource/data heterogeneity (i.e., stragglers). Existing synchronous FL solutions (e.g., Federated Averaging, or FedAvg [30]) 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 [9, 48] 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, which updates local model parameters synchronously within tiers and updates 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 the 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; our analysis shows that FedAT has provable convergence guarantee.
- We utilize a lossy compression technique—polyline encoding—to compress the transferred model data between clients and server to reduce the communication cost without affecting the model accuracy.
- We evaluate FedAT extensively on a medium-scale, 100-client cluster on Chameleon Cloud and a large-scale, 500-client cluster on AWS EC2. Experimental results on five federated datasets including CIFAR-10, Fashion-MNIST, Sentiment140, FEMNIST, and Reddit under an FL benchmarking framework **LEAF** [6] 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× compared to FedAsync [48].

2 Related Work

2.1 Stragglers in Federated Learning

The main premise of FL has been collective learning using a network of computing devices such as smartphones and tablets. In such a training environment that cannot be fully controlled, data heterogeneity and resource heterogeneity may cause stragglers [8], which commonly exist in large-scale FL training scenarios [7, 38, 42]. Furthermore, in real-world scenarios, these clients could be frequently offline due to (computing/network) resource constrains. The assumption made by FedAvg that all clients are available during the whole training process is not practical.
Synchronous FL Frameworks. Li et al. [26] suggest to select a smaller ratio of clients for training in each global iteration to alleviate the straggler's effect, while with more rounds for model convergence. However our experiments in §7.5.1 show that selecting less number of clients for each round produces lower performance. Bonawitz et al. [4] proposes a naive clients selection strategy to mitigate stragglers, where 130% target number of clients are selected for each round. With this approach, the slowest 30% is neglected. However, it comes with more communication cost and potential failure in handling stragglers when the number of stragglers involved in some rounds exceed the 30% tolerance. FedProx [24] tackles system heterogeneity by using distinct local epoch numbers for clients. However, choosing a perfect local epoch number for each client is challenging in real-world applications.

TiFL [7] is a tier based FL framework that uses a synchronous, intra-tier model updating scheme similar to that used in FedAvg. The adaptive tier selection algorithm that TiFL relies on requires collecting test accuracies of all clients every certain rounds. This means higher communication costs and longer training duration. Our experiments results in §7 show that this algorithm fails to

achieve efficient communication given a high level of Non-i.i.d.-ness with larger number of clients and may result in biased training and lower accuracy. In real-world scenarios, however, it is possible that a portion of clients are incorrectly profiled and assigned to a wrong tier as clients' response latencies may largely vary from time to time. FedAT uses TiFL's tiering approach but differs from TiFL in that FedAT combines intra-tier synchronous training with cross-tier asynchronous training to effectively mitigate training biases. With our new asynchronous mechanism, FedAT can tolerate mis-tiering caused by mis-profiling and performance variation.

Asynchronous FL Frameworks. Asynchronism is widely used in distributed systems for shortening overall running time [11, 14, 27, 32]. Asynchronous FL frameworks [9, 28, 48] allow wait-free communication and computation in order to address the straggler problem. These asynchronous approaches, however, suffer from high communication costs as they require much more frequent communications between clients and the server. Worse, frequent aggregation on the server side may lead to slow convergence speed.

2.2 Communication-Efficient Federated Optimization

McMahan et al. propose a FL approach called FedAvg [30], 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. [21] propose two approaches to reduce the uplink communication costs, i.e., structured updates and sketched updates, combined with probabilistic quantization. These two approaches, however, are only suitable for i.i.d. settings in FL. For Non-i.i.d. settings, they can significantly slow 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. [10] 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. [31] adapt FedAvg to use a distributed form of Adam optimization and compress the uploaded parameters. Jeong et al. [17] develop federated distillation that follows an online version of knowledge distillation to compress the model. Reisizadeh et al. [36] 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-world scenario where stragglers are common. In addition to the above mentioned server-client topology, solving communication bottlenecks via quantization and compression has also gained considerable attention in decentralized training [20, 37, 45]. While such techniques can be used to reduce communication costs in FL, a decentralized network topology in distributed learning without a server is fundamentally different and is thus orthogonal to our approach.

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 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 $F_k(w) \stackrel{def}{=} \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell_i(x_i, y_i; w)$, 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:

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

Algorithm 1: Federated Averaging Training Algorithm

Server: Initialize global weights w^0
for *each round* $t = 0$ *to* $T - 1$ **do**
  $\mathcal{S}$ = (random set of clients)
  for *each client* $k \in \mathcal{S}$ ***in parallel*** **do**
    $w_k^{t+1} = w_k^t - \eta \nabla h(w^t)$
  $N = \sum_{k=1}^{|\mathcal{S}|} n_k$
  $w^{t+1} = \sum_{k=1}^{|\mathcal{S}|} \frac{n_k}{N} \cdot w_k^{t+1}$

FedAvg [30] 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). The detailed process of FedAvg is shown in Algorithm 1. This kind of local update methods enable flexible and efficient communication compared to traditional mini-batch methods [44, 46, 50].

