Federated Learning with Server Learning for Non-IID Data

Van Sy Mai
CTL, NIST
Gaithersburg, MD USA
vansy.mai@nist.gov

Richard J. La
ITL, NIST & Univ. of Maryland
Gaithersburg, MD USA
richard.la@nist.gov

Tao Zhang
CTL, NIST
Gaithersburg, MD USA
tao.zhang@nist.gov

Yuxuan Huang
CTL, NIST
Gaithersburg, MD USA
yuxuan.huang@nist.gov

Abdella Battou
CTL, NIST
Gaithersburg, MD USA
abdella.battou@nist.gov

***Abstract*—Federated Learning (FL) has gained popularity as a means of distributed learning using local data samples at clients. However, recent studies showed that FL may experience slow learning and poor performance when client samples have different distributions. In this paper, we consider a server with access to a small dataset, on which it can perform its own learning. This approach is complementary to and can be combined with other approaches, e.g., sample sharing among clients. We study and demonstrate the benefits of proposed approach via experimental results obtained using two datasets – EMNIST and CIFAR10.**

***Index Terms*—Federated learning, Distributed optimization**

I. Introduction

Federated Learning (FL) is a recent paradigm in which multiple clients collaborate under the coordination of a central server to solve a machine learning (ML) problem [6]. A key advantage is that local data at clients need not be shared with the central sever or other clients. Learning locally is becoming necessary as, in many networking applications, a large amount of data is created or collected at the network edge and cannot all be sent to the cloud due to many factors, such as network capacity constraints, latency requirements, and privacy concerns [2].

In its basic form, FL trains a global model for all clients by following an iterative procedure. At each global round: 1) the central server selects a subset of clients and shares the current global model, 2) each selected client updates the model only using its local data and forwards the updated model to the central server, and 3) the server aggregates the updated local models from the clients to update the global model. This process is repeated until certain convergence criteria are satisfied.

Conventional FL techniques, e.g., Federated Averaging (FedAvg) [11], carry out model aggregation by averaging the model parameters received from clients. This performs well when clients have independent and identically distributed (IID) training samples. In practice, however, local data at the clients often do not satisfy this IID assumption for various reasons. For example, different clients may collect data from different sources, using different tools, under different conditions, or may have access only to partial or biased data, which can cause the data distributions at the clients to differ. The differences, often referred to as *drifts* or *shifts*, can also take different forms [6].

It has been shown that conventional FL techniques can suffer from slow convergence when the training data distributions among the clients diverge considerably [4], [5], [7], [9], [13]. Such performance degradation can be brought on by, for example, feature divergence: different training data distributions cause local models to focus on different features and feature representations. Feature divergence makes it difficult to align the optimization objectives among local models. Non-IID training data distributions can also cause each client to optimize its local model toward the local optima for its local data, which can differ significantly from the global optima. This can cause the weights of clients' models to diverge [13]. As a result, simply averaging local models may not push the aggregated model toward a global optimum. Recently, growing efforts have been devoted to improving the performance of FL for non-IID datasets at clients, e.g., [7], [12], [13].

We propose a new FL algorithm to improve the performance on non-IID client data. Specifically, the central server will collect a *small* amount of training data, learn from it, and then distill the knowledge into the global model through FL process in an *incremental* fashion. Unlike the data sharing method in [13], our approach does not need to send any global or shared dataset to clients. Therefore, it does not increase either the computing/storage requirements at the clients or communication overheads compared to the traditional FL. We will show that having the server learn from a small but good dataset can reduce data drifts and improve overall performance. Additionally, thanks to the incremental nature of our learning approach, a significant improvement in convergence speed can be achieved especially at the beginning of the learning process.

The rest of the paper is organized as follows: Section II first defines the ML problem as an optimization problem

and introduces our proposed algorithm (Algorithm 1) and its analysis. Section III presents experimental results obtained using two publicly available datasets, followed by conclusions in Section IV.

