FedSampling: A Better Sampling Strategy for Federated Learning

Tao Qi[1] , **Fangzhao Wu**[2*] , **Lingjuan Lyu**[3] , **Yongfeng Huang**[1,4,5] and **Xing Xie**[2]
[1]Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
[2]Microsoft Research Asia, Beijing 100080, China
[3]Sony AI, 1-7-1 Konan Minato-ku Tokyo 108-0075, Japan
[4]Zhongguancun Laboratory, Beijing 100094, China
[5] Institute for Precision Medicine of Tsinghua University, Beijing 102218, China
{taoqi.qt,wufangzhao}@gmail.com , Lingjuan.Lv@sony.com ,
yfhuang@tsinghua.edu.cn , xingx@microsoft.com

Abstract

Federated learning (FL) is an important technique for learning models from decentralized data in a privacy-preserving way. Existing FL methods usually uniformly sample clients for local model learning in each round. However, different clients may have significantly different data sizes, and the clients with more data cannot have more opportunities to contribute to model training, which may lead to inferior performance. In this paper, instead of client uniform sampling, we propose a novel data uniform sampling strategy for federated learning (*FedSampling*), which can effectively improve the performance of federated learning especially when client data size distribution is highly imbalanced across clients. In each federated learning round, local data on each client is randomly sampled for local model learning according to a probability based on the server desired sample size and the total sample size on all available clients. Since the data size on each client is privacy-sensitive, we propose a privacy-preserving way to estimate the total sample size with a differential privacy guarantee. Experiments on four benchmark datasets show that *FedSampling* can effectively improve the performance of federated learning.

1 Introduction

Federated learning aims to utilize decentralized data to train machine learning models [McMahan *et al.*, 2017], and has become a popular privacy-preserving machine learning paradigm [Dayan *et al.*, 2021; Bai *et al.*, 2021; Jiang *et al.*, 2021; Rothchild *et al.*, 2020]. The mainstream federated learning framework [Reddi *et al.*, 2020; Yang *et al.*, 2018; Sun *et al.*, 2021] usually uniformly samples some clients for local model training in each round. However, sizes of samples on different clients may largely differ [Fraboni *et al.*, 2021b]. The uniform client sampling may prevent the effective exploitation of the clients with more data for model training and lead to a suboptimal performance [Fraboni *et al.*, 2021b]. For instance, consider a scenario with 3 clients (Fig. 1), in which the client c_0, c_1 and c_2 keep 2, 1 and 1 samples respectively, and we need to select a client for training in each round. For uniform client sampling methods, although the sampling probability of each client is uniform and identical to $\frac{1}{3}$, the weights of samples in different clients for model updating are biased. For the client c_0, two samples in it will be used for training at the same time and thus their weights are identical to $\frac{1}{2}$, while the weights of samples in the client c_1 and c_2 are identical to 1. Thus, the model training may pay more attention to samples in the client c_1 and c_2, which leads to a biased data exploitation.

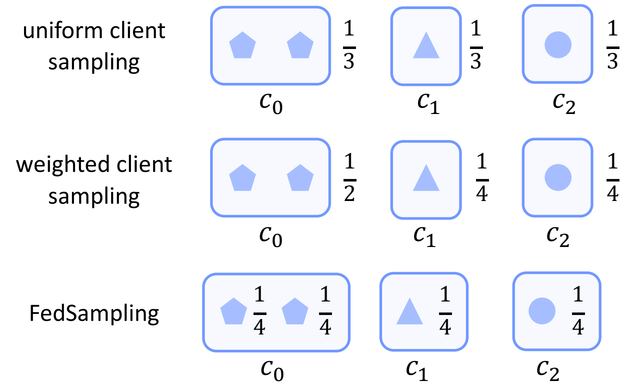

Figure 1: Toy example of different sampling methods, where the numbers indicate sampling probabilities. Both uniform and weighted client sampling are biased in data exploitation.

To address this issue, some recent methods have enabled the server to track the local sample size in each client to perform weighted client sampling [Li *et al.*, 2019; Wang *et al.*, 2020; Fraboni *et al.*, 2021a]. In these methods, the sampling probability of a client is the ratio of its local sample size to the sample size of all clients. However, these methods usually over-emphasize clients with more data, thus fail to satisfy an unbiased exploitation of samples, leading to sub-optimal performance. In the example of Fig. 1, for weighted client sampling methods, the sampling probabilities of data in the client c_0 are identical to $\frac{1}{2}$ while the sampling probabilities of data

[*]The corresponding author.

in the client c_1 and c_2 are identical to $\frac{1}{4}$. Thus, half of the total samples in the client c_0 will be sampled for training more frequently than the remaining half of the total samples, which is still biased in data exploitation. Besides, in some certain scenarios (e.g., financial transactions), the local sample size represents the frequency of some privacy-sensitive behaviors. Thus, allowing the server to track local sample sizes may also arouse privacy concerns [Gerber *et al.*, 2018].

In short, existing client-level sampling methods are usually biased in the data exploitation due to the dependent or non-identical data sampling. In this paper, we propose a uniform data sampling framework (named *FedSampling*) to improve the effectiveness of federated learning. At each training round of *FedSampling*, each sample in each client is independently sampled with an identical probability, where the probability is the ratio of the server desired sample size to the total sample size in all available clients. Considering the privacy of local sample size in each client, we propose a privacy-preserving method to estimate the total sample size based on the local differential privacy (LDP) technique. Each client randomly chooses to send the server a true or a randomly-generated local sample size to protect personal privacy, and the server can estimate an unbiased total sample size from the randomized responses. Besides, the analysis on the utility and privacy of *FedSampling* is also provided. We conduct extensive experiments on four benchmark datasets across different domains. Results show that *FedSampling* can outperform many recent FL methods, especially under imbalanced data size distribution and non-IID data distribution, and meanwhile achieve an effective trade-off between utility and privacy.