In a typical real-world scenario, the data stored across devices follow a non-i.i.d. distribution. 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 converging to a global optimal solution. Therefore, solving

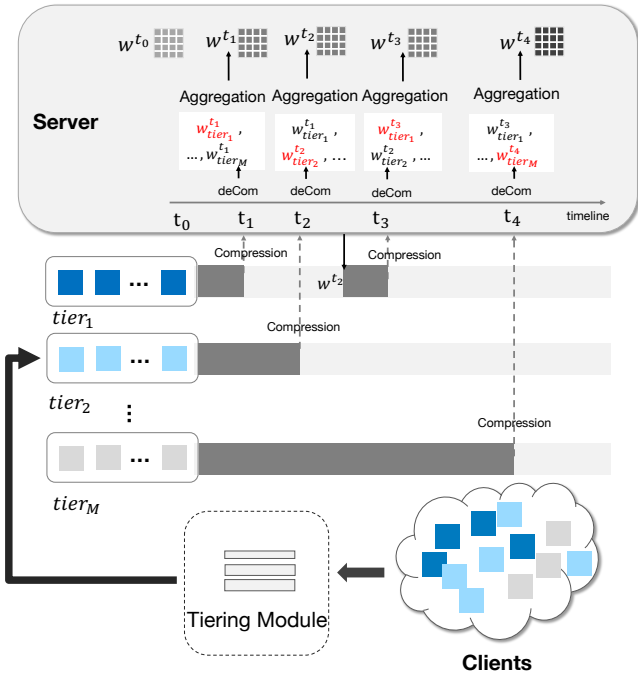


Figure 1: Overview of FedAT. $tier_1, ..., tier_M$ **are** M **tiers, and** $w^t_{tier_1}, ..., w^t_{tier_M}$ **are their according weights, respectively.** `deCom` **denotes the decompression process of clients' models in a certain tier on the server.**

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 causes 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 [7, 9, 10, 17, 48].

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 2). 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 ($\mathcal{S}$) 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 §4.2), 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 Training at Local Clients

As mentioned in §3, 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 add a constraint term to the local subproblem by restricting the local updates to be closer to the global model.

Thus, following [24, 52], instead of just minimizing the local function F_k , client k applies an update with the constraint using the following surrogate objective h_k:

$$h_k(w_k) = F_k(w_k) + \frac{\lambda}{2}||w_k - w||^2, \tag{3}$$

where w_k, w are the local model of client k and server model, respectively.

We use $f_{tier_m}(w)$ as 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 follows:

$$\begin{aligned} f_{tier_m}(w) &= \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} h_k(w_k) \\ &= \sum_{k=1}^{|\mathcal{S}_t|} \frac{n_k}{N_c} (F_k(w_k) + \frac{\lambda}{2}||w_k - w||^2). \end{aligned} \tag{4}$$

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.

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.

4.2 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:

$$f(w) = \sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} f_{tier_m}(w), \tag{5}$$

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. The approach of FedAT is detailed in Algorithm 2.

Algorithm 2: 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^t = **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**
      $n_k = |\mathcal{D}_k|$
      $w_k^{t+1} = w_k^t - \eta \nabla h(w^t)$
    $N_c = \sum_{k=1}^{|\mathcal{S}_m|} n_k$
    $w_{tier_m} = \sum_{k=1}^{|\mathcal{S}_m|} \frac{n_k}{N_c} \cdot w_k^{t+1}$
    $T_{tier_m} = T_{tier_m} + 1$
    $t = t + 1$

function WeightedAverage$(w_{tier_1}, w_{tier_2} ... w_{tier_M})$
if $t == 0$ **then**
  return w^{t_0}
else
  return $\sum_{m=1}^{M} \frac{T_{tier_{(M+1-m)}}}{T} \cdot w_{tier_m}$

4.3 Compression, Marshalling, and Unmarshalling

Previous work 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 [30]. As studied in [39], many compression methods [3, 40] 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, some lossy compression methods such as quantization and dequantization [51] may inevitably lead to huge errors and reduce global performance. 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. With the appropriate precision, the model could achieve the largest communication saving and minor performance loss. Our experiments show that it can achieve a high compression

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

ratio of up to 3.5× under the FL communication scenarios. (We evaluate the effectiveness of compression in §7.2.) 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 in theory. It is consistent with our experiments results shown in Figure 3. First, we introduce the definitions and assumptions as follows.

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

$$f(x_1) - f(x_2) \leq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{L}{2}||x_1 - x_2||^2. \tag{6}$$

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

$$f(x_1) - f(x_2) \geq \langle \nabla f(x_2), x_1 - x_2 \rangle + \frac{\mu}{2}||x_1 - x_2||^2. \tag{7}$$

Definition 5.3 has been made by the work [24].

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 [23], 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 a 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*

$$\mathbb{E}||\bar{g}_t(w^t)||^2 \leq \gamma^2 G^2 c^2, \tag{8}$$

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:*

$$2\mu(f(w^t) - f(w_*)) \leq ||\nabla f(w^t)||^2. \tag{9}$$

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 [5, 33], 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_*:*

$$\begin{aligned} &\mathbb{E}[f(w^T) - f(w_*)] \\ &= (1 - 2\mu B\eta\sigma)^T (f(w^0) - f(w_*)) + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2, \end{aligned} \tag{10}$$

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 the 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.