Notation: For each integer $n > 0$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. For a finite set $\mathcal{D}$, $|\mathcal{D}|$ denotes its cardinality. For any vector x, $\|x\|$ denotes its 2-norm. We denote by $\langle x, y\rangle$ the inner product of two vectors x and y. A function $f : D \to \mathbb{R}$ is said to be smooth with parameter L, or simply L-smooth, if $f(x) - f(y) - \langle \nabla f(y), x - y\rangle \leq \frac{L}{2}\|x - y\|^2$ for all $x, y \in D$. For a random variable X, we use both $\mathbb{E}[X]$ and $\mathbb{E}X$ to denote its expected value.

II. Problem Formulation and Proposed Algorithm

A. Problem Formulation

Consider an ML problem in which we aim to solve the following optimization problem:

$$\min_{x \in \mathbb{R}^d} \quad F(x) \triangleq \frac{1}{n} \sum_{s \in \mathcal{D}} \ell(x, s), \tag{1}$$

where $x \in \mathbb{R}^d$ is the vector of model parameters that need to be learned, $\mathcal{D} = \{s_1, \ldots, s_n\}$ is the set of n training samples, and $\ell(x, s)$ is the loss associated with sample s when the model parameters are x. In FL (and other ML problems), the training goal is to find optimal model parameters x^* that solve (1). However, unlike in more traditional ML problems, in FL, training data or samples are located at the clients that only communicate with a central server (coordinator).

Suppose that there are N clients, and training dataset $\mathcal{D}$ is partitioned into $\{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$, where $\mathcal{D}_i$ is client i's local dataset. Define $p_i := n_i/n$ ($n_i := |\mathcal{D}_i|$) to be the fraction of training samples that reside at client i. Define client i's loss function to be $f_i : \mathbb{R}^d \to \mathbb{R}$, where

$$f_i(x) = \frac{1}{n_i} \sum_{s \in \mathcal{D}_i} \ell(x, s).$$

Then, (1) can be rewritten as

$$\min_{x \in \mathbb{R}^d} \quad \sum_{i \in [N]} p_i \cdot f_i(x).$$

The weights $\{p_i\}_{i \in [N]}$, are used to account for differing clients' data sizes. Here, to simplify our exposition, we assume that each client holds the same number of training samples, i.e., $n_i = n_j$ (or $p_i = p_j = 1/N$) for all $i, j \in [N]$.

We assume that the server will either (i) collect a small set of training samples from the clients or (ii) have access to a small set of extra samples that are not shared with the clients. The server uses these samples to perform its own learning in addition to the learning carried out by clients. Denote this set of training samples at the server by $\mathcal{D}_0$ with $n_0 = |\mathcal{D}_0|$, and the server's loss by

$$f_0(x) = \frac{1}{n_0} \sum_{s \in \mathcal{D}_0} \ell(x, s).$$

Different from existing methods (e.g., [13]), our hybrid scheme combines FL and server learning without distributing common shared training samples to clients. We will show that, compared to traditional FL methods, the proposed scheme improves the learning process, even with a small $\mathcal{D}_0$, when the sample distributions among the clients are heterogeneous.

B. Main Algorithm

We augment FL with local learning at the server, which we refer to as *server learning* (SL). There are several options to incorporate SL into FL. One option is to treat the server as a regular client that participates in every round of FL process [12]. In other words, during each (global) round, the server first updates the current model with its local data and then aggregates it with the updated models reported by the clients. We refer to this approach as *non-incremental* learning. During model aggregation, the server may assign a special weight to its own updated model. However, this weight is often very small when $n_0 \ll n$. Moreover, this approach does not exploit the good quality of $\mathcal{D}_0$ when its distribution is close to that of $\mathcal{D}$ or it includes more diverse samples.