2 Related Work

Due to the importance of user privacy, privacy-persevering machine learning techniques have attracted increasing attention [Avdiukhin and Kasiviswanathan, 2021; Achituve *et al.*, 2021; Jumper *et al.*, 2021]. Federated learning can utilize massive private data distributed on local clients to benefit the training of a shared ML model [Hamer *et al.*, 2020; Wang *et al.*, 2021; Li *et al.*, 2020], and thereby has become a popular privacy-aware machine learning framework [Dayan *et al.*, 2021; Yoon *et al.*, 2021; Huang *et al.*, 2021; Bai *et al.*, 2021]. Existing methods usually follow a similar paradigm, where a server uniformly samples some clients and collects their local model parameters to update a global model, iteratively [Liang *et al.*, 2021; Yuan *et al.*, 2021; Fraboni *et al.*, 2021a]. For example, McMahan *et al.* [2017] proposed to locally train models on clients for multiple steps, and further average local model parameters from sampled clients to update the global model. Reddi *et al.* [2020] proposed a federated version of the Adam optimization algorithm to smooth the model training, where the current local parameters are combined with historical local parameters to learn the global model. However, the sample size of different clients may be highly imbalanced, and clients with more data cannot be sampled more frequently to contribute more to the model training in these methods, which usually results in a sub-optimal performance.

Some methods explored weighted client sampling to handle this problem [Li *et al.*, 2019; Wang *et al.*, 2020; Fraboni *et al.*, 2021a]. For example, Li *et al.* [2019] proposed to first sample clients for local model training based on the ratios between their local samples and the total samples in all clients, then average local model updates in the server to update the global model. However, client-level weighting usually overemphasizes the clients with more data and also cannot achieve a uniform data exploitation. Besides, these methods need the server to track the local sample size of each client, which may be impracticable in some scenarios due to privacy concerns. Different from these methods, we propose a data-level uniform sampling strategy, which can achieve a similar data exploitation like centralized learning to improve the performance of federated model training.

3 FedSampling

3.1 Problem Formulation

In our work, we assume that there are H available clients that can participate in the federated learning, and the set of available clients is denoted as $\mathcal{C}$. Each client c locally keeps a private dataset $\mathcal{D}_c$ and never shares it with the outside, where $\mathcal{D}_c = \{s_i | i = 1, 2, ..., |\mathcal{D}_c|\}$ and s_i denotes the i-th local sample. Moreover, the local sample size $|\mathcal{D}_c|$ of each client c is also assumed to be privacy-sensitive and cannot be disclosed. Besides, there is a server that takes charge of maintaining an ML model and scheduling the workflow of the federated model training. The core problem of this work is how to effectively sample decentralized data for model training to improve the effectiveness of federated learning.

3.2 Uniform Data Sampling

Uniform data sampling has the potential to improve the effectiveness of federated learning since it has a similar data exploitation pattern like centralized learning. Intuitively, by allowing the server to track the local sample size of each client, we can allocate quotas for each client according to the size of its local samples to achieve a uniform data sampling. However, in many scenarios (e.g., medicine and financial transactions) the size of local data is privacy-sensitive [Gerber *et al.*, 2018], and thereby cannot be tracked by the server. For instance, the size of medical records in a client represents the frequency of medical activities, which can be highly privacy-sensitive for many users. In our work, we propose a unified method that can uniformly and independently sample decentralized data for federated training without the collection of local sample sizes.

Without loss of generalization, we assume the server needs K samples to collaboratively update the model in a training round. Besides, we assume the server can obtain an estimated number $\widetilde{N}$ of total samples in all available clients[1]. In *FedSampling*, the server first broadcasts K and $\widetilde{N}$ to each client and then each client can locally sample local data for training according to its local sample size and the server desire. Take a client c as example, we first independently draw a random variable z_i^c from a Bernoulli distribution $\mathcal{B}(\frac{K}{\widetilde{N}})$ for

[1] We will introduce how we obtain $\widetilde{N}$ in the next section.

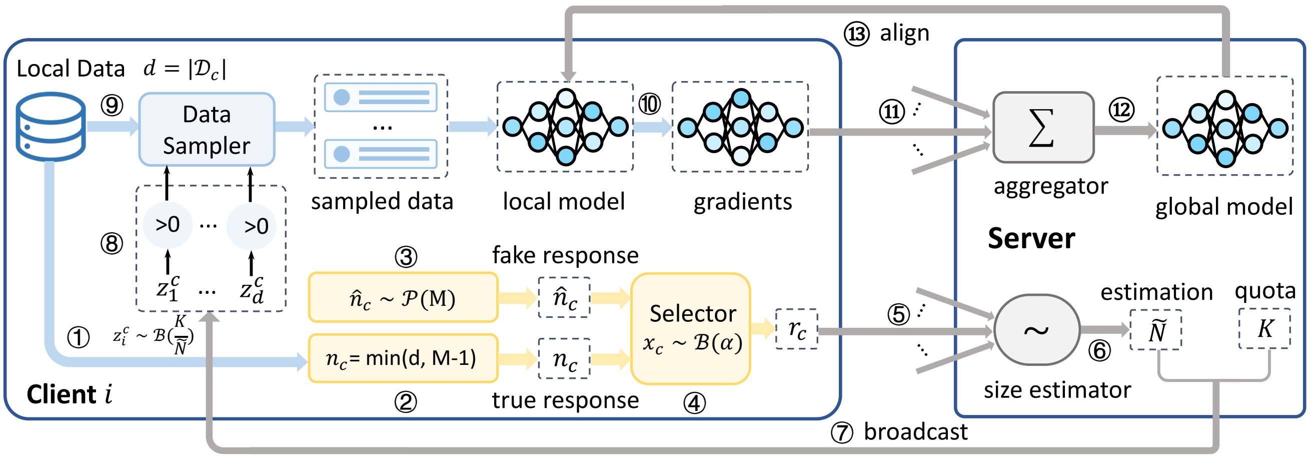

Figure 2: The framework of *FedSampling*.

