GlueFL: Reconciling Client Sampling and Model Masking for Bandwidth Efficient Federated Learning

by

Shiqi He

B.Sc., Hong Kong Polytechnic University, 2020

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

January 2023

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

GlueFL: Reconciling Client Sampling and Model Masking for Bandwidth Efficient Federated Learning

submitted by **Shiqi He** in partial fulfillment of the requirements for the degree of **Master of Science** in **Computer Science**.

Examining Committee:

Ivan Beschastnikh, Associate Professor, Computer Science, UBC
Supervisor

Mathias Lécuyer, Assistant Professor, Computer Science, UBC
Supervisory Committee Member

Abstract

Federated learning (FL) is an effective technique to directly involve edge devices in machine learning (ML) training while preserving client privacy. However, the substantial communication overhead of FL makes training challenging when edge devices have limited network bandwidth. Existing work to optimize FL bandwidth overlooks downstream transmission and does not account for FL client sampling.

We propose *GlueFL*, a framework that incorporates new client sampling and model compression algorithms to mitigate low download bandwidths of FL clients. GlueFL prioritizes recently used clients and bounds the number of changed positions in compression masks in each round.

We analyse FL convergence under GlueFL's sticky sampling, and show that our proposed weighted aggregation preserves unbiasedness of updates and convergence.

We evaluate GlueFL empirically, and demonstrate downstream bandwidth and training time savings on three public datasets. On average, our evaluation shows that GlueFL spends 29% less training time with a 27% less downstream bandwidth overhead as compared to three state-of-the-art strategies.

Lay Summary

Machine learning is known as an effective approach to solving predictive tasks by leveraging large volumes of data. Federated learning moves machine learning training to mobile devices and allows participants to collaboratively train a global model without disclosing their local training data. However, since the training may involve thousands to millions of participants, network usage becomes a performance bottleneck. Participants that have low bandwidth act as stragglers and slow down model training.

We design GlueFL, a federated learning training framework that optimizes bandwidth usage. It consists of novel client sampling and model compression algorithms. These new mechanisms alleviate the impact of client staleness in client sampling and minimize downstream bandwidth. We empirically demonstrate that GlueFL decreases communication costs while preserving model performance on three public datasets. To the best of our knowledge, this is the first work to combine model masking with client sampling to reduce downstream bandwidth.

Preface

This thesis presents the original and unpublished work done by the author, Shiqi He, conducted in the Systopia lab at the Univerisity of British Columbia under the supervision of Prof. Ivan Beschastnikh. Chapters 3, 4 and 5 are primarily adapted from an unpublished manuscript.

This work was performed in collaboration with Qifan Yan, Feijie Wu, Prof. Mathias Lécuyer and Prof. Lanjun Wang. In this work, I developed key ideas, implemented the proposed algorithms, and conducted most experiments. Qifan Yan assisted me with the implementation of sticky sampling described in Chapter 3 and the experiment described in Chapter 5. Feijie Wu helped me complete the theorem presented in Chapter 4. Profs. Ivan Beschastnikh, Mathias Lécuyer and Lanjun Wang offered suggestions on the framework design and contributed to the manuscript writing.

Table of Contents

List of Tables

List of Figures

List of Theorems

List of Propositions

List of Algorithms

Acknowledgements

This research was funded by the Natural Science and Engineering Research Council of Canada (NSERC) Discovery Grant (RGPIN-2020- 05203), as well as Huawei Technologies Co. and the Data Science Institute (DSI) at the University of British Columbia.

First, I would like to express my deepest appreciation to my supervisor, Prof. Ivan Beschastnikh, who encouraged me to follow my interests and guided me throughout this work. Without his constant support and invaluable guidance, this work would not have achieved its present form.

I would like to further thank Profs. Mathias Lécuyer and Lanjun Wang for their contributions. They provided numerous insightful suggestions and motivated me to work smarter. I am also grateful to my collaborators Qifan Yan and Feijie Wu, who spent countless nights with me on this work. It has been a great pleasure working with you all.

In addition, I want to give special thanks to Prof. Frank Wood and his Programming Languages for Artificial Intelligence (PLAI) research group. They offered GPU resources for me to conduct the experiments in this work. I also want to thank my friends in Hong Kong and Vancouver: Yutong Yang, Jiahui Huang, Jincheng Chen, Xindong Lin, Ruochen Qin, and many more who helped me release stress during the pandemic period.

Lastly, I would like to extend my thanks to my parents and my fiancée, Xinyu Ma, for their unconditional love and company. Thank you for making my life wonderful.

Chapter 1

Introduction

Federated learning (FL) moves machine learning (ML) training to the edge. In FL, edge clients communicate with a central server to collaboratively train a global model, while keeping client training data local. We focus on *cross-device* FL, in which there are many clients that are end-user devices. For example, companies like Google and Intel use cross-device FL for computer vision and natural language processing model training across customer devices [11, 12, 45].

One downside of FL is its network usage. This is especially problematic in cross-device FL, which relies on lower-bandwidth mobile or IoT devices [16]. For example, Google Keyboard (Gboard), a virtual keyboard with over 1 billion installs, selects clients from millions of mobile devices to enhance its search query suggestions [45]. In this type of application, clients usually have a diversity of device-to-server (upstream) and server-to-device (downstream) bandwidth. Clients that have either slow upstream or downstream bandwidth act as stragglers and slow down model training.

This heterogeneous bandwidth setting has attracted significant research, with a focus on reducing the communication cost of FL training [7, 24, 30, 32, 36]. One important strategy is *client sampling*, which limits the number of clients that perform training in each round [23, 24]. Client sampling reduces both upstream and downstream bandwidth. However, a client that is not sampled gradually becomes stale: its local state diverges from the state of clients that have been sampled. The next time this client is sampled, the central server must therefore send a larger state update, increasing the downstream transmission overhead.

Another approach to reducing FL bandwidth usage is to apply a mask to the client gradients, such as a *sparsification* mask [32, 38] or a *parameter freezing* mask [5, 7]. In traditional masking schemes, clients apply a mask to their local gradients and only transfer significant gradients to the server. This saves upstream bandwidth. Since each client generates the mask locally and independently, however, the entire model is usually updated at the end of a round and needs to be fully synchronized. In *server masking* schemes, such as Sparse Ternary Compression (STC) [32] and Adaptive Parameter

Freezing (APF) [7],

the server uses a mask to compute the final model update. Since the server only partially updates the model, only a part of the model needs to be sent back to clients; this saves downstream bandwidth.

User sampling and masking approaches are typically considered as orthogonal, compatible approaches [7, 32]. Though existing masking strategies are indeed empirically effective in full participation FL, we show that when client sampling is used they fail to decrease *downstream* bandwidth (§2). For example, with a 0.01 sample ratio and a masking compression ratio of 10%, a single client needs to download 75% of the global model on average.

The reason downstream bandwidth increases is because of *the staleness of local state at the clients*. To see why, let us first consider the full participation case. Intuitively, since the global model is only partially updated by the server under masking, a client only needs to download this partial update and apply it to its local version of the model, saved from the previous round. With client sampling however, a typical client skips multiple rounds by not being sampled, and its local model state becomes stale. When the client is later sampled, it needs to download the new value of all parameters updated in the skipped rounds, which amounts to a large fraction of the model. This effect increases downstream bandwidth usage, voiding the benefits of server masking, and slowing down training when edge devices have limited download capacity [1].

To resolve the incompatibility between masking and client sampling, we propose GlueFL, a new FL training framework specifically designed to retain the benefits of masking when using client sampling. This compatibility is particularly important in cross-device FL deployments, which require both client sampling (full participation is impractical) and bandwidth savings due to mobile or IoT clients. To the best of our knowledge, GlueFL is the first masking design to address the downstream bandwidth bottleneck in cross-device FL with client sampling.

We design GlueFL with two new mechanisms to alleviate client staleness and to optimize downstream bandwidth requirements. First, we introduce *sticky sampling* (§3.1) to prioritize the most recently used clients, thereby reducing the number of stale clients in each update. Since recently selected clients have an up-to-date view of model parameters, they need to download smaller updates. We combine sticky sampling with a weighted central aggregation scheme to ensure that model updates remain unbiased, a requirement for convergence (§4). Sticky sampling is especially important in practical implementations that sample a small fraction of clients in each round [45].

Second, we propose a gradual *mask shifting* strategy (§3.2), to ensure

that consecutive central model updates share a large number of changed parameters, while empirically preserving model convergence. This way, a newly selected client only has to synchronize a subset of the model, even after several rounds of not being sampled.

To sum up, we make three contributions:

- ⋆ We present an FL design called GlueFL, which is based on sticky sampling and mask shifting. These two new mechanisms alleviate the impact of client staleness in client sampling. Both techniques minimize downstream bandwidth in cross-device FL. To the best of our knowledge, this is the first work to combine masking with client sampling to reduce downstream bandwidth.
- ⋆ We analyse FL convergence under GlueFL's sticky sampling, and show that our proposed weighted aggregation preserves unbiasedness of updates and convergence.
- ⋆ We evaluate GlueFL empirically, and demonstrate downstream bandwidth and training time savings on three public datasets. On average, our evaluation shows that GlueFL spends 29% less training time with a 27% less downstream bandwidth overhead as compared to FedAvg [24], STC [32] and APF [7].

Chapter 2

Motivation and Background

We start by reviewing standard FL with client sampling. Then we introduce a state of the art masking strategy called STC [32], and discuss its limitations. Finally, we formalize the problem that we set out to solve in the rest of the thesis. Table 2.1 overviews our notation.

Table 2.1: Summary of notation used in this thesis.

$\mathcal{N}, N, i$	set, total number, index of *clients*
$\mathcal{K}, K$	set, number of *sampled clients*
T, t	number, index of *communication rounds*
E, e	number, index of *local update steps*
$\mathbf{w}^t$	server model in round t
$\mathbf{w}_i^{t,e}, \mathbf{g}_i^{t,e}$	model, gradients of client i in round t and step e
$\mathcal{S}, S$	set, size of *sticky group*
$\mathcal{C}, C$	set, number of clients sampled from $\mathcal{S}$
$\mathcal{R}, R$	set, number of clients sampled from $\mathcal{N} \setminus \mathcal{S}$
$\nu_{i,s}, \nu_{i,r}$	aggregation weight of client i in $\mathcal{C}$, $\mathcal{R}$
q, q_{shr}	total, shared *mask ratio*

2.1 Federated Learning (FL)

Consider a system with N clients, coordinated by a central server. Each client i has a local data distribution $\mathcal{D}_i$. Let us denote the weight of client i as p_i such that $\sum_{i=1}^{N} p_i = 1$. The weight p_i is given by the server and represents the importance of the i-th client's local loss function. Under the non-convex settings, our target is formulated as

$$\min_{\mathbf{w} \in \mathbb{R}^d} \quad F(\mathbf{w}) \stackrel{\triangle}{=} \sum_{i=1}^{N} p_i F_i(\mathbf{w}) \tag{2.1}$$

where $F_i(\mathbf{w}) = \frac{1}{|\mathcal{D}_i|} \sum_{\xi \in \mathcal{D}_i} \ell(\mathbf{w}, \xi)$, and $\ell(\mathbf{w}, \xi)$ is the empirical loss on model $\mathbf{w}$ and sample ξ. In practice, $F_i(\mathbf{w})$ is generally estimated with a ran-

dom realization ξ_i drawn from $\mathcal{D}_i$, which is assumed to be unbiased, i.e., $\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \ell(\mathbf{w}, \xi_i) = F_i(\mathbf{w})$. Let F_* is the minimum value of the global objective, i.e., $F(\mathbf{w}) \geq F_*$ for any $\mathbf{w} \in \mathbb{R}^d$.

FedAvg [24] is a standard algorithm to solve Equation (2.1). To improve communication efficiency, clients are selected uniformly at random in each round. The FedAvg algorithm with client sampling looks as follows:

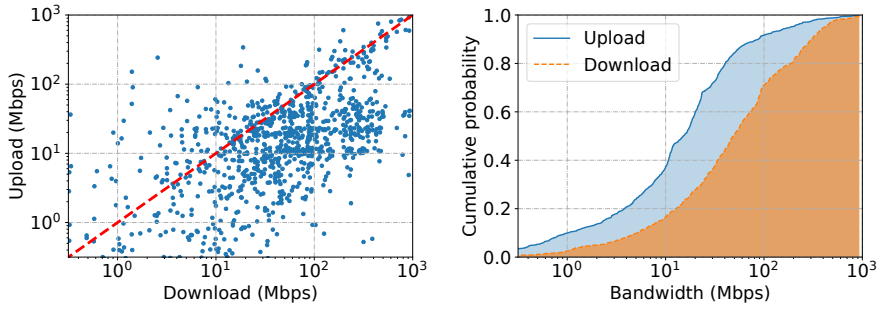
1. At the beginning of round t, the server uniformly at random samples a subset of clients (i.e., $\mathcal{K}$) and broadcasts the latest global model $\mathbf{w}^t$ to these sampled clients.
2. Each sampled client $i \in \mathcal{K}$ receives the model $\mathbf{w}^t$ ($= \mathbf{w}_i^{t,0}$) and runs E local SGD iterations to compute a local update $\Delta_i^t = -\gamma \sum_{e=0}^{E-1} g_i^{t,e}$, where γ is the client learning rate. In each iteration, the client computes the gradient as $g_i^{t,e} = \nabla \ell(\mathbf{w}_i^{t,e}, \xi_i^{t,e})$ where $\xi_i^{t,e}$ is drawn from $\mathcal{D}_i$.
3. The server receives updates Δ_i^t from all sampled clients and aggregates them to compute the new global model [22]

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \frac{N}{K} \sum_{i \in \mathcal{K}} p_i \Delta_i^t \tag{2.2}$$

In expectation, the steps above realize an update form $\mathbb{E}_{\mathcal{K}}\left[\mathbf{w}^{t+1}\right] = \mathbf{w}^t + \sum_{i=1}^{N} p_i \Delta_i^t$ in each round. To ensure that the global loss approaches the optimal one, FedAvg repeats the process for T rounds. FedAvg achieves a convergence rate of $O\left(\sqrt{\frac{E}{KT}}\right)$ [17, 44] under partial worker participation.