These observations motivate us to consider an *incremental learning* scheme in which the server performs additional learning on the aggregated model (Algorithm 1 below). This is similar to the incremental (stochastic) gradient method, which is shown to be much faster (compared to its non-incremental gradient method) when the model is far from a (locally) optimal point [1]. While FL with local stochastic gradient descent (SGD) also works in an incremental fashion, it often needs small learning rates and, hence, longer learning times, to ensure convergence when the distribution of clients' data is heterogeneous. Here, we take advantage of server's training dataset when its distribution is closer to that of aggregate dataset $\mathcal{D}$ than individual clients' datasets. This ensures that the loss function of the server f_0 is similar to the overall loss function F in (1). Consequently, when the current model is far from a (locally) optimal point, the gradient ∇f_0 will approximately track the global gradient ∇F, even when clients' gradients ∇f_i do not follow ∇F closely. We will elaborate on this below.

C. Brief Analysis

For ease of exposition, assume that the server can compute gradient $\nabla f_0(x)$ for any x and consider the usual gradient descent (GD) method for SL. Consider one step of GD carried out by the server starting from any w_0, i.e., $w_1 = w_0 - \eta_0 \nabla f_0(w_0)$. Suppose that ∇F is Lipschitz continuous with parameter L.[1] Then,

$$\begin{aligned} F(w_1) \quad &\leq \quad F(w_0) + \langle \nabla F(w_0), w_1 - w_0\rangle \\ &\qquad +0.5L\|w_1 - w_0\|^2 \\ &= \quad F(w_0) - \eta_0 \langle \nabla F(w_0), \nabla f_0(w_0)\rangle \\ &\qquad +0.5L\eta_0^2 \|\nabla f_0(w_0)\|^2. \end{aligned} \tag{2}$$

[1]This assumption is standard in FL and often holds when training neural networks.

Algorithm 1: FL with Server Learning

Server:
initialize x_0, K, K_0, η_l, η_0
for $t = 0, \ldots, T-1$ **do**
  sample a subset $\mathcal{S}$ of clients
  broadcast x_t to clients
  forall clients $i \in \mathcal{S}$ **do**
    $x_{t,K}^{(i)} = \text{LocalSGD}(x_t, \eta_l, K, \mathcal{D}_i)$
    upload to server: $x_{t,K}^{(i)}$
  $\bar{x}_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} x_{t,K}^{(i)}$
  $x_{t+1} = \text{LocalSGD}(\bar{x}_t, K_0, \eta_0, \mathcal{D}_0)$

$\text{LocalSGD}(x, \eta, K, \mathcal{D})$:
$y_0 = x$
for $k = 0, \ldots, K-1$ **do**
  compute an unbiased estimate $g(y_k)$ of $\nabla f_i(y_k)$
  $y_{k+1} = y_k - \eta g(y_k)$
return: y_K

As we can see, SL can improve FL further when the second term in (2) is sufficiently negative so that

$$2\langle \nabla F(w_0), \nabla f_0(w_0) \rangle > L\eta_0 \|\nabla f_0(w_0)\|^2. \tag{3}$$

This condition holds when $\nabla f_0(w_0)$ makes an acute angle with $\nabla F(w_0)$ (provided that $\|\nabla F(w_0)\| > 0$), in which case progress can be made by using a sufficiently small step size η_0. This is the case when the distribution of $\mathcal{D}_0$ is not very different from that of $\mathcal{D}$ and, when w_0 is far from a (local) minimizer, $-\nabla f_0(w_0)$ will likely be a descent direction of F.

In order to further see the role of $\mathcal{D}_0$, let us rewrite condition (3) as follows:

$$\begin{aligned} &\|\nabla F(w_0)\|^2 + (1 - L\eta_0)\|\nabla f_0(w_0)\|^2 \\ &> \|\nabla F(w_0) - \nabla f_0(w_0)\|^2 \end{aligned} \tag{4}$$