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:*

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

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

6 Experimental Setting

Federated Datasets. We evaluate FedAT using five different federated datasets and an FL benchmarking framework **LEAF** [6] on both convex and non-convex models described as follows:

- CIFAR-10: The CIFAR-10 [22] dataset consists of 60,000 32×32 colour images in 10 classes, with 6000 images per class. There are 50,000 training images and 10,000 test images. We partition

the dataset into 100 clients and follow the same Non-i.i.d. setting of CIFAR-10 as [30].
- Fashion-MNIST: Fashion-MNIST [47] is a dataset that contains a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28×28 grayscale image, associated with a label from 10 classes. We use the same number of clients and Non-i.i.d. setting as CIFAR-10.
- Sentiment140: Sentiment140 [12] is a text dataset that contains 1,600,000 tweets, and the tweets are labeled with one of negative, neutral and positive. Each twitter account corresponds to a client.
- FEMNIST: FEMNIST [6] is an image dataset that consists of 805,263 samples. There are a total of 62 classes, where all the samples are distributed to 3,550 clients with inherent data heterogeneity and Non-i.i.d.-ness.
- Reddit: Reddit [6] is a text dataset that contains processed comments posted on Reddit in December 2017. It has a total of 56,587,343 samples collected from 1,660,820 Reddit users.

FL Methods. We compare FedAT against five synchronous and asynchronous FL methods:
- FedAvg [30]: A baseline synchronous FL method proposed by McMahan et al. At each round, a certain ratio of total clients are randomly selected for training, the server aggregates the weights received from selected clients in an average manner.
- TiFL [7]: A synchronous FL method that partitions training clients into different tiers based on their responding latency. For each round, one tier is selected according to a novel adaptive selection strategy which is related to the test accuracies of all tiers, then certain number of clients in that tier are selected for training. The aggregation method of TiFL is adopted from FedAvg.
- FedProx [24]: A synchronous FL method that claims to handle stragglers due to system heterogeneity across all clients by applying different local epochs for clients. Also FedProx adds proximal term to the objective on the clients for improving performance and smoothing the training curve.
- FedAsync [48]: A baseline asynchronous FL method using weighted averaging to update the server model. Different from previous synchronous FL methods, all clients train concurrently, when the server receives weights from any client, the weights are weighted averaged with current global weights to get the latest global weights, then communicate to all available clients at that time for training.
- ASO-Fed [9]: An asynchronous FL framework designed for online data. Unlike FedAsync, ASO-Fed adds local constraints on the client side. ASO-Fed maintains a copy of weights for each client on the server side, and it calculates global weights by averaging all the copies. Note that we compare ASO-Fed against FedAT on a large-scale, 500-client cluster on FEMNIST (§7.4).

7 Evaluation

Implementation and Setup. We have implemented FedAT and the comparison FL methods all in TensorFlow [2]. For all the experiments except FEMNIST and Reddit, we simulate a FL setup using a 192-core cluster on Chameleon Cloud [1, 18], which consists of three bare-metal servers, each with a 64-core Intel® Xeon® Gold 6242 CPU, and 192 GB main memory; we deploy the FL server exclusively on one server, and all clients on the other two servers, where each client gets assigned one CPU core. We evaluate 100 clients in Chameleon Cloud tests.

For experiments on FEMNIST and Reddit, we deploy the FL server exclusively on one *c5.24xlarge* virtual machine (VM) instance (96 vCPUs and 192 GB memory) on AWS and 500 participating clients on a large-scale, 100-VM cluster, where each VM is a *c5.2xlarge* instance with 8 vCPUs and 16 GB memory.

FedAT is configured to use `Precision 4` as the precision of the compressor (§7.2.2) throughout the evaluation for CIFAR-10, Fashion-MNIST and Sentiment140.

Models. We train a convolutional neural network (CNN) on CIFAR-10 and Fashion-MNIST. 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. For Sentiment140, we train a logistic regression model to evaluate the model performance under a convex setting. For the FEMNIST dataset, we train a similar CNN that classifies images. For the Reddit dataset, we train an LSTM model [16]. The LSTM model starts with an embedding layer with an input dimension of 10,000 and an output dimension of 128, followed by an LSTM layer with a dropout rate of 0.1, a batch normalization layer and a dense layer with 10,000 units.

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 [19] as the local solver and set the local constrain parameter λ as 0.4. For each dataset, we tune the learning rate for FedAvg using the following configuration: *local epoch* $E = 3$, *batch size* $= 10$; we use the same learning rate for the other five FL methods. We set the number of randomly selected clients as 10 for FedAvg, TiFL, FedProx, and FedAT on all datasets.