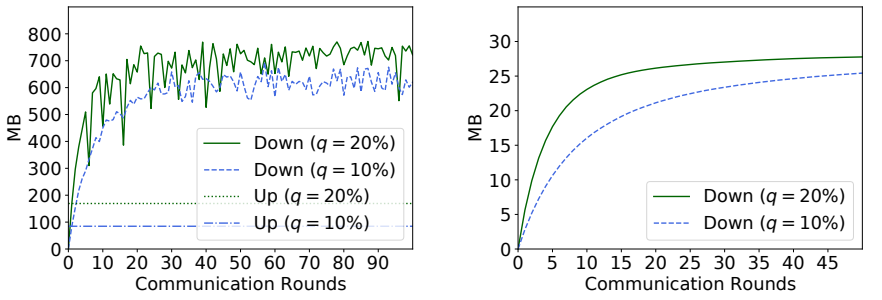
2.2 Cross-device FL Bandwidth Characteristics

The cross-device FL setting relies on a large number of clients. In this case, some clients are likely to have an unreliable or slow network. For example, Figure 2.1 shows the bandwidth distribution estimated by measurement lab [25]. We observe that around 20% of devices have a download bandwidth of at most 10Mbps. These devices can take at least 20s to download a typical model like ShuffleNet_V2 [46], which is specially designed for mobile devices and contains 5 million model parameters.



(a) (b)

Figure 2.1: (a) The distribution of network bandwidth in North America, June 2022 [25], and (b) the cumulative distribution function (CDF) of network bandwidth in (a).



(a) (b)

Figure 2.2: (a) The downstream and upstream bandwidth usage of STC per round, and (b) the model size a client must download when being re-sampled after a certain number of rounds.

2.3 Limitations of Existing Masking Strategies

Prior work has proposed several masking strategies to reduce the amount of transferred data and alleviate low bandwidth issues [5, 7, 32, 38]. To demonstrate how masking fails to optimize downstream bandwidth in FL with client sampling, we use STC [32], a popular server masking strategy, as a representative technique.

STC builds on top-k sparsification [34], a masking approach that selects and uploads the largest q (e.g., 10%) absolute values in a client's local gra-

Algorithm 1: Sparse Ternary Compression (STC)

Output: $\mathbf{w}^T$

1 **for** $t \leftarrow 1$ **to** T **do**
2 `/* Server:client sampling */`
3 Generate set of sampled clients $\mathcal{K}$;
4 Broadcast $\mathbf{w}^t$ to $\mathcal{K}$;
5 `/* Client:local training */`
6 **for** $i \in \mathcal{K}$ *in parallel* **do**
7 $\mathbf{w}_i^{t,0} \leftarrow \mathbf{w}^t$;
8 **for** $e \leftarrow 0$ **to** $E-1$ **do**
9 $\mathbf{w}_i^{t,e+1} \leftarrow \mathbf{w}_i^{t,e} - \gamma \mathbf{g}_i^{t,e}$;
10 **end**
11 `/* Client:sparsification */`
12 $\tilde{\Delta}_i^t \leftarrow top_q(\mathbf{w}_i^{t,E} - \mathbf{w}_i^{t,0})$;
13 **end**
14 `/* Server:aggregation */`
15 Receive $\tilde{\Delta}_i^t$ from worker $i \in \mathcal{K}$;
16 `/* Server:sparsification */`
17 $\tilde{\Delta}^t \leftarrow top_q(\sum_{i \in \mathcal{K}} p_i \frac{N}{K} \tilde{\Delta}_i^t)$;
18 $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \tilde{\Delta}^t$;
19 **end**

dients. In STC, this top-k sparsification technique is applied to both clients' gradients and server updates[1]. Algorithm 1 shows this masking-only version of STC. For a single client sampled in both the current and last round, STC only has to update the weights covered by the server mask (line 17). However, note that a client that has *not* been sampled recently may have to update the entire model, as their local view of the model is stale. The reason is that server masks change in each round, and the client has to synchronize all updated model parameters since it last participated.

To measure the impact of model staleness on downstream bandwidth, we apply STC to FedAvg and conduct experiments on FEMNIST, using $N = 2,800$ clients and a client sample size of $K = 30$. We try compression ratios

[1]For simplicity, we only consider the masking part of STC. STC also includes quantization, an orthogonal technique that can be combined with sparsification [2, 15] and will not change our conclusion, as quantization compresses both downstream and upstream communication.

of 10% and 20%[2]. We examine both downstream and upstream bandwidth usage in each round. The results in Figure 2.2 show that upstream bandwidth is reduced when using a smaller compression ratio, as expected. However, a client still needs to download 70% of the global model on average. Clients with 10Mbps download bandwidth (§2.2) will take at least 14s to receive these changes. This imposes a high downstream bandwidth requirements on participating clients. In general, the more rounds that a client skips, the more updated model state it needs to download (Figure 2.2(b)). As a result, the training bottleneck shifts to downstream communication. We expect these results to hold for other masking strategies as they similarly update different parts of the global model in each communication round. For example, in APF [7], model parameters are frozen in some rounds but will then be updated again after the freezing period ends. The downstream bottleneck is therefore a general limitation across masking strategies.

2.4 Problem Setup

Our goal in GlueFL is to minimize the total expected downstream bandwidth of training, while retaining a low upstream bandwidth, and ensuring that the expected global training loss $F(\mathbf{w}^T)$ converges to a local minimum value, where $\mathbf{w}^T$ is the aggregated global model after T rounds.

[2]Smaller values led STC to require an unacceptable number of rounds to converge with a noticeable drop in convergence accuracy.

Chapter 3

GlueFL Framework Design

GlueFL includes two components to decrease the downstream bandwidth during FL training: sticky sampling (Figure 3.1) and mask shifting (Figure 3.2). The newly designed sampling scheme allows some clients to be re-sampled in a short term and mask shifting restricts the mask from changing too fast. We elaborate on the design of each of these components in §3.1 and §3.2, before describing how to adapt other existing mechanisms in §3.3.

3.1 Sticky Sampling

Client sampling is the process of selecting K out of N clients in each round, to participate in computing the model update. With uniform sampling, each client participates in each round with a probability of K/N. Thus, a client is expected to participate in training every N/K rounds on average (See Proposition 1 in Appendix 3.1.1). In cross-device FL systems, the value of N is often large, and K is small. For example, Gboard samples $K = 100$ clients in each round while there are millions of devices [45]. This produces a low probability of participation in each round, which means that on average clients skip a large number of training rounds before being selected again. As we saw in §2.3, these long skips are responsible for local state staleness. Clients' state must therefore be re-synchronized when they are selected, reducing the benefits of masking on downstream bandwidth.

GlueFL introduces *sticky sampling* to ensure that clients with an up-to-date local state are more likely to be selected. Figure 3.1 illustrates sticky sampling and Algorithm 2 details it. The server maintains a smaller *sticky group* of clients $\mathcal{S}$ with size S, while the remaining clients form a *non-sticky* group, $\mathcal{N} \setminus \mathcal{S}$. We randomly select S clients to initialize $\mathcal{S}$ in the beginning of training, and allow $\mathcal{S}$ to evolve over time.

Figure 3.1 (step 1) illustrates how in each FL training round, the server constructs its sampled set of clients $\mathcal{K}$ from two sources; $\mathcal{K} = \mathcal{C} \cup \mathcal{R}$. It samples C clients to construct $\mathcal{C}$ by sampling from the current sticky group $\mathcal{S}$. It samples $(K-C)$ clients to construct $\mathcal{R}$ by sampling from the non-sticky

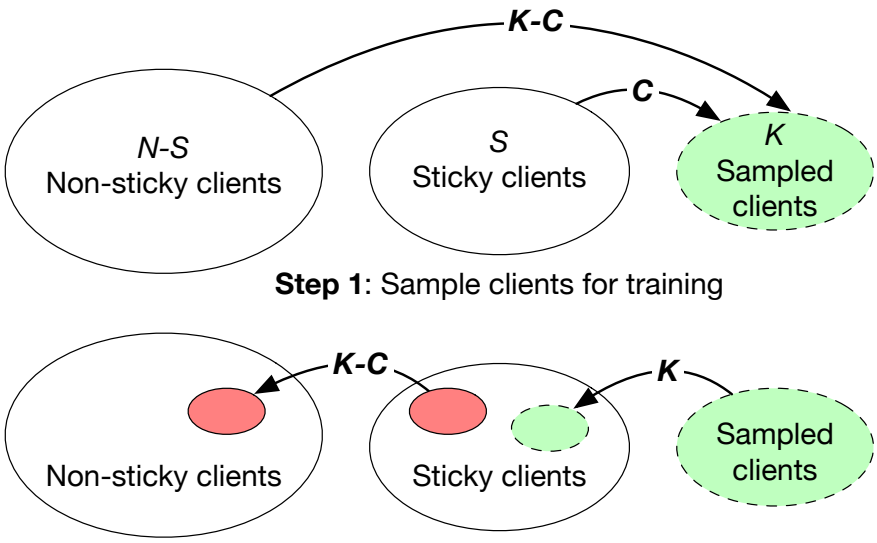


Figure 3.1: Sticky sampling design.

group, without replacement. All sampled clients $\mathcal{K}$ participate in one round of training (Algorithm 2 line 5).

At the end of the round (Figure 3.1 (step 2)), the server randomly selects $(K - C)$ clients from $\mathcal{S} \setminus \mathcal{C}$ (the set of clients in the sticky group that did not participate in the latest round) and removes these clients from the sticky group (Algorithm 2 line 20). The server replaces these clients with $(K - C)$ clients that were *not* sampled from the sticky group and that participated in the last update ($\mathcal{R}$ in Algorithm 2).

Just as with uniform client sampling, sticky client sampling requires N/K rounds to re-sample a client on average (see Proposition 2 for details). However, a client selected with sticky sampling will join the sticky group and then have a higher probability of being selected in the next round than under uniform sampling, as long as $\frac{C}{S} > \frac{K}{N}$. Since a client that exits the sticky group (by not being selected in the current round) is less likely to be selected than under uniform sampling ($\frac{K-C}{N-S} < \frac{K}{N}$ when $\frac{C}{S} > \frac{K}{N}$), we need to ensure that a sticky client has a higher expectation of being included during the next several rounds. This is because after several missed rounds, the whole model needs to be synchronized (see Figure 2.2(b)). Proposition 2 in Appendix 3.1.1 shows the probability for a client in the sticky group to be

Algorithm 2: Sticky Sampling

Output: $\mathbf{w}^T$

for $t \leftarrow 1$ **to** T **do**
  `/* Server:sample clients */`
  Randomly select $|\mathcal{C}| = C$ clients from $\mathcal{S}$;
  Randomly select $|\mathcal{R}| = K - C$ clients from $\mathcal{N} \setminus \mathcal{S}$;
  Set of sampled clients $\mathcal{K} \leftarrow \mathcal{C} \cup \mathcal{R}$;
  Broadcast $\mathbf{w}^t$ to $\mathcal{K}$;
  `/* Client:local training */`
  for $i \in \mathcal{K}$ *in parallel* **do**
    $\mathbf{w}_i^{t,0} \leftarrow \mathbf{w}^t$;
    for $e \leftarrow 0$ **to** $E - 1$ **do**
      $\mathbf{w}_i^{t,e+1} \leftarrow \mathbf{w}_i^{t,e} - \gamma \mathbf{g}_i^{t,e}$;
    end
    $\Delta_i^t \leftarrow \mathbf{w}_i^{t,E} - \mathbf{w}_i^{t,0}$;
  end
  `/* Server:aggregation */`
  Receive Δ_i^t from worker $i \in \mathcal{K}$;
  $\Delta^t \leftarrow \sum_{i \in \mathcal{C}} \nu_{i,s}^t \Delta_i^t + \sum_{i \in \mathcal{R}} \nu_{i,r}^t \Delta_i^t$;
  $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \Delta^t$;
  `/* Server:rebalance non-sticky and sticky groups */`
  Randomly remove $K - C$ clients in $\mathcal{S} \setminus \mathcal{C}$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{R}$;
end

selected after r rounds. We use this formula to select S and C to ensure that this probability is higher than that of uniform sampling for a high enough value of r.

Case Study. Consider a training run on FEMNIST with $N = 2,800$ clients, $K = 30$, $S = 120$, and $C = 24$ (our default experimental setup in §5.2). In this case, using the Proposition 1 and Proposition 2 in Appendix, we can compute the probability of client inclusion over the next 6 rounds for a client starting in the sticky group: $20.0\%, 15.0\%, 11.2\%, 8.5\%, 6.4\%, 4.8\%$. By contrast, uniform sampling re-samples clients with a probability of around 1.1%.

With sticky sampling, clients that just participated in a round, and thus have an up-to-date state, are more likely to participate again in the short term. Such clients will therefore download smaller model updates. This

synergizes with masking approaches that reduce the size of an update in each round. We show in §5 that for cross-device FL, where a large N and a small K are typical, masking approaches with sticky sampling significantly reduce downstream bandwidth usage.