We have the following remarks. First, the quantity $\|\nabla F(w_0) - \nabla f_0(w_0)\|^2$ is in general characterized by the relationship between server's dataset $\mathcal{D}_0$ and the aggregate dataset $\mathcal{D}$; the more dissimilar $\mathcal{D}_0$ is to $\mathcal{D}$, the larger the error and thus the smaller improvement (it can in fact have negative impact if too large). This suggests that the server dataset should be selected carefully in order to maximize the benefits of SL. Consider, e.g., when $\mathcal{D}_0$ consists of IID samples from $\mathcal{D}$. In this case, the error $\|\nabla F(w_0) - \nabla f_0(w_0)\|^2$ tends to decrease with the size of $\mathcal{D}_0$ according to (sampling without replacement)

$$\mathbb{E}_{\mathcal{D}_0} \|\nabla f_0(x) - \nabla F(x)\|^2 = \Big(\frac{n}{n_0} - 1\Big) \frac{\tilde{\sigma}_0^2(x)}{n-1},$$

where $\tilde{\sigma}_0^2(x) = \frac{1}{n} \sum_{s \in \mathcal{D}} \|\nabla_x \ell(x, s) - \nabla F(x)\|^2$ is the population variance. Thus, condition (4) can be satisfied by increasing n_0. Second, for fixed $\mathcal{D}_0$ (with reasonable quality), the inequality in (4) holds when $\|\nabla F(w_0)\|$ is large, i.e., w_0 is far from being a stationary point, which is to be expected at the beginning of the training process. This is true even when $\nabla f_0(x)$ is a biased estimate of $\nabla F(x)$ as long as $\|\nabla F(w_0) - \nabla f_0(w_0)\|^2$ is strictly smaller than $\|\nabla F(w_0)\|^2 + \|\nabla f_0(w_0)\|^2$, i.e., the angle between the gradients is acute as mentioned earlier, for a sufficiently small step size η_0. Third, when $\|\nabla f_0(w_0)\|$ is sufficiently small, e.g., when overfitting happens at the server, the improvement by SL is also insignificant. Finally, when w_0 is near a stationary point of F but far from that of f_0, i.e., $\|\nabla F(w_0)\| \ll \|\nabla f_0(w_0)\|$, the inequality in (4) may be reversed, in which case SL can impair FL, pushing toward server's local stationary points. In this case, our algorithm does not yield exact convergence but oscillates between stationary points of F and f_0, which is expected for an incremental gradient method [1].

The above analysis also applies to multi-step learning at the server. In particular, suppose the server takes K_0 steps of the GD method using step size η_0,

$$w_{t,k+1} = w_{t,k} - \eta_0 \nabla f_0(w_{t,k}), \quad k = 0, \ldots, K_0 - 1,$$

with $w_{t,0} = \bar{x}_t$ and $x_{t+1} = w_{t,K_0}$. Then, we can show

$$\begin{aligned} F(x_{t+1}) \leq F(\bar{x}_t) &- 0.5\eta_0 \textstyle\sum_{k=0}^{K_0-1} \big(\|\nabla F(w_{t,k})\|^2 \\ &+ (1 - L\eta_0)\|\nabla f_0(w_{t,k})\|^2\big) \\ &+ 0.5\eta_0 \textstyle\sum_{k=0}^{K_0-1} \|\nabla F(w_{t,k}) - \nabla f_0(w_{t,k})\|^2. \end{aligned}$$

Here, again, the benefit of taking multiple steps at the server materializes when $w_{t,k}$ is far from being a stationary point of either F or f_0, more precisely, $\|\nabla F(w_{t,k})\|^2 + (1 - L\eta_0)\|\nabla f_0(w_{t,k})\|^2 > \|\nabla F(w_{t,k}) - \nabla f_0(w_{t,k})\|^2$. This also suggests that the server should not overfit its own data, which could happen easily when the data size is small.

For further convergence analysis as well as technical proofs, interested readers are referred to our technical report in [10].

III. Experimental Results

In this section, we present experimental results to demonstrate the benefits of the proposed approach and the effects of server dataset size n_0 on its performance, using two commonly used public datasets.

Datasets and Models We carry out the evaluation using two datasets – CIFAR-10 [8] and EMNIST-Balanced [3]. We use convolutional neural networks (CNNs) for our experiments. We distribute n samples evenly among N clients (n/N samples per client). In order to study the effects of clients' sample heterogeneity, we vary the number of sample labels that reside at each client, which we denote by C: for fixed C, each client has $n/(N \times C)$ samples with the same label.