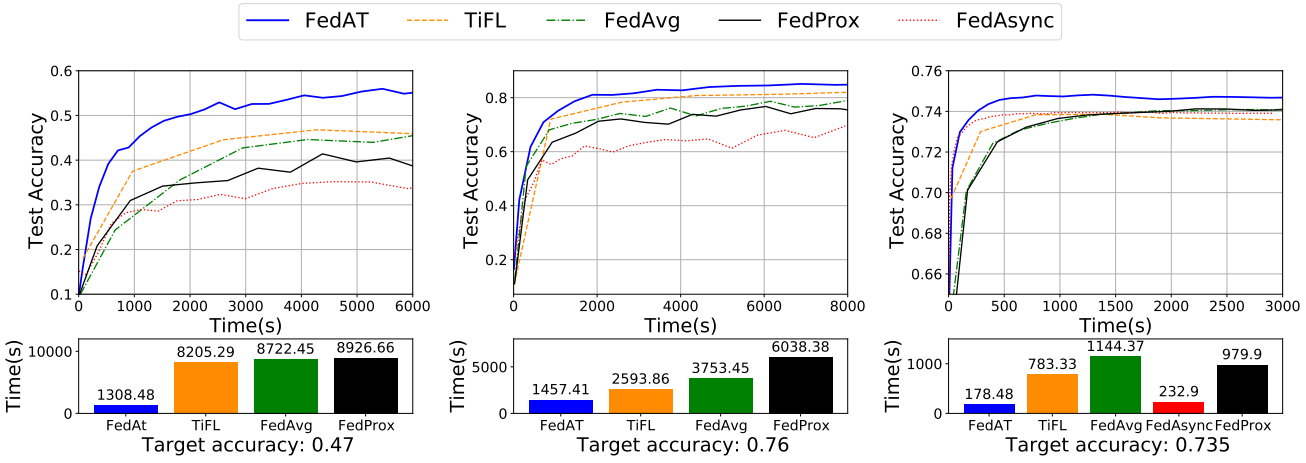
Simulating Different Performance Tiers. Clients in FL are typically edge devices, and their computing power and network connection may not be stable; hence simply assigning a fixed amount of resources is not sufficient to reflect the real situation. Therefore, we assign 1 CPU for each client during the whole training process and add random delays to the computations conducted by clients; the added random delay is to simulate different levels of straggler effects that are 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.

7.1 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

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` rows show the best prediction accuracy that each FL approach reaches after each model converges. The `Norm.Var.` rows 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)		CIFAR-10					Fashion-MNIST	Sentiment140
		#2	#4	#6	#8	i.i.d.	#2	
TiFL	Accuracy	0.527	0.615	0.654	0.655	0.685	0.859	0.739
	Norm. Var.	1.26	2.79	1.33	1.3	2.12	1.29	2.75
FedAvg	Accuracy	0.547	0.628	0.654	0.667	0.686	0.842	0.741
	Norm. Var.	2	5.07	4.33	3.1	4.23	1.86	3.72
FedProx	Accuracy	0.509	0.609	0.624	0.650	0.669	0.831	0.742
	Norm. Var.	1.261	6.75	3.981	2.22	2.992	2.243	3.89
FedAsync	Accuracy	0.480	0.541	0.531	0.561	0.567	0.795	0.740
	Norm. Var.	2	3.93	2.08	1.54	2.69	2	5.69
FedAT	Accuracy	**0.591**	**0.633**	**0.673**	**0.681**	**0.701**	**0.873**	**0.748**
	Abs. Var.	**0.0042**	**0.0014**	**0.0012**	**0.001**	**0.00052**	**0.007**	**2.67e^{-5}**
	impr.(a)	7.44%	0.79%	2.82%	2.05%	2.13%	1.6%	0.93%
	impr.(b)	18.78%	14.53%	21.09%	17.62%	19.11%	8.93%	1.2%

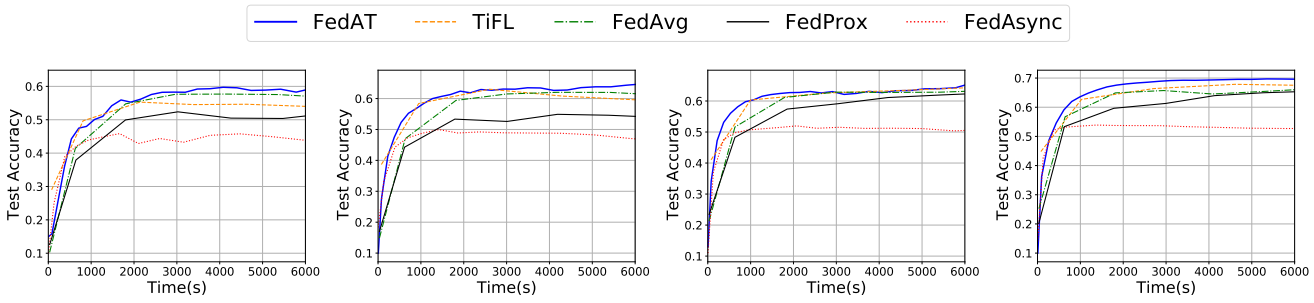


(a) CNN @ CIFAR-10. (b) CNN @ Fashion-MNIST. (c) Logistic @ Sentiment140.

Figure 2: Performance comparison of different FL methods on 2-class Non-i.i.d. CIFAR-10, 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 CIFAR-10 and Fashion-MNIST, thus is omitted.)