However, sticky sampling also introduces new challenges during aggregation. As discussed in §2, the global update should provide appropriate representation for every client in expectation [26, 37, 40]. Formally, the update should be an unbiased estimate of the FedAvg update computed on every client in round t. That is: $\mathbb{E}_{\mathcal{K}}[\Delta^t] = \sum_{i=1}^{N} p_i \Delta_i^t$. Under the FedAvg aggregation function (Equation (2.2)), since sticky clients are selected with higher probability, they would have a larger weight then non-sticky clients. To correct for this bias, GlueFL uses an inverse propensity weighted aggregation function. It assigns a different weight to updates from clients of different groups, corresponding to their importance parameter re-weighted by the inverse probability of selection. Updates from sticky group clients use the weight $\nu_{i,s}^t = \frac{S}{C} p_i$, while non-sticky group clients use the weight $\nu_{i,r}^t = \frac{N-S}{K-C} p_i$. The model update rule then becomes:

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \underbrace{\sum_{i \in \mathcal{C}} \nu_{i,s}^t \cdot \Delta_i^t + \sum_{i \in \mathcal{R}} \nu_{i,r}^t \cdot \Delta_i^t}_{\Delta^t} \tag{3.1}$$

This is shown in lines 17 and 18 of Algorithm 2.

With this reweighting scheme in place, we can show that sticky sampling udpates are unbiased:

Theorem 1 (Unbiased Aggregation)**.** *Let $\mathcal{K} = \mathcal{C} \cup \mathcal{R}$ be the set of sampled clients in sticky sampling. The update Δ^t computed in Equation* (3.1) *is unbiased. That is:*

$$\mathbb{E}_{\mathcal{K}}[\Delta^t] = \sum_{i=1}^{N} p_i \Delta_i^t \tag{3.2}$$

Proof. We can rewrite the update as a sum over all the data, where the

probability of inclusion cancels out with the aggregation weight:

$$\begin{aligned}
\mathbb{E}_{\mathcal{K}}[\Delta^t] &= \mathbb{E}_{\mathcal{K}}\left[\sum_{i\in\mathcal{C}}\frac{S}{C}p_i\Delta_i^t + \sum_{i\in\mathcal{R}}\frac{N-S}{K-C}p_i\Delta_i^t\right] \\
&= \mathbb{E}_{\mathcal{K}}\left[\sum_{i\in\mathcal{S}}\mathbb{1}_{\{i\in\mathcal{C}\}}\frac{S}{C}p_i\Delta_i^t + \sum_{i\in\mathcal{N}\setminus\mathcal{S}}\mathbb{1}_{\{i\in\mathcal{R}\}}\frac{N-S}{K-C}p_i\Delta_i^t\right] \\
&= \sum_{i\in\mathcal{S}}\frac{C}{S}\frac{S}{C}p_i\Delta_i^t + \sum_{i\in\mathcal{N}\setminus\mathcal{S}}\frac{K-C}{N-S}\frac{N-S}{K-C}p_i\Delta_i^t \\
&= \sum_{i=1}^{N} p_i\Delta_i^t
\end{aligned}$$

where $\mathbb{1}_{\{\text{predictate}\}}$ is the indicator function with value 1 when the predicate is true, and 0 otherwise. □

§ 4.3 shows that estimating unbiased updates is key to analyzing the convergence of GlueFL, following proof techniques from [8, 9].

3.1.1 Analysis of Sampling Schemes

In this section, we provide a comparison between uniform sampling and sticky sampling to demonstrate the advantage of sticky sampling. We first analyze the probability that a client is re-sampled after r rounds and then give the expected number of rounds for a client to be re-sampled.

Proposition 1 (Uniform Sampling)**.** *Suppose a client is sampled at the current round. With uniform sampling, there is a probability of* $\frac{K}{N}(1-\frac{K}{N})^{r-1}$ *that the client is sampled after* r *rounds. On average, a client is sampled every* N/K *rounds.*

Proof. The client is sampled with a probability of $\frac{K}{N}$. The client has not been selected for the first $(r-1)$ rounds. Thus, this happens with a probability of $\frac{K}{N}(1-\frac{K}{N})^{r-1}$. Furthermore, the value of averaged sampled rounds is $\sum_{r=1}^{\infty}\frac{K}{N}(1-\frac{K}{N})^{r-1}\cdot r = N/K$. □

Proposition 2 (Sticky Sampling)**.** *Suppose a client is sampled at the current round. Using sticky sampling, the client in the sticky group is sampled with a probability of* $\frac{1}{(N-S)K-(K-C)S}(\frac{K(NC-SK)}{S}(1-\frac{K}{S})^{r-1}+(K-C)^2(1-\frac{K-C}{N-S})^{r-1})$ *after* r *rounds. As expected, the client trains a model every* N/K *rounds.*

Proof. In the sticky group, a client is sampled or moved to the non-sticky group with the probability of $\frac{C}{S}$ and $\frac{K-C}{S}$, respectively. And, a client is sampled from the non-sticky group with probability $\frac{K-C}{N-S}$.

There are two strategies to sample a client that has participated in model training. First, it is sampled from the sticky group, where the probability is $\frac{C}{S}(\frac{S-K}{S})^{r-1}$ after r rounds. Second, it is sampled from the non-sticky group, indicating the client is moved out of the sticky group in the middle. Therefore, the probability is $\sum_{i=1}^{r-1}(1-\frac{K-C}{N-S})^{i-1}\cdot\frac{K-C}{N-S}\cdot(\frac{S-K}{S})^{r-i-1}\cdot(\frac{K-C}{S}) = \frac{(K-C)^2}{(N-S)K-(K-C)S}((1-\frac{K-C}{N-S})^{r-1}-(\frac{S-K}{S})^{r-1})$. By summing up these two probabilities, we can obtain the desired result. Furthermore, similar to Proposition 1, we can calculate the value of averaged sampled rounds. □

Discussion According to the proof of Proposition 2, the probability of a client in the sticky group being sampled after r rounds is greater or equal to $\frac{C}{S}(\frac{S-K}{S})^{r-1}$, which is the probability that it is still sampled from the sticky group. Then, for $r \in \{1,\ldots,1+\left\lfloor\left(\log\frac{CN}{SK}\right)/\left(\log\frac{S(N-K)}{N(S-K)}\right)\right\rfloor\}$, $\frac{C}{S}(\frac{S-K}{S})^{r-1}$ is greater or equal to $\frac{K}{N}(1-\frac{K}{N})^{r-1}$, the probability that a client is sampled after r rounds in uniform sampling (Proposition 1).

3.2 Mask Shifting

Sticky sampling allows clients in a sticky group to be sampled more frequently. However, sticky sampling alone is insufficient. As we have seen in Figure 2.2, a client re-sampled after 10 rounds still needs to download around 50%-80% of the global model on average. This is because the masked updates of two successive rounds (e.g., $\tilde{\Delta}^t$ and $\tilde{\Delta}^{t+1}$) have little overlap.

We solve this issue by designing a gradual mask shifting strategy, that prevents the mask from changing too quickly while ensuring that the total compression ratio is maintained. Figure 3.2 illustrates our mask shifting design. We construct a shared mask with compression ratio q_{shr} (with $q_{shr} < q$), which is represented using a bitmap shared with selected clients in $M^t \in \mathbb{B}^d$ in round t. Clients send their update for parameters in M^t, as well as a $q - q_{shr}$ proportion of locally important parameters. The server will use M^t as well as locally important parameters to calculate the model update, and to shift M^t to obtain M^{t+1}, while keeping a large overlap between consecutive masks.

Algorithm 3 details the mechanism, with sticky sampling from Algorithm 2 used to select clients in lines 5 and 28. The server first synchronizes

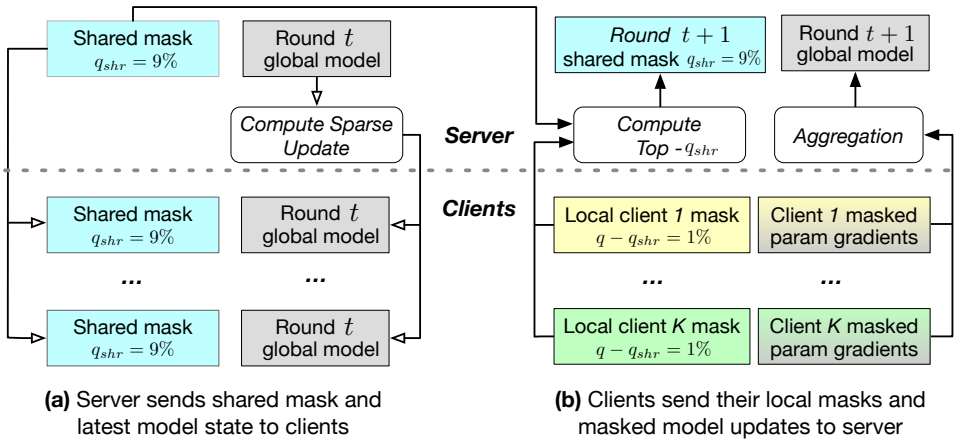


Figure 3.2: Mask shifting design with $q = 10\%$ and $q_{shr} = 9\%$.

the global model $\mathbf{w}^t$ with sampled clients and distributes M^t to them (line 7). In line 16, the client i calculates the shared local gradient $\tilde{\Delta}^t_{i,shr}$ as $M^t \odot \tilde{\Delta}^t_i$, where $\odot$ sets those positions that are not covered by the masks to zero. Next, the algorithm computes unique local gradients $\tilde{\Delta}^t_{i,uni}$ by selecting a $(q - q_{shr})$ proportion of the largest values in other (previously masked) positions, to provide more local information to the server (line 17). Finally, client i sends $\tilde{\Delta}^t_{i,shr}$ and $\tilde{\Delta}^t_{i,uni}$ to the server.

During aggregation, the central server uses sticky sampling importance weights $\nu^t_{i,s}, \nu^t_{i,r}$ given in §3.1. The server first computes the shared update $\tilde{\Delta}^t_{shr}$ based on all client (weighted) updates, and the update based on unique local information by selecting the $(q - q_{shr})$ proportion of largest overall (weighted) gradients (line 23). Formally, each quantity is computed as:

$$\tilde{\Delta}^t_{shr} \leftarrow \sum_{i \in \mathcal{C}} \nu^t_{i,s} \tilde{\Delta}^t_{i,shr} + \sum_{i \in \mathcal{R}} \nu^t_{i,r} \tilde{\Delta}^t_{i,shr} \tag{3.3}$$

$$\tilde{\Delta}^t_{uni} \leftarrow top_{(q-q_{shr})} \left(\sum_{i \in \mathcal{C}} \nu^t_{i,s} \tilde{\Delta}^t_{i,uni} + \sum_{i \in \mathcal{R}} \nu^t_{i,r} \tilde{\Delta}^t_{i,uni} \right) \tag{3.4}$$

These updates are combined and update the global model (line 24). Finally, the shared mask is updated by selecting a share q_{shr} of parameters with the largest update values in the combined update (line 26). Since the new mask M^{t+1} will be used to compute $\tilde{\Delta}^{t+1}$, the overlap of two successive model updates $\tilde{\Delta}^t$ and $\tilde{\Delta}^{t+1}$ is at least q_{shr}.

Algorithm 3: GlueFL

Output: $\mathbf{w}^T$

for $t \leftarrow 1$ **to** T **do**
  `/* Server:sticky sampling */`
  Randomly select $|\mathcal{C}| = C$ clients from $\mathcal{S}$;
  Randomly select $|\mathcal{R}| = K - C$ clients from $\mathcal{N} \setminus \mathcal{S}$;
  Set of sampled clients $\mathcal{K} \leftarrow \mathcal{C} \cup \mathcal{R}$;
  Synchronize $\mathbf{w}^t$ with $\mathcal{K}$ by sending model updates;
  Send shared mask M^t to $i \in \mathcal{K}$;
  `/* Client:local training */`
  for $i \in \mathcal{K}$ *in parallel* **do**
    $\mathbf{w}_i^{t,0} \leftarrow \mathbf{w}^t$;
    for $e \leftarrow 0$ **to** $E - 1$ **do**
      $\mathbf{w}_i^{t,e+1} \leftarrow \mathbf{w}_i^{t,e} - \gamma \mathbf{g}_i^{t,e}$;
    end
    `/* Client:masking */`
    $\Delta_i^t \leftarrow \mathbf{w}_i^{t,E} - \mathbf{w}_i^{t,0}$;
    $\tilde{\Delta}_{i,shr}^t \leftarrow M^t \odot \Delta_i^t$;
    $\tilde{\Delta}_{i,uni}^t \leftarrow top_{(q-q_{shr})}(\neg M^t \odot \Delta_i^t)$;
  end
  `/* Server:aggregation */`
  Receive $\tilde{\Delta}_{i,shr}^t, \tilde{\Delta}_{i,uni}^t$ from worker $i \in \mathcal{K}$;
  Compute $\tilde{\Delta}_{shr}^t$ via Equation (3.3) ;
  Compute $\tilde{\Delta}_{uni}^t$ via Equation (3.4) ;
  $\tilde{\Delta}^t \leftarrow \tilde{\Delta}_{shr}^t + \tilde{\Delta}_{uni}^t$;
  $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \tilde{\Delta}^t$;
  `/* Server:update shared mask */`
  $M^{t+1} \leftarrow top_{q_{shr}}(\tilde{\Delta}_{shr}^t + \tilde{\Delta}_{uni}^t)$;
  `/* Server:update sticky group` $\mathcal{S}$ `*/`
  Randomly remove $K - C$ clients in $\mathcal{S} \setminus \mathcal{C}$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{R}$;
end

3.3 Adapting other Techniques to Work with GlueFL

We further improve the performance of GlueFL by adapting common FL techniques to sticky sampling and mask shifting [7, 10, 29, 33, 35, 42].