each local sample in $\mathcal{D}_c$, where the ratio $\frac{K}{\tilde{N}}$ is the probability of assigning 1 to z_i^c, which means the i-th local sample s_i^c will be sampled for local training. In this way, each sample in each client can be independently sampled for training with identical probabilities. In addition, the expected number of samples for participating in this training round is $N\mathbb{E}[\frac{K}{\tilde{N}}]$, which can asymptotically converge to K and meet the server demand (the convergence is discussed in the following section). After the client c obtains a set $\mathcal{S}_c$ of locally sampled data, the client c will employ the current model Θ to calculate model gradients $\mathbf{g}_s$ on each sample $s \in \mathcal{S}_c$, and build the local model update based on normalized local gradients: $\mathbf{G}_c = \frac{1}{K}\sum_{s\in\mathcal{S}_c}\mathbf{g}_s$. Then the local update is uploaded to the server for the global model updating.

3.3 Privacy-Preserving Ratio Estimation

Next, we will introduce how to estimate the ratio of the server desired sample number K to the total sample number N in a privacy-preserving way. Intuitively, we may bypass this problem if we sample data based on a fixed ratio r (e.g., 1%) instead of a total sample number-aware ratio $\frac{K}{N}$. However, this will arise new challenges that need to be carefully addressed. Specifically, in this naive method, the local update $\mathbf{G}_c$ in the client c cannot be locally normalized by the number of sampled data in this round due to the lack of knowledge of it (i.e., $r \times N$). The client c can only obtain a local model update like mainstream FL frameworks by averaging local gradients: $\mathbf{G}_c = \frac{1}{|\mathcal{S}_c|}\sum_{s\in\mathcal{S}_c}\mathbf{g}_s$. On one hand, to obtain an unbiased global model updating, the server in this naive method needs to employ the number of locally sampled data in each client to weight the aggregation of the corresponding local updates. However, this will cause privacy leakage due to the disclosure of the number of locally sampled data in each client. On the other hand, the naive method can uniformly average the local model updates to avoid the potential privacy leakage. However, this will lead to a biased model update and result in a sub-optimal performance. Thus, the naive method mentioned above is not optimal for uniform data sampling.

Next, we will introduce a carefully designed method to estimate this ratio in a privacy-preserving way. Specifically, the server first queries clients for their local sample sizes. When a client c receives the query, it will first generate a true response n_c by clipping the local sample size $|\mathcal{D}_c|$ via a size threshold M: $n_c = \min(|\mathcal{D}_c|, M-1)$. Then the client c will generate a fake response by drawing a random variable $\hat{n}_c$ from a uniform multinomial distribution: $\hat{n}_c \sim \mathcal{P}(M)$, where we randomly select an integer from 1 to $M-1$ with identical probabilities and assign it to $\hat{n}_c$. Furthermore, the client c will draw a random variable x_c from a Bernoulli distribution for the response selection:

$$r_c = x_c n_c + (1-x_c)\hat{n}_c, \quad x_c \sim \mathcal{B}(\alpha), \tag{1}$$

where α and $1-\alpha$ denotes the probability of assigning 1 and 0 to x_c respectively, and r_c is the selected response. Under the protection of this method, it is difficult for the server to obtain the accurate knowledge on the local sample size of any client. The privacy protection ability of our privacy-preserving ratio estimation method can satisfy the ϵ-LDP, which will be discussed in the following section. Furthermore, after receiving responses from clients, the server further aggregates them to estimate the total sample size N:

$$\tilde{N} = (R - \frac{(1-\alpha)M|\mathcal{C}|}{2})/\alpha, \quad R = \sum_{c\in\mathcal{C}} r_c, \tag{2}$$

where $\tilde{N}$ is the estimated total sample size and is proved to be unbiased in the following section. Then the server distributes the ratio of K to $\tilde{N}$ to clients for uniform data sampling.

3.4 Federated Training Framework

Next, we will introduce the workflow of federated training in *FedSampling*, which is summarized in Fig. 2 and Algorithm 1. The workflow of *FedSampling* is similar to the mainstream FL methods in local model training and global model updating, while different in data sampling. The t-th training round in *FedSampling* includes the following steps.

First, the server broadcasts the current model Θ_{t-1} for model aligning and queries each client for the local sample size, where Θ_{t-1} is the model in the $t-1$-th round. Then clients respond to the server based on Eq. 1 and the server

Algorithm 1 Workflow of *FedSampling*

```
for all $t \leftarrow 1$ to $T$ do
    Broadcast model $\Theta_{t-1}$ and query sample sizes
    for $c$ in $\mathcal{C}$ do
        Draw $x_c$ from $\mathcal{B}(\alpha)$, $\hat{n}_c$ from $\mathcal{P}(M)$
        Obtain $r_c$ via Eq. 1
        Send $r_c$ to the server
    end for
    Calculate $\widetilde{N}$ via Eq. 2, and broadcast $\widetilde{N}$, $K$
    for $c$ in $\mathcal{C}$ do
        Initialize an empty data queue $\mathcal{S}_c$
        for $i \leftarrow 1$ to $|\mathcal{D}_c|$ do
            Draw $z_i^c \sim \mathcal{B}(\frac{K}{\widetilde{N}})$
            Push $s_i^c$ into $\mathcal{S}_c$ if $z_i^c > 0$
        end for
        Build local model update based on samples in $\mathcal{S}_c$
        Send local model update to the server
    end for
    $\Theta_t = \Theta_{t-1} + \eta \sum_{\mathbf{G} \in \mathcal{G}} \mathbf{G}$
end for
```

estimates the total sample size $\widetilde{N}$ via Eq. 2. Second, the server broadcasts K and $\widetilde{N}$ to clients. Then clients can further uniformly sample data to obtain the local updates and upload them to the server. Third, until receiving all local updates, the server aggregates them to update the global model: $\Theta_t = \Theta_{t-1} + \eta \sum_{\mathbf{G} \in \mathcal{G}} \mathbf{G}$, where $\mathcal{G}$ is the set of received local updates, and η is the learning rate.