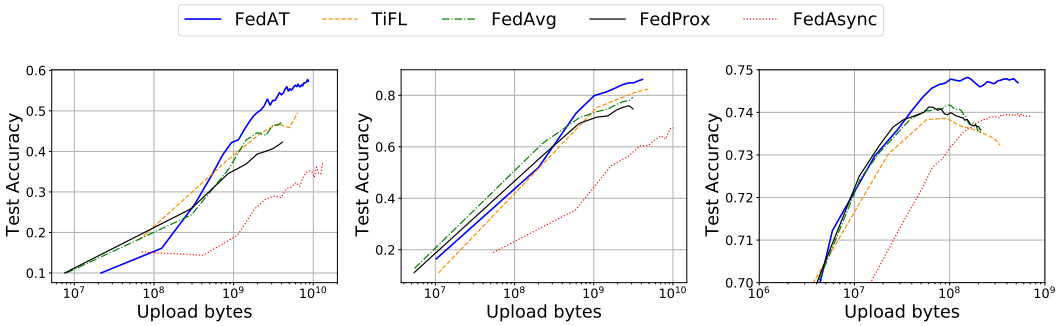
best test accuracy after each training process converges within a global iteration budget. For the 2-class CIFAR-10 dataset, FedAT outperforms the best baseline FL method, FedAvg, by 7.44%, and the worst baseline method, FedAsync, by 18.78%. Using the same tiering scheme as TiFL, FedAT achieves consistently 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 §7.3). 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 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.

7.1.1 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 CIFAR-10 dataset, the test



(a) CIFAR-10 Non-i.i.d. (4). (b) CIFAR-10 Non-i.i.d. (6). (c) CIFAR-10 Non-i.i.d. (8). (d) CIFAR-10 i.i.d.

Figure 3: Convergence speed comparison on CIFAR-10 over different level of Non-i.i.d.-ness. The results are average-smoothed for every 40 global rounds.



(a) CIFAR-10. (b) Fashion-MNIST. (c) Sentiment140.

Figure 4: Test accuracy as a function of the cumulative amounts of data uploaded from 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.

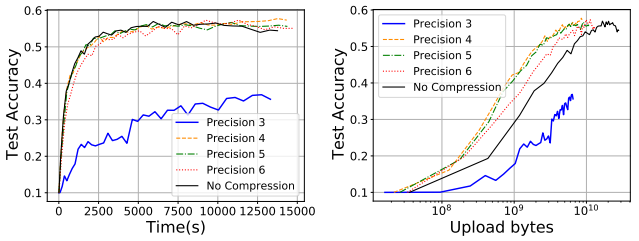
Table 2: Amounts of data (MB) transferred between clients and server to achieve the target accuracy on all datasets with 2-class Non-i.i.d. case. – means that the FL method is not able to achieve the accuracy target within the iteration budget. The best results are highlighted in bold font.

Method	CIFAR-10 (acc. = 0.50)	Fashion-MNIST (acc. = 0.79)	Sentiment140 (acc. = 0.73)
FedAvg	1828.54	1048.25	16.71
TiFL	2140.71	1041.98	17.20
FedProx	–	2169.95	18.42
FedAsync	–	9895.53	82.27
FedAT	**1675.82**	**1041.54**	**16.41**

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 CIFAR-10 dataset. We observe that FedAT outperforms all the other four 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.

7.1.2 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× 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 CIFAR-10 CNN model, TiFL, FedAvg and FedProx spend 5.27×, 5.67× and 5.82× longer time than FedAT. Fashion-MNIST show a similar trend. For Sentiment140, TiFL,



(a) Test accuracy. (b) Uploaded data vs. accuracy (the X-axis is in log-scale).

Figure 5: Impact of FedAT's compression precision on the prediction performance and the communication cost, for the CIFAR-10 Non-i.i.d. 2-class dataset. All results are plotted with the average of every 40 global rounds.

FedAvg, FedProx and FedAsync take 3.39×, 5.41×, 4.49× and 0.3× longer time than FedAT, respectively.

7.2 Communication Efficiency

7.2.1 Communication Cost. We next evaluate the network communication cost of FedAT in terms of the amount 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× of FedAT, and 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.

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 CIFAR-10 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.

7.2.2 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.

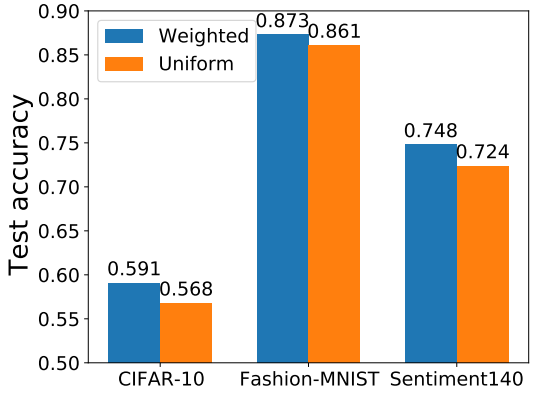
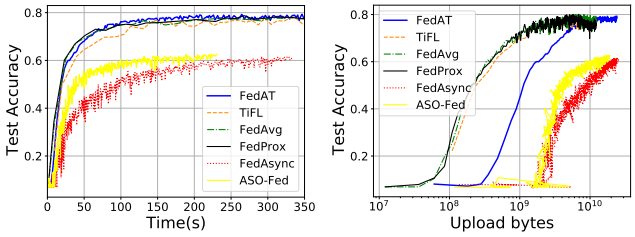


Figure 6: Comparison of FedAT's weighted aggregation heuristic vs. a uniform baseline that assigns uniform weights when aggregating models from different tiers.



(a) Accuracy over time. (b) Accuracy over uploaded bytes.

Figure 7: Prediction accuracy of FEMNIST as a function of training time (a) and cumulative amount of data uploaded from clients to the server (b).