Shared Mask Regeneration Previous work [7, 10] showed that model parameters converge at different rates. Meanwhile, a parameter that has converged may become unstable in later rounds. For example, according to [7], it is possible that some 10% of parameters are unstable in both round t and $t+1$, while another 5% parameters are only unstable in round $t+1$. In this case, a small $(q - q_{shr})$ (e.g., 2%) value will slow down convergence, as the shared mask fails to cover the gradients of the unstable 5% of parameters and a large $(q - q_{shr})$ (e.g., 10%) value incurs more bandwidth cost.

To address this, we use a small $(q - q_{shr})$ value while re-generating the entire shared mask M^t every I rounds. To regenerate, we set $q_{shr} = 0$ and update M^t as $top_{q_{shr}}(\tilde{\Delta}^t_{uni})$ (Algorithm 3, line 26). Although this process introduces more downstream overhead in the next few rounds, it speeds up training and reduces overall bandwidth.

Error-Compensation Compression methods, such as quantization and sparsification, slow down model convergence due to the loss of information in client updates [29, 35, 42]. Error-compensation is a technique to alleviate this problem, first proposed to accelerate convergence in 1-bit SGD [33]. The key idea is for clients to (1) remember their local compression error (the difference between their true update and what is actually sent to the server), and (2) add it into the next round's computed local gradient before compression. In GlueFL, we apply error compensation as:

$$\Delta^t_i \leftarrow \Delta^t_i + \frac{\nu^{\varphi(t)}_i}{\nu^t_i} \cdot h^{\varphi(t)}_i \tag{3.5}$$

where ν^t_i is the aggregation weight applied at step t for client i (i.e., $\nu^t_{i,s}$ if they are in the sticky group, $\nu^t_{i,r}$ otherwise; their exact values are defined in §3.1), $\varphi(t)$ indicates the step-index when client i was last selected, and $h^{\varphi(t)}_i$ the compensation vector for client i in round $\varphi(t)$. After that, the client computes $\tilde{\Delta}^t_{i,shr}$ and $\tilde{\Delta}^t_{i,uni}$ (Algorithm 3, lines 16-17). Then, the compensation vector is calculated as $h^t_i = \Delta^t_i - (\tilde{\Delta}^t_{i,shr} + \tilde{\Delta}^t_{i,uni})$.

The reason for scaling with $h^{\varphi(t)}_i$ in Equation (3.5) is to ensure that client i's compensation is consistent with the aggregation in sticky sampling. As

the compensation only applies to a client's local gradient before masking, this optimization does not introduce extra bandwidth and improves convergence performance.

Chapter 4

Convergence Analysis

From a theoretical perspective, we show that GlueFL without masking can achieve convergence at a rate of $O(1/\sqrt{T})$ for smooth non-convex functions under two assumptions (§4.1). §4.2 states our result and their interpretation, with details in §4.3.

4.1 Assumptions

We make a standard assumption that clients sample a mini-batch in each local update such that the computed gradient is equal to the true gradient in expectation [17, 22, 37, 41, 44]. That is, $\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \nabla f_i(\mathbf{w}, \xi_i) = \nabla F_i(\mathbf{w})$ for all workers $i \in \{1, \ldots, N\}$ and the model $\mathbf{w} \in \mathbb{R}^d$, where ξ_i and $\mathcal{D}_i$ represent the mini-batch and the local training set, respectively.

We make two more assumptions:

Assumption 1 (Bounded Local Variance)**.** *There exists a constant $\sigma > 0$, such that the variance of each local gradient estimator is bounded by,*

$$\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \|\nabla f_i(\mathbf{w}, \xi_i) - \nabla F_i(\mathbf{w})\| \leq \sigma^2, \quad \forall i \in [N].$$

We also assume that the local objective functions (i.e., $F_1, \ldots, F_N$) and their derivatives are Lipschitz continuous.

Assumption 2 (Continuity and Smoothness)**.** *The local objective functions are L_c-continuous and L_s-smooth.*

4.2 Convergence Result

Here we analyze the convergence rate of sticky sampling (Algorithm 2) on non-convex local objective functions. See §4.3 for the complete proof.

Theorem 2 (Convergence)**.** *Suppose Assumptions 1 and 2 hold, and set the aggregation weights as* $\nu_{i,s}^t = \frac{S}{C} p_i$ *and* $\nu_{i,r}^t = \frac{N-S}{K-C} p_i$*. Let the learning rate be*

$$\gamma = \sqrt{\frac{1}{E(\sigma^2 + E)} \cdot \frac{K}{TA}} \tag{4.1}$$

Algorithm 2 is such that:

$$\min_{t \in \{1,\ldots,T\}} \left\| \nabla F(\mathbf{w}^t) \right\|_2^2 = O\left(\sqrt{\left(1 + \frac{\sigma^2}{E}\right) \cdot \frac{A}{KT}} \right) + O\left(\frac{K}{TA}\right) \tag{4.2}$$

where $A = \frac{K}{N}\left(\frac{S^2}{C} + \frac{(N-S)^2}{K-C}\right)\left(\sum_{i=1}^N p_i^2\right)$*. We treat* L_s*,* L_c*, and* $F(\mathbf{w}^1) - F^*$ *as constants.*

This result gives a convergence rate for reaching a fixed point during model training.

Comparison with FedAvg. If all clients have equal weights, (i.e., $p_i = \frac{1}{N}$ for all workers $i \in \{1, \ldots, N\}$), and the sticky group does not exist (i.e., $S = 0$), the algorithm reduces to FedAvg, and $A = 1$. As we can see, when we set the number of local updates $E \geq \sigma^2$ and T is sufficiently large, the convergence result is led by $O\left(\sqrt{\frac{1}{KT}}\right)$. This is comparable to the state-of-the-art works on convergence of FedAvg as described in §2.1. Sticky-sampling introduces a variance cost (the $\frac{S^2}{C} + \frac{(N-S)^2}{K-C}$ term in A) to remain unbiased under non-uniform client sampling. Next, we show empirically that this is a favorable trade-off given the bandwidth savings enabled by sticky sampling (§5).

4.3 Proof of Theorem 2

In this section, we theoretically analyze the convergence rate of sticky-sampling in GlueFL on non-convex functions, under Assumptions 1 and 2. The conclusion has been mentioned in Theorem 2. The proof follows the same template as those of [4, 17, 37, 44], and proceeds as follows: (1) we use Lemma 1 to bound the expected progress in each step (§4.3.2) by a sum of two terms. (2) We bound the first term through a bound on local updates (§ 4.3.4) and our unbiased aggregation. (3) We bound the second term by adapting a bound on the norm between two consecutive models to account for our aggregation weights (§4.3.5). (4) We use the bound on the expected

progress in each step in a telescopic sum to bound the overall progress over training (§4.3.3).

We first present steps (1) and (4) in § 4.3.2 and 4.3.3, which represent the high level articulation of the proof, before presenting the lower level results for steps (2) and (3) in § 4.3.4 and 4.3.5.

4.3.1 Some Useful Lemmas

In this section, we provide two useful lemmas, which will apply to our subsequent analysis in §4.3. Lemma 1 is used to present the progress in one single step in FL (§4.3.2) and Lemma 2 is used to bound the gap between two successive global models (Lemma 4).

Lemma 1 ([4]). *Suppose a function $\mathcal{H}$ is L_c-continuous and L_s-smooth. For any $w, v \in \mathbb{R}^d$, the following inequality holds for $\mathcal{H}$:*

$$\|\nabla \mathcal{H}(w)\|_2 \leq L_c; \qquad \mathcal{H}(w) \leq \mathcal{H}(v) + \langle \nabla \mathcal{H}(v), w - v \rangle + \frac{L_s}{2}\|w - v\|_2^2$$

Lemma 2 (Lemma 4 in [17]). *Let $\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_a\}$ be a random variables in $\mathbb{R}^d$, which are not assumed to be independent. If $\mathbb{E}[\varepsilon_i] = e_i$, and the variance is bounded by $\mathbb{E}\left[\|\varepsilon_i - e_i\|_2^2\right] \leq \sigma^2$, we have:*

$$\mathbb{E}\left[\left\|\sum_{i=1}^{a} \varepsilon_i\right\|_2^2\right] \leq \left\|\sum_{i=1}^{a} e_i\right\|_2^2 + a^2\sigma^2$$

If we further suppose that $\mathbb{E}[\varepsilon_i | \varepsilon_{i-1}, \ldots, \varepsilon_1] = e_i$, in which case the $\{\varepsilon_i - e_i\}$ form a martingale difference sequence, and the bound of the variance $\mathbb{E}\left[\|\varepsilon_i - e_i\|_2^2\right] \leq \sigma^2$ holds, we have the following, tighter bound:

$$\mathbb{E}\left[\left\|\sum_{i=1}^{a} \varepsilon_i\right\|_2^2\right] \leq 2\left\|\sum_{i=1}^{a} e_i\right\|_2^2 + 2a\sigma^2$$

4.3.2 Progress in one single step

We first bound the expected progress after one step of the model update. By definition, $\mathbf{w}^{t+1} = \mathbf{w}^t - \gamma \sum_{i \in \mathcal{K}^t} \nu_i^t \sum_{e=0}^{E-1} \cdot \mathbf{g}_i^{t,e}$, where ν_i^t can be either $\nu_{i,s}^t$ or $\nu_{i,r}^t$ depending on the client's membership. Since all local objective

functions are L_s-smooth, the global objective F is L_s-smooth as well. Thus, according to Lemma 1, we have:

$$\mathbb{E}_{t+1|t}\Big[F\left(\mathbf{w}^{t+1}\right)\Big] - F(\mathbf{w}^t) \leq \underbrace{\mathbb{E}_{t+1|t}\left\langle \nabla F(\mathbf{w}^t), \mathbf{w}^{t+1} - \mathbf{w}^t \right\rangle}_{\mathcal{Q}_1} \tag{4.3}$$

$$+ \frac{L_s}{2} \underbrace{\mathbb{E}_{t+1|t}\left\|\mathbf{w}^{t+1} - \mathbf{w}^t\right\|_2^2}_{\mathcal{Q}_2} \tag{4.4}$$

where $\mathbb{E}_{t+1|t}$ means the expected value at round $(t+1)$, condition on all information at round t, including the model $\mathbf{w}^t$ and the participants $\mathcal{K}^{t-1}$. The expectation is over the randomness of client selection ($\mathcal{K}^t$) and batch selection at the client's ($\xi_i \sim \mathcal{D}_i$ from § 2.1).

We first provide the upper bound analysis for term $\mathcal{Q}_1$. Intuitively, our unbiased aggregation combines with a technical client local drift bound adapted from previous work (§ 4.3.4) to decompose this term. Remember that as Theorem 1 indicates, our weighted update is an unbiased estimate of the true update over all clients. That is:

$$\mathbb{E}_{t+1|t}\Big[\mathbf{w}^{t+1} - \mathbf{w}^t\Big] = \mathbb{E}_{t+1|t}\Big[\Delta^t\Big] = \sum_{i=1}^{N} p_i \mathbb{E}_{\xi_i \sim \mathcal{D}_i}\big[\Delta_i^t\big], \tag{4.5}$$

where we decomposed $\mathbb{E}_{t+1|t}$ in the randomness over client sampling, and local updates. The expectation in the right-hand side is over the local training steps of each client. Based on the form of local updates, we have that $\mathbb{E}_{\xi_i \sim \mathcal{D}_i}\big[\mathbf{w}_i^{t,E} - \mathbf{w}^t\big] = -\gamma \sum_{e=0}^{E-1} \mathbb{E}_{\xi_i \sim \mathcal{D}_i}\big[\mathbf{g}_i^{t,e}\big]$. Considering the unbiased estimation assumption mentioned in Section 2.1, we have that $\forall e, i : \mathbb{E}_{\xi_i \sim \mathcal{D}_i}\big[\mathbf{g}_i^{t,e}\big] = -\nabla F_i(\mathbf{w}_i^{t,e})]$. Therefore, the term $\mathcal{Q}_1$ above can be bounded as follows:

$$\mathcal{Q}_1 = \mathbb{E}_{t+1|t} \left\langle \nabla F(\mathbf{w}^t), \mathbf{w}^{t+1} - \mathbf{w}^t \right\rangle \tag{4.6}$$

$$= \left\langle \nabla F(\mathbf{w}^t), -\gamma \sum_{i=1}^{N} p_i \cdot \left(\sum_{e=0}^{E-1} \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left[g_i^{t,e} \right] \right) \right\rangle \tag{4.7}$$

$$= -\gamma E \cdot \left\langle \sum_{i=1}^{N} p_i \nabla F_i(\mathbf{w}^t), \sum_{i=1}^{N} \sum_{e=0}^{E-1} \frac{p_i}{E} \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left[\nabla f_i \left(\mathbf{w}_i^{t,e} \right) \right] \right\rangle \tag{4.8}$$

$$= -\frac{\gamma E}{2} \cdot \left\| \nabla F(\mathbf{w}^t) \right\|_2^2 - \frac{\gamma E}{2} \left\| \sum_{i=1}^{N} \sum_{e=0}^{E-1} \frac{p_i}{E} \nabla F_i \left(\mathbf{w}_i^{t,e} \right) \right\|_2^2 \tag{4.9}$$