3.5 Discussions on Utility and Privacy

Next, we will analyze the utility and privacy of *FedSampling* in theory and compare it with mainstream FL methods.

Utility Analysis on FedSampling

Definition 3.1. *An FL algorithm is **data-level unbiased**, iff any data can be independently sampled for a training step with identical probabilities to centralized training.*

Based on this definition, data-level unbiased FL methods have a similar data exploitation pattern with centralized learning. Thus, data-level unbiased FL methods have the potential to more effectively exploit decentralized data under imbalanced data size distribution and even the non-IID data distribution. We remark that FL methods based on the uniform or weighted client sampling are not data-level unbiased, due to the violation of independent or identical sampling conditions. Furthermore, based on Lemma 3.1, in *FedSampling* data can be sampled with probabilities identical to centralized learning in a training round. Besides, since *FedSampling* controls the sampling of data via independent random variables, and thereby it is data-level unbiased.

Lemma 3.1. *Let $p(x)$ and $\hat{p}(x)$ denote the probability of a sample x that can participate in a training step in the centralized learning and FedSampling. The mean square error between $p(\cdot)$ and $\hat{p}(\cdot)$ asymptotically converges to* 0.

$$\lim_{|\mathcal{C}| \to \infty} \mathbb{E}[(\hat{p}(x) - p(x))^2] = 0. \tag{3}$$

Proof. Denote the data size distribution as $\mathcal{P}_{\mathcal{D}}$, then we can view the selected response r_c as a combination of three random variables (Eq. 4), and further obtain its expectation:

$$r_c = x_c n_c + (1 - x_c)\hat{n}_c, \quad n_c \sim \mathcal{P}_{\mathcal{D}}, \tag{4}$$

$$\mathbb{E}[r_c] = \alpha \overline{n} + (1 - \alpha)\frac{M}{2}, \tag{5}$$

where $\overline{n}$ is the averaged sample size of different clients. Furthermore, we prove $\widetilde{N}$ is an unbiased estimation of N:

$$\mathbb{E}[R] = \mathbb{E}[\sum_{c \in \mathcal{C}} r_c] = \alpha N + (1 - \alpha)\frac{M|\mathcal{C}|}{2}, \tag{6}$$

$$\mathbb{E}[\widetilde{N}] = (\mathbb{E}[R] - \frac{(1-\alpha)M|\mathcal{C}|}{2})/\alpha = N. \tag{7}$$

Furthermore, without the loss of generalization, we assume that there are K samples selected for model training in a single step. Based on the uniform data sampling strategy in the centralized learning, and the sampling strategy in *FedSampling*, we can obtain the following equations:

$$p(x) = \frac{K}{N}, \quad \hat{p}(x) = \frac{K}{\widetilde{N}}. \tag{8}$$

Let $f(x) = (\frac{1}{x} - \frac{1}{N})^2$, based on the Taylor series, we have:

$$f(x) = f(\mathbb{E}[x]) + \sum_{l=1} \frac{f^l(\mathbb{E}[x])}{l!}(x - \mathbb{E}[x])^l. \tag{9}$$

Let $x = \widetilde{N}$, and note that $f(N) = 0$ and $f^l(N) = \frac{(-1)^l(l-1)l!}{N^{l+2}}$, then we have the following equation:

$$\mathbb{E}[f(\widetilde{N})] = \sum_{l=2} \frac{(-1)^l(l-1)}{N^{l+2}} \mathbb{E}[(\widetilde{N} - N)^l]. \tag{10}$$

Furthermore, the following inequation holds:

$$\mathbb{E}[(\frac{1}{\widetilde{N}} - \frac{1}{N})^2] < \sum_{l=2} \frac{(l-1)}{N^{l+2}} \mathbb{E}[|\widetilde{N} - \mathbb{E}[\widetilde{N}]|^l]. \tag{11}$$

Further, based on tight version of the C_r inequation [Bahr and Esseen, 1965], we have:

$$\mathbb{E}[|\widetilde{N} - \mathbb{E}[\widetilde{N}]|^l] < \frac{2|\mathcal{C}| - 1}{\alpha^l} \mathbb{E}[|r_c - \mathbb{E}[r_c]|^l]. \tag{12}$$

Note that $\mathbb{E}[|r_c - \mathbb{E}[r_c]|^l]$ is a constant determined by the data size distribution and the setting of our size anonymization method (i.e., the value of α and M), and we denote it as δ_l. Note that $|\mathcal{C}| < N$, then we obtain this inequation:

$$\mathbb{E}[(\hat{p}(x) - p(x))^2] < K^2 \sum_{l=2} \frac{2(l-1)}{|\mathcal{C}|^{l+1}\alpha^l} \delta_l. \tag{13}$$

Note that δ_l, and α are constants and irrelevant with $|\mathcal{C}|$, then we can obtain our expected conclusion:

$$\lim_{|\mathcal{C}| \to \infty} \mathbb{E}[(\hat{p}(x) - p(x))^2] < K^2 \lim_{|\mathcal{C}| \to \infty} \sum_{l=2} \frac{2(l-1)}{|\mathcal{C}|^{l+1}\alpha^l} \delta_l = 0. \tag{14}$$

□

Corollary 3.1. *The mean square error between $\frac{NK}{\widetilde{N}}$ and K can asymptotically converge to* 0*:*

$$\lim_{|\mathcal{C}| \to \infty} \mathbb{E}[(\frac{KN}{\widetilde{N}} - K)^2] = 0. \tag{15}$$

Privacy Analysis on FedSampling

For privacy protection, most FL methods disclose sample sizes and may violate privacy constrains. We are the first to propose a privacy-preserving ratio estimation method that can protect privacy in the local sample sizes. Our method can achieve ϵ-LDP in a training round via Lemma 3.2.