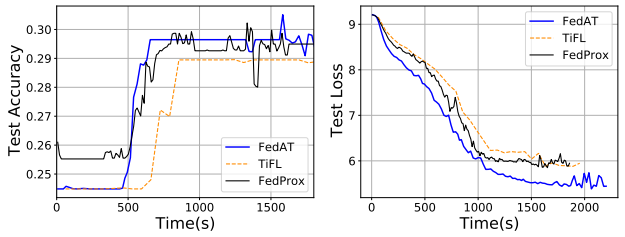
7.3 Effectiveness of Weighted Aggregation

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.

7.4 Large-Scale Training

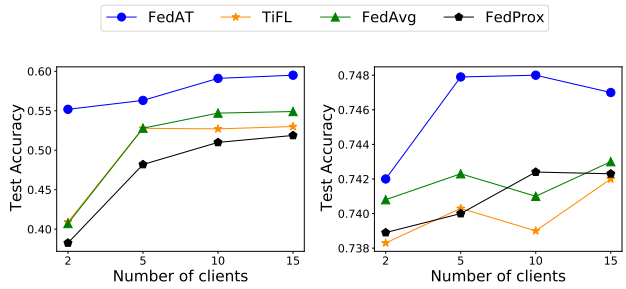
In this test, we conduct large-scale experiments on the FEMNIST and Reddit datasets with 500 participating clients deployed on 100 *c5.2xlarge* AWS EC2 VMs.

As shown in Figure 7(a), FedAT achieves the highest accuracy at the early stage of the training process, while maintaining at least 1.2% higher accuracy than state-of-the-art synchronous methods, FedProx and TiFL. The two asynchronous FL methods, FedAsync and ASO-Fed, still perform worse than other synchronous methods.



(a) Accuracy over time. (b) Loss over time.

Figure 8: Prediction accuracy (a) and loss (b) of Reddit as a function of training time.



(a) CIFAR-10 Non-i.i.d.(2). (b) Sentiment140.

Figure 9: Prediction accuracy of CIFAR-10 (left) and Sentiment140 (right) as a function of the number of participating clients in one iteration. This test compares FedAT against three other FL methods (FedAvg, TiFL, FedProx), which all feature synchronous updating.

In addition, FedAsync and ASO-Fed see much higher communication cost than that of FedAT.

Although FedAT incurs higher communication cost than the synchronous FL methods at the early training stage due to asynchronous, cross-tier training, FedAT eventually achieves similar communication efficiency as the synchronous methods when these synchronous methods reach the highest prediction accuracy. This is because, with more frequent model update between tiers and the server, FedAT converges faster than synchronous baselines.

Figure 8 shows the prediction accuracy and loss on the Reddit dataset. Asynchronous FL baselines (FedAsync and ASO-Fed) have much lower prediction performance with no convergence trend on the Reddit dataset, therefore we omit their results in this test. We compare FedAT with TiFL and FedProx, which perform the best among all baseline FL methods. As shown in Figure 8, we observe similar learning trend for the three frameworks, but FedAT has better prediction performance. Figure 8(b) shows that FedAT achieves the lowest loss during the whole training process.

7.5 Sensitivity Analysis

7.5.1 Impact of Client Participation Level. We next conduct a sensitivity study to quantify the impact of client participation level on the training accuracy. In a real-world situation, for the communication efficiency consideration, it is often desirable to have as few clients as possible that participate in each global iteration. In

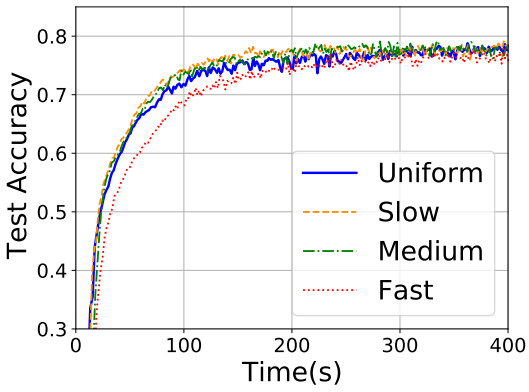


Figure 10: Comparison of prediction accuracy over time on FEMNIST under different configurations of client distribution across tiers.

FedAvg, if a non-representative subset of clients is selected, the optimization process can deviate away from the minimum, which might lead to catastrophic forgetting [13]. Although TiFL adopts the same tiering strategy as FedAT, TiFL achieves a similar performance as FedAvg. This is because, as discussed in §2.1, TiFL uses the same synchronous update scheme as FedAvg, and tiering inevitably slows down the convergence speed but may not affect the final prediction accuracy when the training eventually converges.

We observe in Figure 9 that reducing the level of client participation has negative effect on all of the four FL methods. FedAT is robust in the non-i.i.d. case, where the prediction accuracy slightly decreases when the number of client participation decreases. While partial participation may reduce the convergence speed of FedAT , the optimization can still achieve an optimal solution with the local constraint term. FedAT suffers much less from a reduced participation level than FedAvg and TiFL. Even in the extreme case where only 2 out of 100 clients participate in each round of training, FedAT still achieves 14.47%, 14.28% and 16.93% higher accuracy than FedAvg, TiFL, and FedProx on CIFAR-10, respectively. This is because the asynchronous, cross-tier training allows more clients to contribute to the global model, thus increasing the test accuracy on all clients.