$$+ \frac{\gamma E}{2} \left\| \sum_{i=1}^{N} \sum_{e=0}^{E-1} \frac{p_i}{E} \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left[\nabla f_i(\mathbf{w}^t) - \nabla f_i \left(\mathbf{w}_i^{t,e} \right) \right] \right\|_2^2 \tag{4.10}$$

$$\leq -\frac{\gamma E}{2} \cdot \left\| \nabla F(\mathbf{w}^t) \right\|_2^2 - \frac{\gamma}{2E} \left\| \sum_{i=1}^{N} \sum_{e=0}^{E-1} p_i \nabla F_i \left(\mathbf{w}_i^{t,e} \right) \right\|_2^2 \tag{4.11}$$

$$+ \frac{\gamma E}{2} \cdot \sum_{i=1}^{N} \sum_{e=0}^{E-1} \frac{p_i}{E} \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left[\left\| \nabla f_i(\mathbf{w}^t) - \nabla f_i \left(\mathbf{w}_i^{t,e} \right) \right\|_2^2 \right] \tag{4.12}$$

$$\leq -\frac{\gamma E}{2} \cdot \left\| \nabla F(\mathbf{w}^t) \right\|_2^2 - \frac{\gamma}{2E} \left\| \sum_{i=1}^{N} \sum_{e=0}^{E-1} p_i \nabla F_i \left(\mathbf{w}_i^{t,e} \right) \right\|_2^2 \tag{4.13}$$

$$+ \frac{\gamma L_s^2}{2} \cdot \sum_{i=1}^{N} \sum_{e=0}^{E-1} p_i \mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left[\left\| \mathbf{w}^t - \mathbf{w}_i^{t,e} \right\|_2^2 \right] \tag{4.14}$$

where the last equality follows from the fact that $\langle a, b \rangle = \frac{1}{2}a^2 + \frac{1}{2}b^2 - \frac{1}{2}(a-b)^2$ and the assumption we make in § 4.1 that $\mathbb{E}_{\xi_i \sim \mathcal{D}_i} \left[\nabla f_i(\mathbf{w}, \xi_i) \right] = \nabla F_i(\mathbf{w})$; the first inequality follows from Jensen's Inequality because $\sum_{i=1}^{N} \sum_{e=0}^{E-1} \frac{p_i}{E} = 1$; and the next inequality from the L_s-smoothness assumption.

Plugging Lemma 3 into the above bound for $\mathcal{Q}_1$, and using Lemma 4 to

bound $\mathcal{Q}_2$, we have that:

$$\mathbb{E}_{t+1|t}\left(F\left(\mathbf{w}^{t+1}\right)\right) - F\left(\mathbf{w}^{t}\right) \tag{4.15}$$

$$\leq -\frac{\gamma E}{2}\|\nabla F(\mathbf{w}^t)\|_2^2 + \frac{3\gamma^3 E^2 L_s^2}{2}(EL_c + \sigma^2) \tag{4.16}$$

$$+ \frac{L_s\gamma^2}{2}\cdot\mathbb{E}\left(\sum_{i\in\mathcal{C}^t}\frac{S^2}{C^2}p_i^2 + \sum_{i\in\mathcal{R}^t}\left(\frac{N-S}{K-C}\right)^2 p_i^2\right)E\sigma^2 \tag{4.17}$$

$$+ \frac{L_s\gamma^2}{2}E^2L_c^2\left(\sum_{i\in\mathcal{S}^t}\frac{S}{C}p_i^2 + \sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}\frac{N-S}{K-C}p_i^2\right) \tag{4.18}$$

$$- \left(\frac{\gamma}{2E} - \frac{L_s\gamma^2}{2}\right)\left\|\sum_{i=1}^{N}\sum_{e=0}^{E-1}p_i\nabla F_i\left(\mathbf{w}_i^{t,e}\right)\right\|_2^2 \tag{4.19}$$

4.3.3 Final Convergence Result

Let $\gamma \leq \frac{1}{EL_s}$. By averaging the above inequality over t from 1 to T, we have:

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{t+1|t}(F(\mathbf{w}^{t+1}) - F(\mathbf{w}^t)) \tag{4.20}$$

$$\leq -\frac{\gamma E}{2T}\|\nabla F(\mathbf{w}^t)\|_2^2 + \frac{3\gamma^3 E^2 L_s^2}{2}(EL_c + \sigma^2) \tag{4.21}$$

$$+ \frac{L_s\gamma^2 E\sigma^2}{2T}\sum_{t=1}^{T}\mathbb{E}\left(\sum_{i\in\mathcal{C}^t}\frac{S^2}{C^2}p_i^2 + \sum_{i\in\mathcal{R}^t}\left(\frac{N-S}{K-C}\right)^2 p_i^2\right) \tag{4.22}$$

$$+ \frac{L_s\gamma^2 E^2 L_c^2}{2T}\sum_{t=1}^{T}\left(\sum_{i\in\mathcal{S}^t}\frac{S}{C}p_i^2 + \sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}\frac{N-S}{K-C}p_i^2\right) \tag{4.23}$$

$$= -\frac{\gamma E}{2T}\|\nabla F(\mathbf{w}^t)\|_2^2 + \frac{3\gamma^3 E^2 L_s^2}{2}(EL_c + \sigma^2) \tag{4.24}$$

$$+ \frac{L_s\gamma^2 E(\sigma^2 + EL_c^2)}{2T}\sum_{t=1}^{T}\left(\sum_{i=1}^{N}\frac{S^2}{CN}p_i^2 + \sum_{i=1}^{N}\frac{(N-S)^2}{N(K-C)}p_i^2\right) \tag{4.25}$$

where the last equation follows that (i) a client in the sticky group and the non-sticky group with the probability of $\frac{S}{N}$ and $\frac{N-S}{N}$, respectively; (ii) a client is sampled from the sticky group and the non-sticky group with the

probability of $\frac{C}{S}$ and $\frac{K-C}{N-S}$, respectively. Therefore, the convergence rate is

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla F(\mathbf{w}^t)\|_2^2 \leq \frac{2(F(\mathbf{w}^1)-F_*)}{\gamma ET} + 3\gamma^2 E L_s^2(EL_c+\sigma^2) \tag{4.26}$$

$$+\frac{L_s\gamma(\sigma^2+EL_c^2)}{N}\left(\frac{S^2}{C}+\frac{(N-S)^2}{K-C}\right)\sum_{i=1}^{N}p_i^2 \tag{4.27}$$

By setting the learning rate as devised in Theorem 2, we can obtain the desired result.

4.3.4 Bounded Gap between two successive local updates.

Lemma 3. *Suppose, for all $i \in \{1,\ldots,N\}$, the local objective function F_i is L_c-continuous and L_s-smooth. Then, for all $e \in \{0,\ldots,E-1\}$, we have*

$$\mathbb{E}\|\mathbf{w}_i^{t,e}-\mathbf{w}^t\|_2^2 \leq 3E\left(E\gamma^2L_c^2+\gamma^2\sigma^2\right) \tag{4.28}$$

Proof. As we know, the recurrence formula for $\mathbf{w}_i^{t,e} = \mathbf{w}_i^{t,e-1} - \gamma g_i^{t,e-1}$. Through this relationship, we can bound for $\mathbb{E}\left\|\mathbf{w}_i^{t,e}-\mathbf{w}^t\right\|_2^2$,

$$\mathbb{E}\|\mathbf{w}_i^{t,e}-\mathbf{w}^t\|_2^2 = \mathbb{E}\|\mathbf{w}_i^{t,e-1}-\gamma g_i^{t,e-1}-\mathbf{w}^t\|_2^2 \tag{4.29}$$

$$\overset{(a)}{=} \mathbb{E}\|\mathbf{w}_i^{t,e-1}-\mathbf{w}^t-\gamma\nabla F_i(\mathbf{w}_i^{t,e-1})\|_2^2 \tag{4.30}$$

$$+\gamma^2\cdot\mathbb{E}\left\|g_i^{t,e-1}-\nabla F_i(\mathbf{w}_i^{t,e-1})\right\|_2^2 \tag{4.31}$$

$$\overset{(b)}{\leq}\left(1+\frac{1}{E-1}\right)\cdot\mathbb{E}\|\mathbf{w}_i^{t,e-1}-\mathbf{w}^t\|_2^2 \tag{4.32}$$

$$+E\gamma^2\cdot\mathbb{E}\|\nabla F_i(\mathbf{w}_i^{t,e-1})\|_2^2 \tag{4.33}$$

$$+\gamma^2\cdot\mathbb{E}\left\|g_i^{t,e-1}-\nabla F_i(\mathbf{w}_i^{t,e-1})\right\|_2^2 \tag{4.34}$$

$$\overset{(c)}{\leq}\left(1+\frac{1}{E-1}\right)\cdot\mathbb{E}\|\mathbf{w}_i^{t,e-1}-\mathbf{w}^t\|_2^2+E\gamma^2L_c^2+\gamma^2\sigma^2 \tag{4.35}$$

$$\leq\sum_{\varphi=0}^{e-1}\left(1+\frac{1}{E-1}\right)^{\varphi}\cdot(E\gamma^2L_c^2+\gamma^2\sigma^2) \tag{4.36}$$

$$\leq 3E\left(E\gamma^2L_c^2+\gamma^2\sigma^2\right) \tag{4.37}$$

In the above proof, equation (a) separates the mean and the variance, the first inequality (b) uses $(a+b)^2 \leq (1+\alpha)a^2+(1+\frac{1}{\alpha})b^2$, and the inequality (c) follows Assumptions 1 and 2. □

4.3.5 Bounded gap between two successive global models

Inspired by the proof of Theorem 2 in [44], we derive the following lemma to bound Q_2 accounting for GlueFL reweighted aggregation in Algorithm 2:

Lemma 4. *Suppose Assumption 1 and 2 hold. With Algorithm 2 by setting the weights $\nu_{i,s}^t = \frac{S}{C} p_i$ and $\nu_{i,r}^t = \frac{N-S}{K-C} p_i$ mentioned in Section 3.1, let $\alpha_i = p_i \sum_{e=0}^{E-1} \nabla F_i(w_i^{t,e})$, the bound for two successive models should be*

$$\mathbb{E}_{t+1|t} \left\| \mathbf{w}^{t+1} - \mathbf{w}^t \right\|_2^2 \tag{4.38}$$

$$\leq \gamma^2 E \sigma^2 \mathbb{E}_{t+1|t} \left(\sum_{i \in \mathcal{C}^t} \left(\frac{S}{C} p_i \right)^2 + \sum_{i \in \mathcal{R}^t} \left(\frac{N-S}{K-C} p_i \right)^2 \right) + \gamma^2 \mathbb{E}_{t+1|t} \left(\frac{S}{C} \sum_{i \in \mathcal{S}^t} p_i^2 E^2 L_c^2 + \frac{N-S}{K-C} \sum_{i \in \mathcal{N} \setminus \mathcal{S}^t} p_i^2 E^2 L_c^2 + \left\| \sum_{i=1}^{N} \alpha_i \right\|_2^2 \right) \tag{4.39}$$

Proof. As we know, the relationship between two successive models is

$$\mathbb{E}_{t+1|t} \left\| \mathbf{w}^{t+1} - \mathbf{w}^t \right\|_2^2 \tag{4.40}$$

$$= \gamma^2 \cdot \mathbb{E}_{t+1|t} \left\| \sum_{i \in \mathcal{C}^t} \nu_{i,s}^t \sum_{e=0}^{E-1} g_i^{t,e} + \sum_{i \in \mathcal{R}^t} \nu_{i,r}^t \sum_{e=0}^{E-1} g_i^{t,e} \right\|_2^2 \tag{4.41}$$

$$\leq \gamma^2 \mathbb{E}_{t+1|t} \left(\sum_{i \in \mathcal{C}^t} (\nu_{i,s}^t)^2 + \sum_{i \in \mathcal{R}^t} (\nu_{i,r}^t)^2 \right) \cdot E \sigma^2 \tag{4.42}$$

$$+ \gamma^2 \mathbb{E}_{t+1|t} \left\| \sum_{i \in \mathcal{C}^t} \nu_{i,s}^t \sum_{e=0}^{E-1} \nabla F_i(\mathbf{w}_i^{t,e}) + \sum_{i \in \mathcal{R}^t} \nu_{i,r}^t \sum_{e=0}^{E-1} \nabla F_i(\mathbf{w}_i^{t,e}) \right\|_2^2 \tag{4.43}$$

where the inequality is based on Lemma 2. Next, we ignore the coefficient and find the bound for the second term of Equation (4.43) by plain expanding the term as proposed in [44]: Let $\alpha_i = p_i \sum_{e=0}^{E-1} \nabla F_i(w_i^{t,e})$, and since $\nu_{i,s}^t =$