Methods We compare our approach against two baseline methods: (1) Federated Learning FedAvg (`FL`) [11] and (2) `FL` combined with Data Sharing (`DS`) [13]. This is done by ensuring that all clients have access to a small set of *common* samples to which the server has access and uses for *pre-training* the model. Because our approach SL is complementary in nature, in order to evaluate its benefits, we compare the performance of `FL` and `DS` with and without SL. We refer to `FL` and `DS` with SL as `FSL` and `DSL`, respectively.

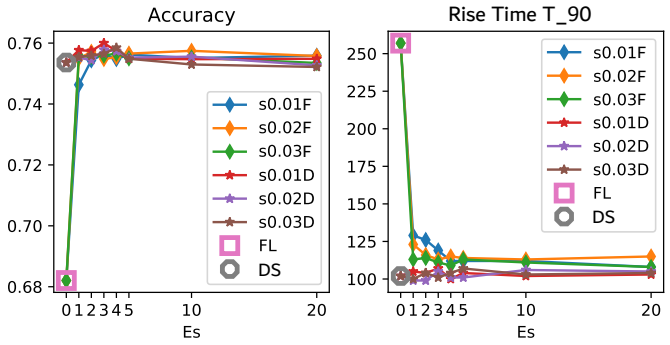

Fig. 1. Plot of accuracy and rise time (F = FSL, D = DSL, $n_0 = 235, C = 2, S = 5, \eta_l = 0.04$).

In order to facilitate our comparison, we set the number of common samples shared by all clients in DS and DSL to the number of samples available to the server (and used in SL). This correspond to setting the data sharing parameter $\alpha = 1$ in DS [13]. We do not present the results for the non-incremental server learning scheme (described in Section II) as it underperforms our incremental version in Algorithm 1.

Implementation We implemented all considered algorithms in TensorFlow (version 2.7). We denote the number of clients that are polled at each (global) round by S, which is fixed to be $S = 5$ for EMNIST dataset and $S = 3$ for CIFAR-10 dataset in our study. For a study on the impact of varying the value of S, we refer an interested reader to [10]. A client chosen by the server at a round trains its local model for E_c epochs over its local data using batch size B. When SL is employed, at each round, the server updates its local model for E_s epochs with the same batch size. In our experiments, we fix $E_c = 1$ and $B = 100$.

Evaluations We run all algorithms for 400 rounds, and compare the test accuracy and convergence rate measured by rise time T_{90} (which is the first time the accuracy averaged over the latest 20 rounds reaches 90% of the average accuracy during the last 20 rounds of the experiment). The reported numbers are the averages of 5 runs.

A. Numerical Results

1) Dataset #1 (EMNIST): We use EMNIST-Balanced dataset with 112,800 samples of 47 label classes for training. There are 47 clients, each of which holds 2,400 samples and client learning rate $\eta_l = 0.04$. Samples from each label is hosted by two clients, i.e., each client has 1,200 samples with the same label.

• **Benefits of SL:** Fig. 1 plots the average accuracy of the last 20 rounds and the rise time T_{90} for different server learning rates ($\eta_0 \in \{0.01, 0.02, 0.03\}$). For DS, FSL, DSL, the server has 235 samples (which are also shared with clients in DS, DSL). At each round, 5 clients are polled to update the model. Hence, only a fraction of labels are considered at each round. First, the plots suggest that SL provides a substantial improvement for FL. However, the performance