7.5.2 Impact of Number of Clients in Tiers. We next evaluate the robustness and resilience of FedAT to changing number of clients in different tiers. We partition a total of 500 clients to five performance tiers, where each tier either has good resources or less amount of training data. Specially, we test the following four configurations: *Uniform*: the baseline configuration that assigns the same number of clients to each tier with a distribution of 100/100/100/100/100. *Slow*: where the slowest tiers (i.e., Tier 5) has the largest number of clients with a distribution of 50/50/100/100/200. *Medium*: where the medium tier (i.e., Tier 3) gets the largest number of clients with a distribution of 50/100/200/100/50. *Fast*: where the fastest tier (i.e., Tier 1) has the largest number of clients with a distribution of 200/100/100/50/50. As shown in Figure 10, all the four partition configurations eventually converge with close prediction performance, thanks to FedAT's hybrid, synchronous, intra-tier training and asynchronous, cross-tier training strategy. *Slow* and *Medium*

converges slightly faster than *Fast*, since clients in *Fast* configuration either have good resources or less amount of data. As a result, training for the same number of rounds yield less accurate model in *Fast* configuration as we are training on overall less amount of data. The results indicate that varying the tier sizes would affect the convergence speed marginally but would not impact the model performance eventually when the training converges.

8 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 mechanism. 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 model performance guarantee. Our evaluation has empirically 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.

Acknowledgments

We are grateful to the anonymous reviewers for their valuable feedback and suggestions that improved the paper. This work is sponsored in part by the National Science Foundation (NSF) under CCF-1919075, CCF-1919113, CMMI-2134689, IIS-1755850, CNS-1841520, IIS-2007716, OAC-2007976, IIS-1942594, IIS-1907805, a Jeffress Memorial Trust Award, Amazon Research Award, NVIDIA GPU Grant, and Design Knowledge Company (subcontract number: 10827.002.120.04). Part of the results presented in this paper were obtained using the Chameleon testbed supported by NSF.

References

[1] Chameleon Cloud: A configurable experimental environment for large-scale cloud research. https://www.chameleoncloud.org/.

[2] 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)* (2016), pp. 265–283.

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

[4] Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H. B., et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

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

[6] Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097* (2018).

[7] 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)* (2020), p. 125–136.

[8] Chai, Z., Fayyaz, H., Fayyaz, Z., Anwar, A., Zhou, Y., Baracaldo, N., Ludwig, H., and Cheng, Y. Towards taming the resource and data heterogeneity in federated learning. In *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)* (2019), pp. 19–21.

[9] Chen, Y., Ning, Y., and Rangwala, H. Asynchronous online federated learning for edge devices. *arXiv preprint arXiv:1911.02134* (2019).

[10] 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* (2019).

[11] Fan, W., Lu, P., Yu, W., Xu, J., Yin, Q., Luo, X., Zhou, J., and Jin, R. Adaptive asynchronous parallelization of graph algorithms. *ACM Transactions on Database Systems (TODS) 45*, 2 (2020), 1–45.

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

[13] 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).

[14] Han, M., and Daudjee, K. Giraph unchained: Barrierless asynchronous parallel execution in pregel-like graph processing systems. *Proceedings of the VLDB Endowment 8*, 9 (2015), 950–961.

[15] Hard, A., Rao, K., Mathews, R., Ramaswamy, S., Beaufays, F., Augenstein, S., Eichner, H., Kiddon, C., and Ramage, D. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* (2018).

[16] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural Comput. 9*, 8 (Nov. 1997), 1735–1780.

[17] 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).

[18] Keahey, K., Anderson, J., Zhen, Z., Riteau, P., Ruth, P., Stanzione, D., Cevik, M., Colleran, J., Gunawi, H. S., Hammock, C., Mambretti, J., Barnes, A., Halbach, F., Rocha, A., and Stubbs, J. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.

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

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

[21] 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).

[22] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images.

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

[24] 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).

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

[26] 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* (2019).

[27] Lian, X., Zhang, W., Zhang, C., and Liu, J. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning* (2018), PMLR, pp. 3043–3052.

[28] Lu, Y., Huang, X., Dai, Y., Maharjan, S., and Zhang, Y. Differentially private asynchronous federated learning for mobile edge computing in urban informatics. *IEEE Transactions on Industrial Informatics 16*, 3 (2019), 2134–2143.

[29] Ludwig, H., Baracaldo, N., Thomas, G., Zhou, Y., Anwar, A., Rajamoni, S., Ong, Y., Radhakrishnan, J., Verma, A., Sinn, M., et al. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987* (2020).

[30] 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* (2017), pp. 1273–1282.

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

[32] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (2016), pp. 1928–1937.

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

[34] 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)* (2019), IEEE, pp. 1–7.

[35] 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 (2004), 772.

[36] 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* (2020), pp. 2021–2031.

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

[38] Reisizadeh, A., Tziotis, I., Hassani, H., Mokhtari, A., and Pedarsani, R. Straggler-resilient federated learning: Leveraging the interplay between statistical accuracy and system heterogeneity, 2020.

[39] 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* (2019).

[40] 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)* (2019), IEEE.

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

[42] Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[43] Tankard, C. What the gdpr means for businesses. *Network Security 2016*.

[44] 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).

[45] 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).

[46] 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* (2018), pp. 8496–8506.

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

[48] Xie, C., Koyejo, S., and Gupta, I. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934* (2019).

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

[50] 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 (2018), 7.