$\frac{S}{C}p_i$ and $\nu_{i,r}^t = \frac{N-S}{K-C}p_i$, we have

$$\mathbb{E}_{t+1|t}\left\|\sum_{i\in\mathcal{C}^t}\nu_{i,s}^t\sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_i^{t,e}) + \sum_{i\in\mathcal{R}^t}\nu_{i,r}^t\sum_{e=0}^{E-1}\nabla F_i(\mathbf{w}_i^{t,e})\right\|_2^2 \tag{4.44}$$

$$= \mathbb{E}_{t+1|t}\left\|\sum_{i\in\mathcal{C}^t}\frac{S}{C}\alpha_i + \sum_{i\in\mathcal{R}^t}\frac{N-S}{K-C}\alpha_i\right\|_2^2 \tag{4.45}$$

$$= \mathbb{E}_{t+1|t}\Bigg(\underbrace{\sum_{i\in\mathcal{C}^t}\left\|\frac{S}{C}\alpha_i\right\|_2^2}_{C \text{ terms}} + \underbrace{\sum_{i\in\mathcal{R}^t}\left\|\frac{N-S}{K-C}\alpha_i\right\|_2^2}_{(K-C) \text{ terms}} + \underbrace{\sum_{i\neq j,i,j\in\mathcal{C}^t}\left(\frac{S}{C}\right)^2\langle\alpha_i,\alpha_j\rangle}_{C(C-1) \text{ terms}}$$

$$+ \underbrace{\sum_{i\neq j,i,j\in\mathcal{R}^t}\left(\frac{N-S}{K-C}\right)^2\langle\alpha_i,\alpha_j\rangle}_{(K-C)(K-C-1) \text{ terms}} + 2\underbrace{\sum_{i\in\mathcal{C}^t,j\in\mathcal{R}^t}\left(\frac{S}{C}\right)\left(\frac{N-S}{K-C}\right)\langle\alpha_i,\alpha_j\rangle}_{C(K-C) \text{ terms}}\Bigg) \tag{4.46}$$

Before analyzing the bound of Equation (4.46), we provide the constant results for the following expectations:

$$\mathbb{E}\left\|\alpha_i\right\|_2^2 = \frac{1}{S}\sum_{s\in\mathcal{S}^t}\left\|\alpha_s\right\|_2^2, \text{for } i\in\mathcal{S}^t \tag{4.47}$$

$$\mathbb{E}\left\|\alpha_i\right\|_2^2 = \frac{1}{N-S}\sum_{r\in\mathcal{N}\setminus\mathcal{S}^t}\left\|\alpha_r\right\|_2^2, \text{for } i\in\mathcal{N}\setminus\mathcal{S}^t \tag{4.48}$$

$$\mathbb{E}\left\langle\alpha_i,\alpha_j\right\rangle = \frac{1}{S^2}\sum_{s_1,s_2\in\mathcal{S}^t}\left\langle\alpha_{s_1},\alpha_{s_2}\right\rangle, \text{for } i,j\in\mathcal{S}^t \tag{4.49}$$

$$\mathbb{E}\left\langle\alpha_i,\alpha_j\right\rangle = \frac{1}{(N-S)^2}\sum_{r_1,r_2\in\mathcal{N}\setminus\mathcal{S}^t}\left\langle\alpha_{r_1},\alpha_{r_2}\right\rangle, \text{for } i,j\in\mathcal{N}\setminus\mathcal{S}^t \tag{4.50}$$

$$\mathbb{E}\left\langle\alpha_i,\alpha_j\right\rangle = \frac{1}{S(N-S)}\sum_{s\in\mathcal{S}^t,r\in\mathcal{N}\setminus\mathcal{S}^t}\left\langle\alpha_s,\alpha_r\right\rangle, \text{for } i\in\mathcal{S}^t, j\in\mathcal{N}\setminus\mathcal{S}^t \tag{4.51}$$

Therefore, the bound of Equation (4.46) is analyzed as follows:

$$\mathbb{E}_{t+1|t}\left\|\sum_{i\in\mathcal{C}^t}\frac{S}{C}\alpha_i+\sum_{i\in\mathcal{R}^t}\frac{N-S}{K-C}\alpha_i\right\|_2^2 \tag{4.52}$$

$$=\mathbb{E}_{t+1|t}\left(\frac{C}{S}\cdot\left(\frac{S}{C}\right)^2\sum_{i\in\mathcal{S}^t}\|\alpha_i\|_2^2+\frac{K-C}{N-S}\left(\frac{N-S}{K-C}\right)^2\sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}\|\alpha_i\|_2^2\right. \tag{4.53}$$

$$+\frac{C(C-1)}{S^2}\left(\frac{S}{C}\right)^2\sum_{i,j\in\mathcal{S}^t}\langle\alpha_i,\alpha_j\rangle \tag{4.54}$$

$$+\frac{(K-C)(K-C-1)}{(N-S)^2}\left(\frac{N-S}{K-C}\right)^2\sum_{i,j\in\mathcal{N}\setminus\mathcal{S}^t}\langle\alpha_i,\alpha_j\rangle \tag{4.55}$$

$$\left.+2\frac{(K-C)C}{(N-S)S}\left(\frac{S}{C}\right)\left(\frac{N-S}{K-C}\right)\sum_{i\in\mathcal{S}^t,j\in\mathcal{N}\setminus\mathcal{S}^t}\langle\alpha_i,\alpha_j\rangle\right) \tag{4.56}$$

$$=\mathbb{E}_{t+1|t}\left(\frac{C}{S}\cdot\left(\frac{S}{C}\right)^2\sum_{i\in\mathcal{S}^t}\|\alpha_i\|_2^2+\frac{K-C}{N-S}\left(\frac{N-S}{K-C}\right)^2\sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}\|\alpha_i\|_2^2\right. \tag{4.57}$$

$$+\frac{C(C-1)}{S^2}\left(\frac{S}{C}\right)^2\left\|\sum_{i\in\mathcal{S}^t}\alpha_i\right\|_2^2 \tag{4.58}$$

$$+\frac{(K-C)(K-C-1)}{(N-S)^2}\left(\frac{N-S}{K-C}\right)^2\left\|\sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}\alpha_i\right\|_2^2 \tag{4.59}$$

$$\left.+2\frac{(K-C)C}{(N-S)S}\left(\frac{S}{C}\right)\left(\frac{N-S}{K-C}\right)\sum_{i\in\mathcal{S}^t,j\in\mathcal{N}\setminus\mathcal{S}^t}\langle\alpha_i,\alpha_j\rangle\right) \tag{4.60}$$

$$\leq\mathbb{E}_{t+1|t}\left(\frac{S}{C}\sum_{i\in\mathcal{S}^t}\|\alpha_i\|_2^2+\frac{N-S}{K-C}\sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}\|\alpha_i\|_2^2+\left\|\sum_{i=1}^N\alpha_i\right\|_2^2\right) \tag{4.61}$$

$$\leq\mathbb{E}_{t+1|t}\left(\frac{S}{C}\sum_{i\in\mathcal{S}^t}p_i^2E^2L_c^2+\frac{N-S}{K-C}\sum_{i\in\mathcal{N}\setminus\mathcal{S}^t}p_i^2E^2L_c^2+\left\|\sum_{i=1}^N\alpha_i\right\|_2^2\right) \tag{4.62}$$

where the first equation is due to the independent client sampling with replacement in both groups, and the last inequality follows Assumption 2. Therefore, with the result from Equation (4.62), we can obtain the desired result based on Equation (4.43). □

Chapter 5

Experimental Evaluation

We evaluate GlueFL across several datasets and network distributions. Our goal is to answer three questions:

1. What model accuracy does GlueFL achieve?
2. How does GlueFL impact bandwidth usage?
3. How quickly does the model converge with GlueFL?

5.1 Implementation

We build GlueFL on top of FedScale, a scalable and extensible open-source FL library [19]. We implemented sticky sampling in `ClientManager`. We also customized the `Aggregator` and `Client` to implement mask shifting. While mask shifting introduces extra overheads to transmit sparse vectors, we take care to minimize these overheads using coding strategies from previous work [32, 43]. For example, we communicate the sparse vector using absolute positions or bitmaps.

Aggregation for Batch Normalization layers A Batch Normalization (BN) layer contains five parameters:

- Trainable parameters: `weight`, `bias`
- Non-trainable summary statistics: `running_mean`, `running_var`, and `num_batches_tracked`

While GlueFL updates trainable parameters as all model parameters (Algorithm 3), non-trainable parameters need to be treated differently. We perform the aggregation of these non-trainable parameters $\mathbf{v}$ as follows:

$$\Delta_i^t \leftarrow \mathbf{v}_i^{t,E} - \mathbf{v}_i^{t,0} \tag{5.1}$$

$$\mathbf{v}^{t+1} \leftarrow \mathbf{v}^t + \frac{1}{K} \sum_{i \in \mathcal{K}} \Delta_i^t \tag{5.2}$$

where Δ_i^t represents the local change of $\mathbf{v}^t$ on client i in round t. Note that we do not perform re-weighting on Δ_i^t as this produces the best empirical results. This aggregation rule is consistent with the FedScale implementation [19].

5.2 Experimental Setup

We deployed GlueFL on a set of VMs in one data-center with a total of 14 NVIDIA Tesla V100 GPUs. To reproduce real-world heterogeneous client performance, we use FedScale's client behavior trace and the NDT dataset [25] to simulate the availability pattern and bandwidth capacity of clients, respectively. To mitigate stragglers and offline clients, FedScale introduces an `over-commitment (OC)` variable [3] which we set to 1.3 in all experiments. That is, we sample $1.3 \times K$ clients in each round and use the first K uploaded updates.

Datasets and Models We use three datasets: FEMNIST [6], OpenImage [18], Google Speech [39]. The first two datasets are frequently used for image classification and consist of 640K and 1.3M colored images, respectively. Google Speech is a dataset with 105K speech samples. We partition the data using FedScale's real-world non-iid client-data mapping [19] and remove those clients that have fewer than 22 samples as the default setting in FedScale. In total, we use $2,800$, $10,625$, and $2,066$ clients in our experiments, respectively. The models we use are ShuffleNet [46] and MobileNet [31] for both FEMNIST and OpenImage, and ResNet-34 [13] for Google Speech. We set the number of sampled clients $K = 30$, 100, and 30 for FEMNIST, OpenImage, and Google Speech, respectively.

Baselines We compare GlueFL with FedAvg [24], the most widely used FL algorithm with no model compression methods. We also compare GlueFL with STC [32] and APF [7], which are the state-of-the-art sparsification and parameter freezing strategies, respectively.

Metrics We measure the total data volume and total training time to address Q2 and Q3, respectively. We also analyze the downstream bandwidth and download time. For download time, we pick the slowest client in each round and sum up their download time. To address Q1, similar to Oort [20], we average the test accuracy over 5 rounds and report the results when the averaged accuracy first reaches the target accuracy.

Table 5.1: **D**ownstream transmission **V**olume (**DV**, in $\times 10^2$ GB) and **T**otal transmission **V**olume (**TV**) for training different models/datasets. We measure Top-1 accuracy for FEMNIST and Google Speech, and Top-5 accuracy for OpenImage [19]. We set the target accuracy to be the highest achievable accuracy by all approaches. Our target accuracies are comparable with previous work [19, 20]. The best results are in bold.

Dataset	# Clients	Target Acc.	Model	**FedAvg**	**STC**	**APF**	**GlueFL**
				DV (TV)	**DV (TV)**	**DV (TV)**	**DV (TV)**
FEMNIST	2,800	73.3%	ShuffleNet	2.6 (4.6)	2.6 (3.4)	2.3 (3.2)	**2.2 (3.1)**
			MobileNet	1.2 (2.1)	1.5 (1.9)	1.5 (2.0)	**0.9 (1.4)**
OpenImage	10,625	66.8%	ShuffleNet	25.2 (45.0)	33.9 (50.0)	27.1 (43.1)	**21.3(31.4)**
			MobileNet	17.4 (31.1)	16.7 (24.5)	20.3 (30.9)	**14.9(22.1)**
Google Speech	2,066	61.2%	ResNet-34	12.8 (23.0)	13.5 (18.5)	15.8 (21.9)	**7.2 (12.5)**

Table 5.2: **D**ownload **T**ime (**DT**, in hours) and **T**otal training **T**ime (**TT**) for training different models/datasets. The best results are in bold.

Dataset	# Clients	Target Acc.	Model	**FedAvg**	**STC**	**APF**	**GlueFL**
				DT (TT)	**DT (TT)**	**DT (TT)**	**DT (TT)**
FEMNIST	2,800	73.3%	ShuffleNet	2.7 (7.6)	2.7 (5.7)	2.3 (5.7)	**2.2 (5.3)**
			MobileNet	1.5 (4.6)	1.7 (3.9)	1.6 (4.5)	**0.8 (3.3)**
OpenImage	10,625	66.8%	ShuffleNet	11.2 (28.8)	14.8 (29.9)	12.3 (29.8)	**8.0 (19.2)**
			MobileNet	7.1 (22.4)	7.1 (19.1)	8.8 (21.0)	**5.8 (14.4)**
Google Speech	2,066	61.2%	ResNet-34	20.1 (60.9)	16.0 (42.3)	19.1 (54.1)	**12.1(27.8)**

Training Parameters Clients perform 10 local updates per round. We use PyTorch's SGD optimizer with a momentum factor of 0.9 for all tasks. For FEMNIST, OpenImage, and Google Speech, the initial learning rate is set to 0.01, 0.05, and 0.01, respectively, with a decay factor of 0.98 every 10 rounds. To obtain the best performance, we set the total mask ratio $q = 20\%$ for ShuffleNet, and $q = 30\%$ for MobileNet and ResNet-34 in STC. For APF, we set the threshold for effective perturbation, which reflects the compression ratio, to 0.1 for all tasks. The remaining STC and APF parameters are set to their optimal values [7, 32]. For GlueFL, the default sticky group parameters

are $S = 4K$ and $C = 4K/5$. For ShuffleNet, the default mask shifting parameters are $q = 20\%$ and $q_{shr} = 16\%$. For MobileNet and ResNet-34, we set $q = 30\%$ and $q_{shr} = 24\%$. We use $I = 10$ to regenerate the shared mask every 10 rounds. We choose these values as they produce the best performance across most tasks.