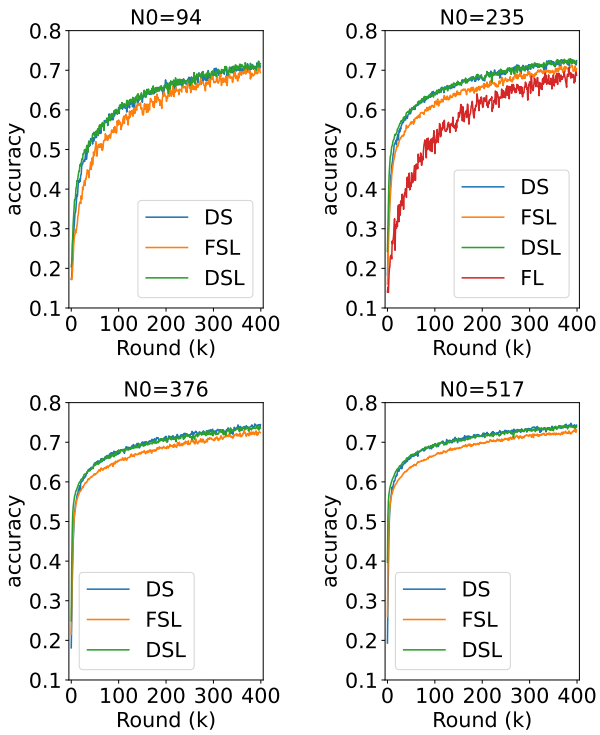

Fig. 2. Plot of accuracy for different server dataset sizes.

of DS and DSL is nearly identical. Although DS (and DSL) slightly outperforms FSL, the performance difference is minor. We believe that these observations owe to the possibility that learning from EMNIST dataset is 'easy' in the sense that any reasonable approach can learn the task efficiently, including DS. As a consequence, three schemes – DS, FSL, and DSL – have comparable performance. Second, the figures indicate that the server iterations E_s can be set small. This is expected; when the server dataset is small, setting E_s large will likely lead to overfitting, thereby degrading learning. Thus, for SL performed on a small server dataset (which is the setting we are interested in), E_s should be set small. Finally, although we do not report them here due to a space constraint, similar observations hold in other settings with varying C, S, and learning rates.

• **Effects of server dataset size:** Next, we examine the performance of DS, FSL, DSL as we vary the number of samples available to the server (n_0). Here, we do not use pretraining (so that all methods start at the same point) and we fixed the server learning rate to be $\eta_0 = 0.03$. Fig. 2 shows the accuracy for $n_0 \in \{94, 235, 376, 517\}$. There are a few noteworthy observations: first, only a small number of samples are needed for DS and SL to enjoy a significant improvement in accuracy. In fact, the plots suggest that the performance gain slows down considerably after $n_0 = 376$ (8 samples per label). Second, perhaps somewhat surprisingly, even having access to 5 samples per label ($n_0 = 235$) delivers a meaningful improvement in accuracy over FL. We believe that this is another indication that EMNIST dataset is easy to

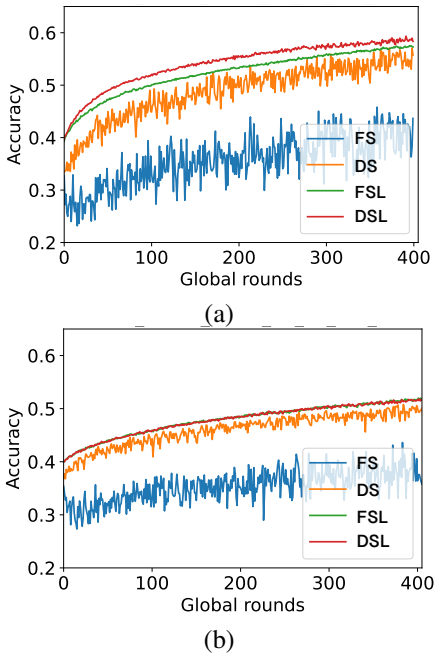

Fig. 3. Plot of accuracy as a function of iterations with 400 samples at the server. (a) $N = 10$, (b) $N = 100$.

learn from and only a small number samples from each label is needed to learn the tasks.

2) Dataset #2 (CIFAR-10): The second set of experiments is conducted using CIFAR-10 dataset with $n = 40{,}000$ samples. Both the client local learning rate and the server local learning rate are set to 0.02 unless stated otherwise.

• **Benefits of SL:** Fig. 3 shows the accuracy as a function of global rounds k for two scenarios – (a) $N = 10$ and (b) $N = 100$. We chose parameters $C = 2$ and $S = 3$, and $n_0 = 400$ (40 samples per label) in both cases. Furthermore, we used the same pre-trained initial CNN model using the server samples so that we can compare accuracy for all four considered methods.