Definition 3.2. *A randomized mechanism $\mathcal{M}(\cdot)$ satisfies ϵ-LDP, iff for two arbitrary input x and x', and any output $y \in range(\mathcal{M})$, the following inequation holds:*

$$Pr[\mathcal{M}(x) = y] \leq e^{\epsilon} \cdot Pr[\mathcal{M}(x') = y]. \tag{16}$$

The privacy budget ϵ quantifies the privacy protection ability, where a smaller ϵ means better privacy protection.

Lemma 3.2. *Given an arbitrary size threshold M, FedSampling can meet ϵ-LDP in protecting the privacy of local sample size in each client, when $\alpha = \frac{e^{\epsilon}-1}{e^{\epsilon}+M-2}$.*

Proof. Let $\mathcal{M}(\cdot)$ denote the privacy preserving ratio estimation method. Consider an arbitrary client c with a true response n_c, and a protected response r_c, then we obtain:

$$Pr[\mathcal{M}(n_c) = r_c | r_c = n_c] = \alpha + \frac{1-\alpha}{M-1}, \tag{17}$$

$$Pr[\mathcal{M}(n_c) = r_c | r_c \neq n_c] = \frac{1-\alpha}{M-1}, \tag{18}$$

where we can find the Eq. 17 always larger or equal than Eq. 18. Furthermore, for two arbitrary c and c' in $\mathcal{C}$ and any output $y \in \{y | y = 1, 2, ..., M-1\}$, we have:

$$\begin{aligned} &\max_{c,c',y} \frac{Pr[\mathcal{M}(n_c) = y]}{Pr[\mathcal{M}(n_{c'}) = y]} \\ &= \frac{Pr[\mathcal{M}(n_c) = y | y = n_c]}{Pr[\mathcal{M}(n_{c'}) = y | y \neq n_{c'}]} \\ &= \frac{(M-2)\alpha + 1}{1-\alpha}. \end{aligned} \tag{19}$$

Furthermore, according to the definition of ϵ-LDP, we have:

$$e^{\epsilon} = \max_{c,c',y} \frac{Pr[\mathcal{M}(r_c) = y]}{Pr[\mathcal{M}(r_{c'}) = y]} = \frac{(M-2)\alpha + 1}{1-\alpha}. \tag{20}$$

Thus, when M and ϵ is fixed, we can obtain:

$$\alpha = \frac{e^{\epsilon} - 1}{e^{\epsilon} + M - 2}. \tag{21}$$

□

We can further de-link responses from the same client at different iterations via client anonymity [Sun *et al.*, 2021] to avoid the accumulation of privacy budget. Thus, *FedSampling* can meet ϵ-LDP in the whole training process. Besides, the disclose of local updates may also leak user privacy. We remark that this concern is out of the scope of this paper, as there are already many works [Lyu *et al.*, 2020] that protect local updates via LDP. It is straightforward to apply many of them to *FedSampling* for further privacy protection.

4 Experiment

4.1 Datasets and Experimental Settings

Experiments are conducted on four benchmark datasets: a text dataset *MIND* [Wu *et al.*, 2020], two Amazon review datasets (i.e., *Toys* and *Beauty*) [McAuley *et al.*, 2015], and an image dataset *EMNIST* [Cohen *et al.*, 2017]. First, *MIND*, *Toys* and *Beauty* are used to evaluate model effectiveness under imbalanced data size distribution. These three datasets are used for the text classification task, where *Text-CNN* [Kim, 2014] and the mainstream NLP model *Transformer* [Vaswani *et al.*, 2017] are used as the basic model. Data in *Toys* and *Beauty* can be naturally partitioned into clients based on user IDs, which follows a long-tail distribution (shown in Appendix). Besides, since texts in *MIND* dot not include user information, we partition *MIND* based on a typical long-tail distribution, i.e., log-normal distribution. The average size of samples in each client is set to 2 and we adjust the variance σ of the distribution to control the data size imbalance. σ is set to 4 in experiments. Second, we use *MIND* and *EMNIST* to verify how non-IID data distribution affects different methods. Motivated by McMahan *et al.* [2017], we sort training data in each dataset by their labels, and partition them into different clients of size 300 and 6000 respectively. Thus, distributions of local data in different clients are varied. We use the *ResNet* network [He *et al.*, 2016] for the image classification on *EMNIST*. Marco-F1 and Accuracy are used for classification task evaluation. In our *FedSampling* method, the size threshold M and privacy budget ϵ are set to 300 and 3 respectively. The learning rate η is set to 0.05, and the number K of samples for participating in a training round is set to 2048. All hyper-parameters are selected on the validation set. More model details are in the Appendix and Codes (https://github.com/taoqi98/FedSampling).