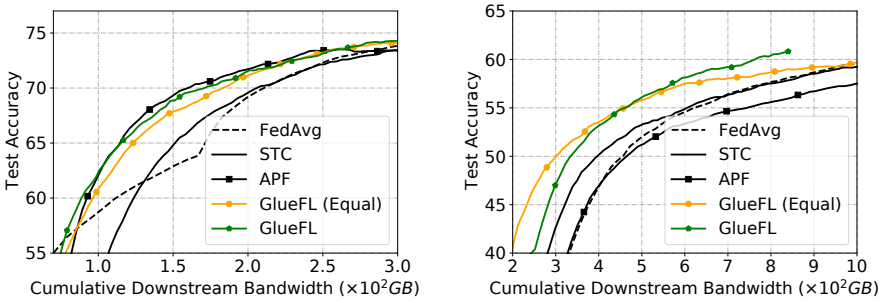
5.3 Performance Results

Communication costs Tables 5.1 and 5.2 list the data volume and training time for FedAvg (baseline), STC, APF, and GlueFL (our framework). It shows that STC and APF outperform FedAvg as they require less bandwidth to reach the target accuracy, reducing volume by 8% on average. However, STC and APF consume substantial downstream bandwidth. For example, when training MobileNet on FEMNIST, STC only takes 40 GB to upload gradients but uses 150 GB for downstream synchronization. GlueFL reduces downstream bandwidth (Table 5.1): for OpenImage, GlueFL provides a saving of 15% compared with FedAvg, while for Google Speech GlueFL saves 42%. We further compare the performance of GlueFL with STC and APF. In each case, while consuming nearly the same amount of upstream bandwidth (note upstream bandwidth volume = **TV-DV** in Table 5.1), GlueFL uses the least downstream bandwidth across all three datasets. For example, when training MobileNet on OpenImage, APF, STC, and GlueFL all consume around 900 GB to upload gradients. However GlueFL lowers download bandwidth by 11% and 26% as compared with STC and APF, respectively. This is because STC and APF do not bound the changes of masks in a communication round and the update size rapidly increases.

Wall-clock Time Table 5.2 indicates that downstream bandwidth is the bottleneck. For example, when training MobileNet on FEMNIST, FedAvg uses 32% of its total training time for model synchronization while STC uses 43%. GlueFL reduces total training time by reducing downstream bandwidth and saving download time, which speeds up the training by 15% and 26% as compared with STC and APF.

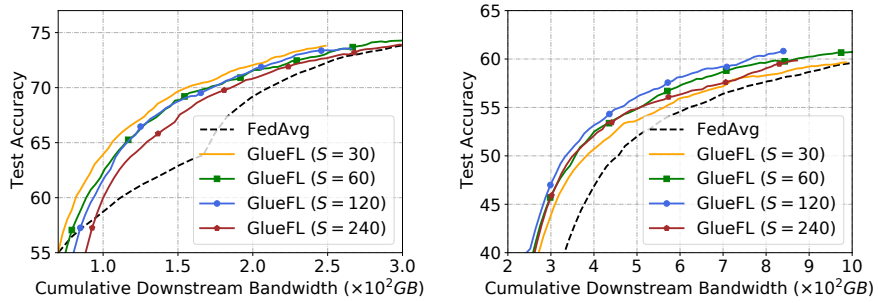
5.4 Sensitivity Analysis

We evaluate the influence of GlueFL parameters on training performance on FEMNIST with ShuffleNet and Google Speech with ResNet-34. Similar to §5.3, we use $K = 30$. When evaluating one parameter, we use defaults



(a) ShuffleNet (FEMNIST) (b) ResNet-34 (Google Speech)

Figure 5.1: Effect of aggregation weights $\nu_{i,s}^t$ and $\nu_{i,r}^t$: GlueFL (Equal) is biased (equal weights), while GlueFL is unbiased.

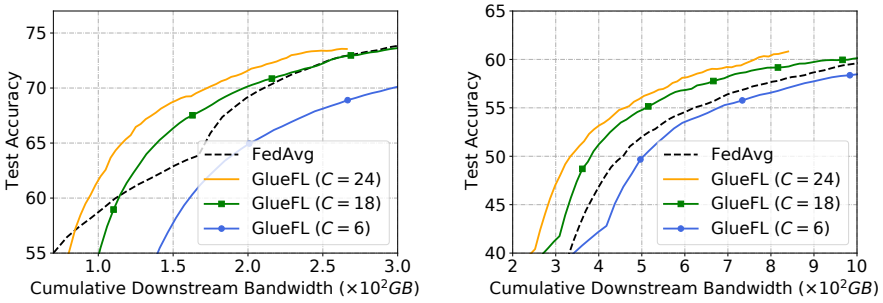


(a) ShuffleNet (FEMNIST) (b) ResNet-34 (Google Speech)

Figure 5.2: Effect of sticky group size S.

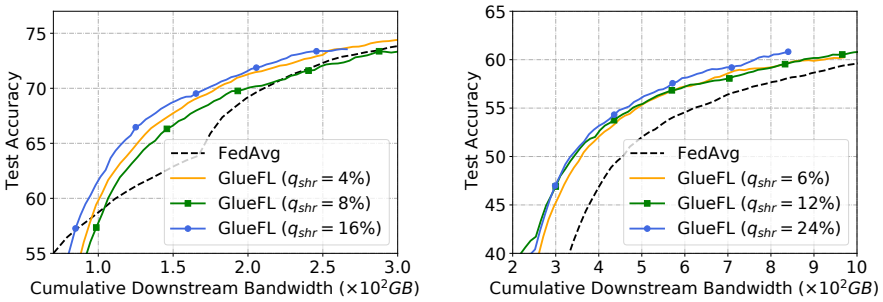
for the others (see §5.2). For each setting, we run GlueFL for 1,000 rounds and report the average test accuracy over 20 rounds with respect to the cumulative downstream bandwidth.

Effect of aggregation weights $\nu_{i,s}^t$ and $\nu_{i,r}^t$ Figure 5.1 demonstrates the impact of two settings of aggregation weights on training performance: equal (i.e., $\nu_{i,s}^t = \nu_{i,r}^t = 1/K$) and unbiased (see §3.1). Overall, unbiased aggregation weights lead to similar or better convergence speed for the same amount of cumulative downstream bandwidth usage. In the case of Google Speech, unbiased aggregation was able to achieve convergence while saving 41% of downstream bandwidth.



(a) ShuffleNet (FEMNIST) (b) ResNet-34 (Google Speech)

Figure 5.3: Effect of sticky sampling parameter C.



(a) ShuffleNet (FEMNIST) (b) ResNet-34 (Google Speech)

Figure 5.4: Effect of shared mask ratio q_{shr}.

Sticky sampling parameters S and C Figure 5.2 shows the impact of sticky group size S on training performance. Typically, a larger sticky group size means more diverse training data for the sticky clients and indirectly better accuracy at the cost of more communication. It follows that choosing an appropriately large S is important for optimizing performance. For instance, the $S = 120$ setting for Google Speech reached the target accuracy with almost 20% less downstream communication compared with $S = 60$. However, the same S is unable to help GlueFL achieve a speedup for FEMNIST.

Next, we evaluate the impact of the sticky sampling parameter C (Figure 5.3). C clients in the sticky group are sampled and $(K - C)$ clients are replaced by clients from the non-sticky group. Across $C = 6, 18,$ and 24 in

Figure 5.3, we do not observe a large improvement in accuracy for smaller C. By contrast, $C = 6$ adds 76% download bandwidth in each round as GlueFL is unable to capitalize on the savings from sticky sampling due to more new clients. This indicates that a large C does not harm accuracy and saves more bandwidth.

Mask shifting parameter q_{shr} Figure 5.4 shows the effect of the shared mask ratio q_{shr} on performance. On average, a higher value ($q_{shr} = 16\%$) does not cause accuracy to drop substantially and is preferable as GlueFL uses the least downstream bandwidth to reach the convergence accuracy of FedAvg. This is because GlueFL optimizes mask shifting with shared mask regeneration and error compensation.

5.5 Network Environment

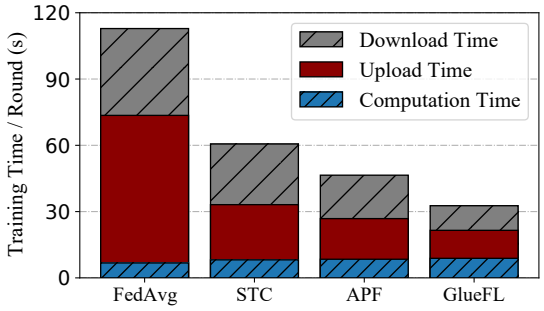
To further test our framework on high-throughput environments, we repeated the experiment in Tables 5.1 and 5.2 on commercial 5G [28] and Google Cloud [27] with the default settings for GlueFL (see §5.2). Figure 5.5 shows the total share of download, upload, and computation time for the three environments.

According to Figure 5.5(a), transmission time remains a bottleneck in the end-user edge devices environment as shown in Table 5.2. We attribute this to low-bandwidth clients. The ratio of download to upload time increases as we introduce compression. Since clients usually download faster than upload [1, 25]: new clients in FedAvg spend 70% more time uploading than downloading the same-sized update. However, for STC and APF, download time takes on average 8% longer than upload, confirming the discussion in §2.3. To address this limitation, GlueFL saves downstream bandwidth and reduces download time by at least 42% as compared with other approaches. This is because clients in the sticky group are required to download less updates and are therefore less likely to become stragglers.

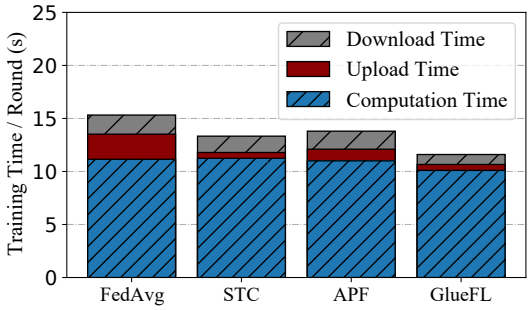
In 5G and intra-datacenter networks, computation dominates the per-round training time. Yet, straggler clients still exist and they ultimately determine the end-to-end training time.

5.6 Ablation Study

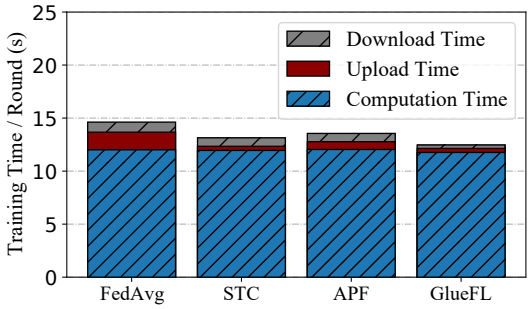
We described two optimization techniques in §3.3: shared mask regeneration and error-compensation. The first technique regenerates the shared mask



(a) End-user device network



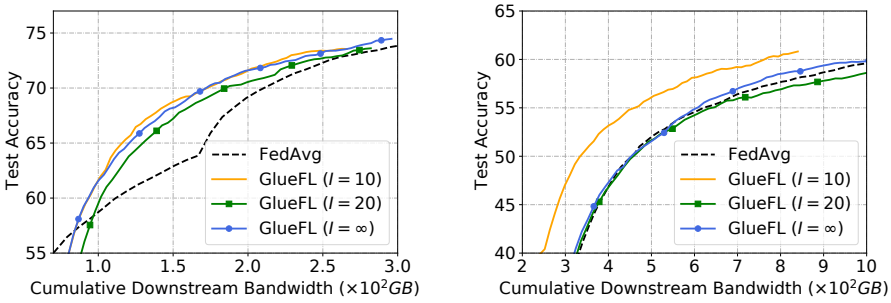
(b) Commercial 5G network



(c) Google Cloud datacenter network

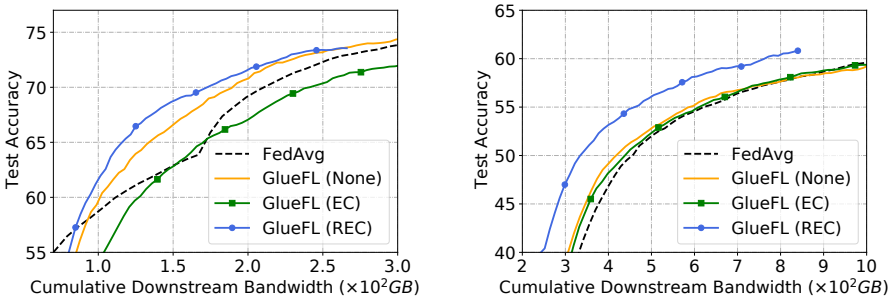
Figure 5.5: Average share of time spent per round downloading (grey), uploading (red) and computing (blue).

M^t every I rounds and the second technique adds a re-scaled compensation



(a) ShuffleNet (FEMNIST) (b) ResNet-34 (Google Speech)

Figure 5.6: Effect of shared mask regeneration.



(a) ShuffleNet (FEMNIST) (b) ResNet-34 (Google Speech)

Figure 5.7: Effect of error-compensation.

vector $h_i^{\varphi(t)}$ to local updates Δ_i^t. In this section, we conduct ablation studies to evaluate the effect of these techniques.

We run GlueFL on FEMNIST with ShuffleNet and Google Speech with ResNet-34. In each round, the server samples 30 clients out of 2,800 clients (for FEMNIST) and 30 clients out of 2,066 clients (for Google Speech). For each experiment, we run 1,000 rounds and measure the downstream bandwidth and test accuracy. While GlueFL consists of both sticky sampling and mask shifting, we only change the corresponding part in mask shifting and keep other training settings the same as §5.2.