[51] 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)* (July 2020), USENIX Association.

[52] Zhang, S., Choromanska, A., and LeCun, Y. Deep learning with elastic averaging sgd. *arXiv preprint arXiv:1412.6651* (2014).

[53] 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).

Appendix

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

$$\nabla h_k(w^t) = F_k(w^t) + \lambda(w^t - w_0) \tag{12}$$

$$||\nabla h_k(w^t)|| \leq \gamma ||\nabla F_k(w^t)|| \tag{13}$$

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

$$\begin{aligned} ||\bar{g}_t(w^t)||^2 &= \frac{1}{N_c^2} ||n_1 \nabla h_1(w^t) + n_2 \nabla h_2(w^t) +, ..., + n_c \nabla h_c(w^t)||^2 \\ &\leq \frac{1}{N_c^2} \cdot N_c^2 ||\nabla h_1(w^t) + \nabla h_2(w^t) +, ..., + \nabla h_c(w^t)||^2 \\ &\leq m^2 ||\nabla h_{k^*}(w^t)||^2 \quad (k^* = \arg\max_k \nabla h_k(w^t)) \\ &\leq m^2 \gamma^2 ||\nabla F_{k^*}(w^t)||^2 \quad (with\ Eq.(13)) \end{aligned} \tag{14}$$

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

$$\begin{aligned} \mathbb{E}||\bar{g}_t(w^t)||^2 &\leq m^2 \gamma^2 \mathbb{E}||\nabla F_{k^*}(w^t)||^2 \\ &\leq \gamma^2 G^2 c^2 \end{aligned} \tag{15}$$

□

A.2 Proof of Lemma 5.2

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

$$f(w') - f(w^t) \geq \langle \nabla f(w^t), w' - w^t \rangle + \frac{\mu}{2} ||w' - w^t||^2, \tag{16}$$

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

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

$\Gamma(w')$ 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:

$$\Gamma_{\min} = f(w^t) - \frac{||\nabla f(w^t)||^2}{2\mu}, \tag{18}$$

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

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

$$2\mu(f(w^t) - f(w_*)) \leq ||\nabla f(w^t)||^2. \tag{20}$$

□

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:

$$\begin{aligned} & f(w^{t+1}) - f(w^t) \\ & \leq \langle \nabla f(w^t), w^{t+1} - w^t \rangle + \frac{L}{2}||w^{t+1} - w^t||^2 \qquad (f(\cdot) \text{ is } L\text{-smooth}) \\ & = -\nabla f(w^t)^\top \frac{T_{tier_{(M+1-m)}}}{T} \eta \bar{g}_t(w^t) + \frac{L\eta^2}{2} \frac{T^2_{tier_{(M+1-m)}}}{T^2} ||\bar{g}_t(w^t)||^2 \qquad (w^{t+1} = w^t - \frac{T_{tier_{(M+1-m)}}}{T} \eta \bar{g}_t(w^t)) \end{aligned} \tag{21}$$

Let $B = \frac{T_{tier_{(M+1-m)}}}{T}$. Then with Lemma 5.1, we can update Equation 21 as

$$\begin{aligned} & \mathbb{E}[f(w^{t+1})] - f(w^t) \\ & \leq -\nabla f(w^t)^\top B\eta \mathbb{E}[\bar{g}_t(w^t)] + \frac{L}{2}\eta^2 B^2 \mathbb{E}||\bar{g}_t(w^t)||^2 \\ & \leq -\nabla f(w^t)^\top B\eta \mathbb{E}[\bar{g}_t(w^t)] + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2 \end{aligned} \tag{22}$$

Then from Assumption 5.3, we have

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

Then with Lemma 5.2, Equation (23) can be updated as

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

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

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

Taking the whole expectations and rearranging (25), we obtain

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

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

$$\begin{aligned} & \mathbb{E}[f(w^{t+1}) - f(w_*)] - \frac{L\eta\gamma^2 BG^2 c^2}{4\mu\sigma} \\ & \leq (1 - 2\mu B\eta\sigma)(\mathbb{E}[(f(w^t) - f(w_*))] - \frac{L\eta\gamma^2 BG^2 c^2}{4\mu\sigma}) \end{aligned} \tag{27}$$

The left side of (27) is a geometric series with common ratio $1 - 2\mu B\eta\sigma$, when $t + 1 = T$, we get Equation (10), then we complete the proof. □

A.4 Proof of Theorem 5.2

Proof. Take expectation at both sides of Equation (23), we have

$$\begin{aligned} & \mathbb{E}[f(w^{t+1})] - \mathbb{E}[f(w^t)] \\ & \leq -B\eta\sigma\mathbb{E}[||\nabla f(w_t)||^2] + \frac{L}{2}\eta^2\gamma^2 B^2 G^2 c^2 \end{aligned} \tag{28}$$

Then sum Equation (28) at both sides over global iteration T. We have

$$\begin{aligned} & \mathbb{E}[f(w^{t+1})] - f(w^0) \\ & \leq \sum_{t=0}^{T-1} -B\eta\sigma\mathbb{E}[||\nabla f(w_t)||^2] + \frac{L}{2}T^2\eta^2\gamma^2 B^2 G^2 c^2 \end{aligned} \tag{29}$$

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

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

Rearrange (30) we can get

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

Then we complete the proof. □