4.2 Performance Evaluation

We compare *FedSampling* with several recent federated learning methods. We first compare the standard *FedAvg* algorithm and three adaptive federated optimization methods proposed to improve model effectiveness under heterogeneous data distribution, including: (1) *FedAvg* [McMahan *et al.*, 2017]: uniformly selecting clients for federated model training, and averaging local model updates to train the global model. (2) *FedYogi* [Reddi *et al.*, 2020]: a federated version of the *Yogi* optimization algorithm that smooths global model updates based on momentum strategy [Reddi *et al.*, 2018]. (3) *FedAdagrad* [Reddi *et al.*, 2020], a federated version of the *Adagrad* optimization algorithm [Lydia and Francis, 2019]. (4) *FedAdam* [Reddi *et al.*, 2020], a federated version of the *Adam* optimization algorithm [Kingma and Ba, 2015]. We remark all that these baseline FL methods are based on uniform client sampling according to their original settings. We also apply the centralized model training (*Centralization*) as a baseline method to quantify the performance decline of different federated learning methods.

We repeat each experiment five times and show average performance and standard deviations in Table 1. First, compared with *Centralization*, baseline FL methods have serious performance declines on all datasets. This is because

Model	Training Algorithm	MIND		Toys		Beauty	
		Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy
Text-CNN	Centralization	51.52±0.57	71.14±0.45	39.61±1.13	63.71±0.22	43.90±0.97	62.20±0.67
	FedAvg	48.11±0.66	69.23±0.73	35.32±0.78	61.63±0.33	38.44±1.43	60.75±0.36
	FedYogi	49.12±0.71	68.92±0.40	35.62±2.34	61.22±0.39	38.77±0.89	60.35±0.91
	FedAdagrad	48.55±0.92	67.74±1.89	34.69±0.70	60.63±1.36	37.20±1.90	60.64±0.70
	FedAdam	48.54±0.65	68.22±0.50	35.27±1.59	61.35±0.32	39.09±0.80	60.43±1.05
	FedSampling	**51.33**±0.62	**71.15**±0.30	**40.15**±1.27	**63.41**±0.74	**43.04**±0.83	**62.96**±0.16
Transformer	Centralization	53.73±0.62	72.19±0.28	41.86±0.96	63.56±0.57	44.31±0.70	62.92±0.48
	FedAvg	50.46±0.99	70.74±0.52	38.68±0.93	60.30±2.06	37.82±1.36	60.41±0.27
	FedYogi	50.94±0.59	70.29±0.53	37.75±1.87	61.44±0.36	38.10±1.07	60.17±0.33
	FedAdagrad	50.99±0.68	70.65±0.48	38.06±0.61	59.69±1.60	38.59±1.56	59.87±0.51
	FedAdam	50.69±0.58	70.83±0.28	37.58±0.77	60.59±1.24	38.44±1.42	60.65±0.46
	FedSampling	**53.43**±0.57	**71.98**±0.37	**41.63**±1.12	**64.03**±0.46	**43.47**±0.94	**62.67**±0.60

Table 1: Results on the text classification task. Best results in federated settings are in bold.

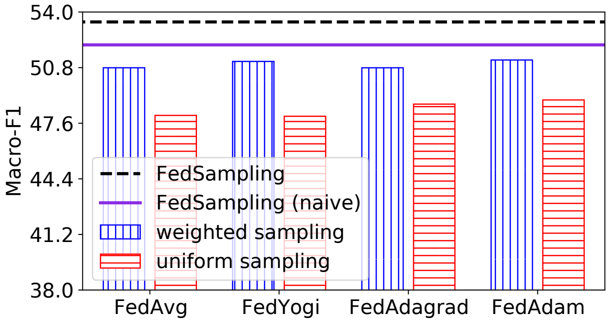

Figure 3: Comparisons of *FedSampling* with weighted client sampling on different federated learning methods.

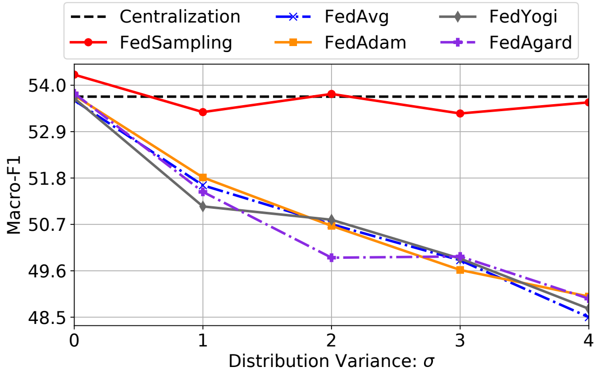

Figure 4: Influence of data size imbalance degree.

the size distributions of many real-world user data are usually imbalanced. However, baseline FL methods usually uniformly select some clients to participate in a training round, and thereby fail to effectively exploit massive training data in some long-tail clients. Second, our *FedSampling* method can consistently achieve comparable performance with *Centralization*. This is because our *FedSampling* method has a sample-level sampling strategy, which can uniformly and independently sample data to achieve a uniform and unbiased data exploitation like centralized learning.

To more effectively exploit decentralized data for model training, some existing federated learning methods [Li *et al.*, 2019; Wang *et al.*, 2020] sample clients based on local sample sizes. Thus, we apply the weighted client sampling strategy to baseline FL methods for further comparisons. Due to space limitations, we only show results on *MIND* with the *Transformer* model in the following sections. Results are presented in Fig. 3, and we have several findings. First, we can find that weighted client sampling significantly improves the performance of baseline FL methods. This is because based on weighted client sampling long-tail clients have more opportunities to participate in a training round, and thereby massive data in them can be more effectively exploited. Second, compared with *Centralization* and *FedSampling*, performance of baseline FL methods with weighted sampling are still inferior. This is because weighted client sampling usually emphasizes long-tail clients during model training while head clients can also accumulate non-neglect samples. These methods still cannot achieve uniform data exploitation, which leads to sub-optimal performance. Different from these methods, we propose a sample-level sampling strategy which can independently and uniformly sample decentralized data for training. In addition, the weighted client sampling strategy requires the server to track local samples sizes of each client, which may arouse privacy concerns due to the privacy sensitivity of local sample sizes. Different from them, *FedSampling* protects the local sample size of each client via a privacy-preserving ratio estimation method with a theoretical privacy guarantee. Besides, we also compare the naive method introduced in approach section with *FedSampling*. Results show that performance of the naive method is inferior to *FedSampling*. This is because to avoid the privacy leakage on sample sizes, the naive method needs to uniformly average local updates to learn the global model, which is biased in data exploitation.