Shared Mask Regeneration As described in §3.3, we set $q_{shr} = 0$ and regenerate the shared mask as $M^t \leftarrow top_{q_{shr}}(\tilde{\Delta}^t_{uni})$ every I rounds. A larger

I value indicates that M^t will be regenerated less frequently. We do not regenerate M^t when $I = \infty$.

In Figures 5.6(a) and 5.6(b), we plot results for three I values: 10, 20, ∞. Both figures show that setting $I = 10$ achieves the best overall performance, saving around 22% downstream bandwidth at the target accuracy for Google Speech. The impact of I on FEMNIST training in Figure 5.6(a) is less pronounced but the $I = 10$ setting still has the best accuracy. Thus, in practice, we need to set an appropriate I value (e.g., 10) to avoid a drop in accuracy.

Error-Compensation In §3.3 we noted that error compensation can be used to accelerate convergence when applying compression methods in FL training. GlueFL re-scales the compensation vector $h_i^{\varphi(t)}$, following Equation (3.5), to make it compatible with sticky sampling. In this section, we report on experiments for three error compensation settings: no compensation (**None**), compensation without re-scaling (**EC**), compensation with re-scaling (**REC**). The convergence results are shown in Figures 5.7(a) and 5.7(b). Both figures show that removing re-scaling from error compensation immediately breaks GlueFL and harms the convergence performance. This demonstrates that it is necessary to apply re-scaling with error compensation.

5.7 Availability and Stragglers

In §5.2 we discussed a default value of 1.3 for `over-commitment`. This means that GlueFL will sample $0.3 \times K$ additional clients to mitigate stragglers and clients that might become unavailable (e.g., go offline). In this section, we explore different values and strategies in over-commitment for GlueFL.

In GlueFL's default setting, the over-commitment applies to both sticky group $\mathcal{S}$ and non-sticky group $\mathcal{N} \setminus \mathcal{S}$. The server will sample $0.3 \times K \times (C/K)$ and $0.3 \times K \times (1 - (C/K))$ additional clients from $\mathcal{S}$ and $\mathcal{N} \setminus \mathcal{S}$, respectively. However, as GlueFL only includes the fastest $K - C$ clients in all sampled non-sticky clients to $\mathcal{S}$ in each round, clients in $\mathcal{S}$ are less likely to become stragglers. It follows that we can improve the over-commitment strategy by sampling fewer additional clients in $\mathcal{S}$ while sampling more additional clients from $\mathcal{N} \setminus \mathcal{S}$.

Table 5.3(a) presents the results from using four over-commitment strategies for training ShuffleNet on FEMNIST. Similar to previous tasks: we select 30 clients out of 2,800 clients in each round and we choose another 9 (i.e.,

Table 5.3: **D**ownstream transmission **V**olume (**DV**, in $\times 10^2$ GB), **D**ownload **T**ime (**DT**, in hours), **T**otal transmission **V**olume (**TV**) and **T**otal training **T**ime (**TT**) for training ShuffleNet on FEMNIST with different over-commitment (OC) settings.

(a) Results of different over-commitment strategies with a constant over-commit value (1.3)

OC Strategy	$(S : N \setminus S)$	**DV**	**TV**	**DT**	**TT**
10%	1 : 8	2.1	3.1	0.6	2.7
30%	3 : 6	2.2	3.0	0.9	3.1
50%	5 : 4	2.1	2.9	1.3	3.8
C/K (Default)	7 : 2	2.2	3.1	2.2	5.3

(b) Results for different over-commitment values with a constant strategy (row 1 in Table 5.3(a))

OC Value	**DV**	**TV**	**DT**	**TT**
1.0	1.5	2.3	32.0	67.8
1.1	2.2	3.1	3.5	10.7
1.2	2.2	3.0	1.0	3.9
1.3	2.1	3.1	0.6	2.7
1.4	2.9	3.7	0.5	2.6
1.5	3.1	4.0	0.5	2.4

0.3×30) clients for over-commitment. We report transmission volume and training time when the model reaches the target test accuracy of 73.3%. In the table, the OC strategy row of 10% means that 1 (i.e., $0.3 \times 30 \times 10\%$), and 8 (i.e., $0.3 \times 30 \times (1 - 10\%)$) additional clients are sampled from $\mathcal{S}$ and $\mathcal{N} \setminus \mathcal{S}$, respectively. The results show that by choosing fewer additional clients from the sticky group, GlueFL consumes less training time without increasing the downstream bandwidth volume.

Next, we use the best setting of 10% (from Table 5.3(a)) to evaluate different OC values. Table 5.3(b) shows the results for OC values of 1.0 to 1.5. With increasing OC values, we find that training time decreases faster than downstream volume increases. As an example, when OC value is changed from 1.0 to 1.3, training time is decreased by 96% and downstream volume increases by 40%. However, increasing the OC value from 1.3 to 1.5 only reduces 11% training time while consuming 47% more downstream volume. In practice, one should set the OC value carefully to balance the trade-off between bandwidth and training time.

Chapter 6

Related Work

The synchronization bottleneck is an established problem in FL. Existing solutions fall into roughly two categories: (1) use *client sampling* to constrain the number of clients in each round; and, (2) *compress model data* with strategies like sparsification and parameter freezing.

Client sampling. FedAvg proposed a uniform sampling of clients to participate in each round. Uniform sampling has been shown to be biased, and multinomial distribution (MD) sampling was proposed to address this issue [21]. Clustered sampling [9] reduced the variance of client update aggregation by improving client representation. Oort [20] introduced a practical client selection algorithm, which considers both data utility and clients speed.

Sparsification. The idea of sparsification is to send only the most informative gradients. Gaia [14] transfers gradients whose absolute or relative values are larger than a given threshold. Stich et al. [34] proposed *Top-K* that, given a compression ratio, selects a fraction of gradients based on their absolute values to meet the ratio. STC [32] extended Top-K to FL training and also uses server-side compression.

Parameter freezing. Parameter freezing reduces bandwidth by freezing the gradients that converged. Brock et al. [5] proposed FreezeOut, which gradually froze the first few layers of a deep neural network that were observed to converge first. However, it has a coarse layer-based granularity and it degrades accuracy. APF [7] improves on FreezeOut by freezing at a fine granularity and achieves a communication speed-up while preserving model convergence.

Our goal with GlueFL is to coherently combine client sampling with model compression. To our knowledge, we are the first to propose a combination that is unbiased, achieves high accuracy, and lowers downstream bandwidth usage.

Chapter 7

Conclusions

We proposed **GlueFL**, a framework to optimize downstream bandwidth in cross-device FL. GlueFL uses *sticky sampling* for client selection and *mask shifting* for model compression to mitigate the low download bandwidth of FL clients. We also provide a theoretical convergence guarantee for GlueFL. In comparison with FedAvg, GlueFL achieves similar accuracy while decreasing total training time by 36% and uses 22% less downstream bandwidth. GlueFL also outperforms STC [32] and APF [7].

Bibliography

[1] Speedtest global index. `https://www.speedtest.net/global-index`, 2022.

[2] Debraj Basu, Deepesh Data, Can Karakus, and Suhas Diggavi. Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[3] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. In *Proceedings of Machine Learning and Systems (MLSys)*, 2019.

[4] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[5] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Freezeout: Accelerate Training by Progressively Freezing Layers. In *NIPS Workshop on Optimization for Machine Learning (OPTML)*, 2017.

[6] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečnỳ, H Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings. *arXiv preprint arXiv:1812.01097*, 2018.

[7] Chen Chen, Hong Xu, Wei Wang, Baochun Li, Bo Li, Li Chen, and Gong Zhang. Communication-Efficient Federated Learning With Adaptive Parameter Freezing. In *International Conference on Distributed Computing Systems (ICDCS)*, 2021.

[8] Wenlin Chen, Samuel Horvath, and Peter Richtarik. Optimal Client Sampling for Federated Learning. *arXiv preprint arXiv:2010.13723*, 2020.

[9] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered Sampling: Low-Variance and Improved Representativity for Clients Selection in Federated Learning. In *International Conference on Machine Learning (ICML)*, 2021.

[10] Pengchao Han, Shiqiang Wang, and Kin K Leung. Adaptive Gradient Sparsification for Efficient Federated Learning: An Online Learning Approach. In *International Conference on Distributed Computing Systems (ICDCS)*, 2020.

[11] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated Learning for Mobile Keyboard Prediction. *arXiv preprint arXiv:1811.03604*, 2018.

[12] Florian Hartmann, Sunah Suh, Arkadiusz Komarzewski, Tim D Smith, and Ilana Segall. Federated Learning for Ranking Browser History Suggestions. *arXiv preprint arXiv:1911.11807*, 2019.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017.

[15] Peng Jiang and Gagan Agrawal. A Linear Speedup Analysis of Distributed Deep Learning With Sparse and Quantized Communication. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[16] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[17] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *International Conference on Machine Learning (ICML)*, 2020.

[18] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *IJCV*, 2020.

[19] Fan Lai, Yinwei Dai, Sanjay S. Singapuram, Jiachen Liu, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. FedScale: Benchmarking Model and System Performance of Federated Learning at Scale. In *International Conference on Machine Learning (ICML)*, 2022.

[20] Fan Lai, Xiangfeng Zhu, Harsha V. Madhyastha, and Mosharaf Chowdhury. Oort: Efficient Federated Learning via Guided Participant Selection. In *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2021.

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. In *Proceedings of Machine Learning and Systems (MLSys)*, 2020.

[22] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the Convergence of FedAvg on Non-iid Data. In *International Conference on Learning Representations (ICLR)*, 2020.

[23] Bing Luo, Wenli Xiao, Shiqiang Wang, Jianwei Huang, and Leandros Tassiulas. Tackling System and Statistical Heterogeneity for Federated Learning with Adaptive Client Sampling. In *IEEE Conference on Computer Communications (INFOCOM)*, 2022.

[24] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks From Decentralized Data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[25] Measurement Lab. The M-Lab NDT data set. `https://measurementlab.net/tests/ndt`, (2022-06-01 – 2022-07-01).

[26] Aritra Mitra, Rayana Jaafar, George J Pappas, and Hamed Hassani. Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[27] Ricky KP Mok, Hongyu Zou, Rui Yang, Tom Koch, Ethan Katz-Bassett, and Kimberly C Claffy. Measuring the Network Performance of Google Cloud Platform. In *Internet Measurement Conference (IMC)*, 2021.

[28] Arvind Narayanan, Xumiao Zhang, Ruiyang Zhu, Ahmad Hassan, Shuowei Jin, Xiao Zhu, Xiaoxuan Zhang, Denis Rybkin, Zhengxuan Yang, Zhuoqing Morley Mao, et al. A Variegated Look at 5G in the Wild: Performance, Power, and Qoe Implications. In *ACM Special Interest Group on Data Communication (SIGCOMM)*, 2021.

[29] Xun Qian, Peter Richtárik, and Tong Zhang. Error Compensated Distributed SGD Can Be Accelerated. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[30] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. FedPAQ: A Communication-Efficient Federated Learning Method With Periodic Averaging and Quantization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Robust and Communication-Efficient Federated Learning From Non-iid Data. *IEEE transactions on neural networks and learning systems*, 31(9):3400–3413, 2019.

[33] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-Bit Stochastic Gradient Descent and Its Application to Data-Parallel Distributed Training of Speech DNNs. In *International Speech Communication Association (ISCA)*, 2014.

[34] Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified SGD with Memory. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[35] Hanlin Tang, Chen Yu, Xiangru Lian, Tong Zhang, and Ji Liu. DoubleSqueeze: Parallel Stochastic Gradient Descent With Double-Pass Error-Compensated Compression. In *International Conference on Machine Learning (ICML)*, 2019.

[36] Shay Vargaftik, Ran Ben Basat, Amit Portnoy, Gal Mendelson, Yaniv Ben Itzhak, and Michael Mitzenmacher. Eden: Communication-Efficient and Robust Distributed Mean Estimation for Federated Learning. In *International Conference on Machine Learning (ICML)*, 2022.

[37] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[38] Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient Sparsification for Communication-Efficient Distributed Optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[39] Pete Warden. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[40] Feijie Wu, Song Guo, Haozhao Wang, Zhihao Qu, Haobo Zhang, Jie Zhang, and Ziming Liu. From Deterioration to Acceleration: A Calibration Approach to Rehabilitating Step Asynchronism in Federated Optimization. *arXiv preprint arXiv:2112.09355*, 2021.

[41] Feijie Wu, Shiqi He, Song Guo, Zhihao Qu, Haozhao Wang, Weihua Zhuang, and Jie Zhang. Sign Bit is Enough: A Learning Synchronization Framework for Multi-Hop All-Reduce with Ultimate Compression. In *ACM/IEEE Design Automation Conference (DAC)*, 2022.

[42] Jiaxiang Wu, Weidong Huang, Junzhou Huang, and Tong Zhang. Error Compensated Quantized SGD and Its Applications to Large-Scale Distributed Optimization. In *International Conference on Machine Learning (ICML)*, 2018.

[43] Hang Xu, Chen-Yu Ho, Ahmed M Abdelmoniem, Aritra Dutta, El Houcine Bergou, Konstantinos Karatsenidis, Marco Canini, and Panos Kalnis. Grace: A Compressed Communication Framework for Distributed Machine Learning. In *International Conference on Distributed Computing Systems (ICDCS)*, 2021.

[44] Haibo Yang, Minghong Fang, and Jia Liu. Achieving linear speedup with partial worker participation in non-iid federated learning. *arXiv preprint arXiv:2101.11203*, 2021.

[45] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied Federated Learning: Improving Google Keyboard Query Suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[46] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.