4.3 Influence of Data Size Distribution Imbalance

Next, we will analyze how the imbalance of data size distribution affects different methods. According to our experimental settings, the data partition of *MIND* is based on a log-normal distribution. Thus, we can control the size imbalance by adjusting the variance σ of the log-normal distribution, where

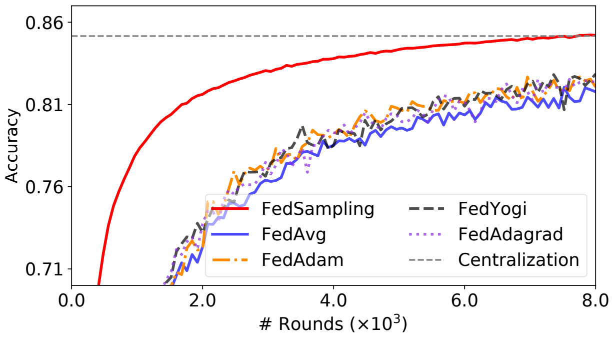

Figure 5: Performance comparisons under non-IID data distribution on the *EMNIST* dataset.

larger σ leads to more imbalanced sample sizes. Results in Fig. 4 show the influence of σ on the effectiveness of different methods, from which can we summarize two major phenomenons. First, with the increase of σ, the performance of baseline methods consistently declines. This is because the increase of σ makes sample sizes more imbalanced, where long-tail clients can keep more data. Thus, there will be more data that fails to be effectively exploited by baseline FL methods. Second, *FedSampling* has comparable performance with *Centralization* under different σ. This is because *FedSampling* achieves uniform sampling in data level and can effectively learn models under imbalanced data size distribution.

4.4 Comparisons under Non-IID Distribution

Since *FedSampling* can uniformly sample decentralized data, it has the potential to improve the effectiveness of FL under non-IID data distribution. Next, we compare different methods under the non-IID data distribution. We set clients to have equal sample size, i.e., 300, which makes no differences for baseline FL methods to perform uniform or weighted client sampling. Thus, in this section, we only show results of baseline FL methods with uniform client sampling. Results are shown in Fig. 5, from which we can find the convergence of *FedSampling* is faster and smoother. This is because distributions of data on different clients are highly heterogeneous, where most clients only keep data with the same classification category. However, baseline FL methods sample data for training at the client level, which results in heterogeneous local model updates and hurts the model convergence. Different from these methods, *FedSampling* can independently select samples in different clients for model training with identical opportunities. Thus, *FedSampling* can achieve a similar data sampling pattern with the canonical uniform data sampling in centralized learning, and improve the effectiveness of FL models under the non-IID data distribution.

4.5 Utility and Privacy Analysis

Next, we will analyze the trade-off between utility and privacy of our *FedSampling* method. As shown in Fig. 6, given a size threshold M, we evaluate the performance of *FedSampling* under different privacy budget ϵ. First, with a constant size threshold M, a smaller privacy budget ϵ usually leads to a worse model performance. This is because a smaller privacy budget ϵ increases the probability of clients for sending a fake response to the server, which hurts the accuracy of the total sample number estimation and the quality of data sampling. Second, when the privacy budget ϵ is fixed, *FedSampling* can achieve the best performance under a moderate size threshold M, e.g., 10^2. This is because when M is too small, many true responses may be truncated by a small value and cause bias on the estimation of total sample size. Besides, according to Lemma 3.2, when the privacy budget ϵ is fixed, a larger size threshold M requires larger probabilities of sending fake responses, which hurts the accuracy of data size distribution estimation. Thus, we select a moderate value of size threshold M (i.e., 100) and privacy budget ϵ (i.e., 3) in experiments.

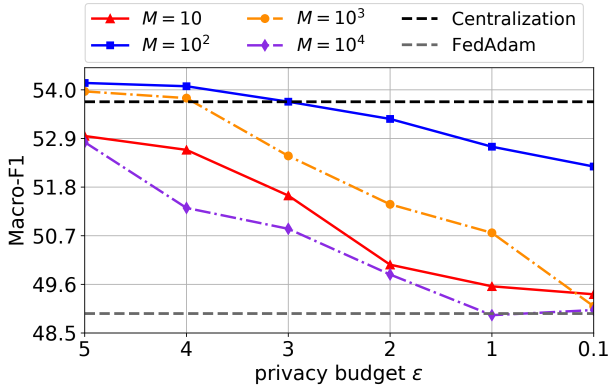

Figure 6: Utility and privacy analysis of *FedSampling*.

5 Conclusion

In this paper, we propose a uniform data sampling strategy for federated learning (named *FedSampling*), which can achieve an uniform data exploitation. In each round of *FedSampling*, each sample in each client is independently sampled for model training according to an identically probability, which is based on the ratio of the server desired sample size to the total sample size in all available clients. We also propose a privacy-preserving method to estimate an unbiased total sample size with an LDP privacy guarantee on the local sample size of each client. We conduct extensive experiments on four benchmark datasets. Experimental results show that *FedSampling* can outperform many FL methods especially under imbalanced data size distribution and non-IID data distribution. Although effective, *FedSampling* may increase the communication cost of federated learning. In our future work we will explore to handle this problem.

Acknowledgments

This work was supported by the National Key Research and Development Program of China 2022YFC3302104, Tsinghua University Initiative Scientific Research Program of Precision Medicine under Grant number 2022ZLA0073, Tsinghua-Toyota Joint Research Fund 20213930033, and China Postdoctoral Science Fundation (No.2022M721892).

References

[Achituve *et al.*, 2021] Idan Achituve, Aviv Shamsian, Aviv Navon, Gal Chechik, and Ethan Fetaya. Personalized federated learning with gaussian processes. *NeurIPS*, 2021.

[Avdiukhin and Kasiviswanathan, 2021] Dmitrii Avdiukhin and Shiva Kasiviswanathan. Federated learning under arbitrary communication patterns. In *ICML*, pages 425–435, 2021.

[Bahr and Esseen, 1965] Bengt von Bahr and Carl-Gustav Esseen. Inequalities for the rth absolute moment of a sum of random variables, $1 \leq \mathrm{r} \leq 2$. *The Annals of Mathematical Statistics*, pages 299–303, 1965.

[Bai *et al.*, 2021] Xiang Bai, Hanchen Wang, Liya Ma, Yongchao Xu, Jiefeng Gan, Ziwei Fan, Fan Yang, Ke Ma, Jiehua Yang, Song Bai, et al. Advancing covid-19 diagnosis with privacy-preserving collaboration in artificial intelligence. *Nature Machine Intelligence*, pages 1–9, 2021.

[Cohen *et al.*, 2017] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *IJCNN*, pages 2921–2926, 2017.

[Dayan *et al.*, 2021] Ittai Dayan, Holger R Roth, Aoxiao Zhong, Ahmed Harouni, Amilcare Gentili, Anas Z Abidin, Andrew Liu, Anthony Beardsworth Costa, Bradford J Wood, Chien-Sung Tsai, et al. Federated learning for predicting clinical outcomes in patients with covid-19. *Nature Medicine*, pages 1735–1743, 2021.

[Fraboni *et al.*, 2021a] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. Clustered sampling: Low-variance and improved representativity for clients selection in federated learning. *arXiv preprint arXiv:2105.05883*, 2021.

[Fraboni *et al.*, 2021b] Yann Fraboni, Richard Vidal, Laetitia Kameni, and Marco Lorenzi. On the impact of client sampling on federated learning convergence. *arXiv preprint arXiv:2107.12211*, 2021.

[Gerber *et al.*, 2018] Nina Gerber, Paul Gerber, and Melanie Volkamer. Explaining the privacy paradox: A systematic review of literature investigating privacy attitude and behavior. *Computers & security*, pages 226–261, 2018.

[Hamer *et al.*, 2020] Jenny Hamer, Mehryar Mohri, and Ananda Theertha Suresh. Fedboost: A communication-efficient algorithm for federated learning. In *ICML*, pages 3973–3983, 2020.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Huang *et al.*, 2021] Baihe Huang, Xiaoxiao Li, Zhao Song, and Xin Yang. Fl-ntk: A neural tangent kernel-based framework for federated learning analysis. In *ICML*, pages 4423–4434, 2021.

[Jiang *et al.*, 2021] Ziyue Jiang, Yi Ren, Ming Lei, and Zhou Zhao. Fedspeech: Federated text-to-speech with continual learning. In *IJCAI*, pages 3829–3835, 2021.

[Jumper *et al.*, 2021] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, pages 583–589, 2021.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, 2014.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Li *et al.*, 2019] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. In *ICLR*, 2019.

[Li *et al.*, 2020] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, pages 50–60, 2020.

[Liang *et al.*, 2021] Feng Liang, Weike Pan, and Zhong Ming. Fedrec++: Lossless federated recommendation with explicit feedback. In *AAAI*, pages 4224–4231, 2021.

[Lydia and Francis, 2019] Agnes Lydia and Sagayaraj Francis. Adagrad—an optimizer for stochastic gradient descent. *IJICS*, 2019.

[Lyu *et al.*, 2020] Lingjuan Lyu, Han Yu, Xingjun Ma, Lichao Sun, Jun Zhao, Qiang Yang, and Philip S Yu. Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337*, 2020.

[McAuley *et al.*, 2015] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52, 2015.

[McMahan *et al.*, 2017] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, pages 1273–1282, 2017.

[Reddi *et al.*, 2018] S Reddi, Manzil Zaheer, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. In *NIPS*, 2018.

[Reddi *et al.*, 2020] Sashank J Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In *ICLR*, 2020.

[Rothchild *et al.*, 2020] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. Fetchsgd: Communication-efficient federated learning with sketching. In *ICML*, pages 8253–8265, 2020.

[Sun *et al.*, 2021] Lichao Sun, Jianwei Qian, and Xun Chen. Ldp-fl: Practical private aggregation in federated learning with local differential privacy. In Zhi-Hua Zhou, editor, *IJCAI*, pages 1571–1578, 2021.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2020] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *NeurIPS*, 2020.

[Wang *et al.*, 2021] Zheng Wang, Xiaoliang Fan, Jianzhong Qi, Chenglu Wen, Cheng Wang, and Rongshan Yu. Federated learning with fair averaging. In *IJCIA*, pages 1615–1623, 2021.

[Wu *et al.*, 2020] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *ACL*, pages 3597–3606, 2020.

[Yang *et al.*, 2018] 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.

[Yoon *et al.*, 2021] Jaehong Yoon, Wonyong Jeong, Giwoong Lee, Eunho Yang, and Sung Ju Hwang. Federated continual learning with weighted inter-client transfer. In *ICML*, pages 12073–12086, 2021.

[Yuan *et al.*, 2021] Honglin Yuan, Manzil Zaheer, and Sashank Reddi. Federated composite optimization. In *ICML*, pages 12253–12266, 2021.