We make following observations from the figure: (i) In both cases, SL improves the accuracy. In particular, FSL significantly outperforms FL, suggesting that indeed SL can help alleviate the problem caused by the skewness in client samples; (ii) in these cases, SL offers greater benefits than data sharing in DS. Compared to DS in which clients leverage *shared common* samples to cope with sample distribution skewness, SL achieves faster learning at the beginning; and (iii) the accuracy is much more stable with SL. This is due to the fact that, under proposed SL, in the second stage of each round, a noisy estimate of correct gradient computed by the server (using samples from *all* labels) pulls the incorrect, biased gradient from updated models of polled clients obtained using samples with a *subset* of labels in the right direction. While DS benefits from the shared samples, since they are mixed with (larger) sets of clients' samples, their contribution to the computed gradient is diminished in DS.

• **Effects of client sample distribution:** Fig. 4 shows the effect of the skewness in client sample distribution (captured

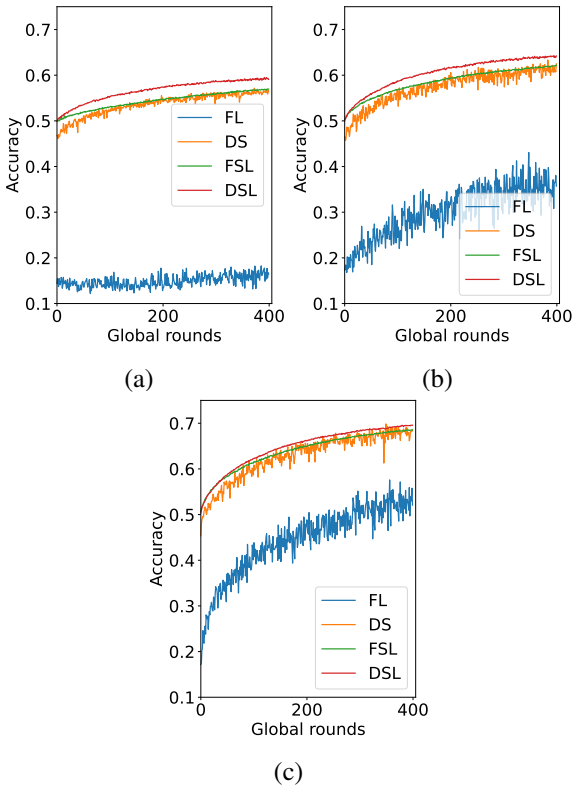

Fig. 4. Plot of accuracy for $n_0 = 1000$, $N = 10$, $S = 3$ and (a) $C = 1$, (b) $C = 2$, (c) $C = 5$.

by C) on the considered algorithms. For this experiment, we did not pre-train the initial model for FL (as it would not be in practice), which is reflected in much worse initial accuracy of FL. First, it is clear that the performance of FL improves with increasing C as samples with more diverse labels are used to update the model at each round. Second, when the performance of FL worsens (with samples with the same label becoming concentrated at fewer clients), the relative benefits of SL become more pronounced. Third, when client sample distribution is highly biased, FSL performs on par or better than DS. This suggests that SL, when used by itself, could serve as a more effective means of handling the bias of client sample distribution, than sharing a set of common samples (as done in DS) when datasets are difficult to learn from. However, when client samples are more evenly distributed, SL may experience a slight degradation in performance. This is due to the fact that when server performs learning on a small dataset, larger variance in its gradient can slow down learning process, leading to a trade-off.

• **Effects of server dataset size:** Next, we study how n_0 affects learning. Since we are interested in scenarios with small n_0, we consider $n_0 \in \{200, 400, 600, 800, 1000\}$. The accuracy is plotted in Fig. 5 for the considered dataset sizes. Clearly, when more samples are available to the server for SL (FSL and DSL) or are shared among the clients (FL and DSL), the performance of all approaches improves, especially at the beginning. Moreover, except for the initial discrepancy, both

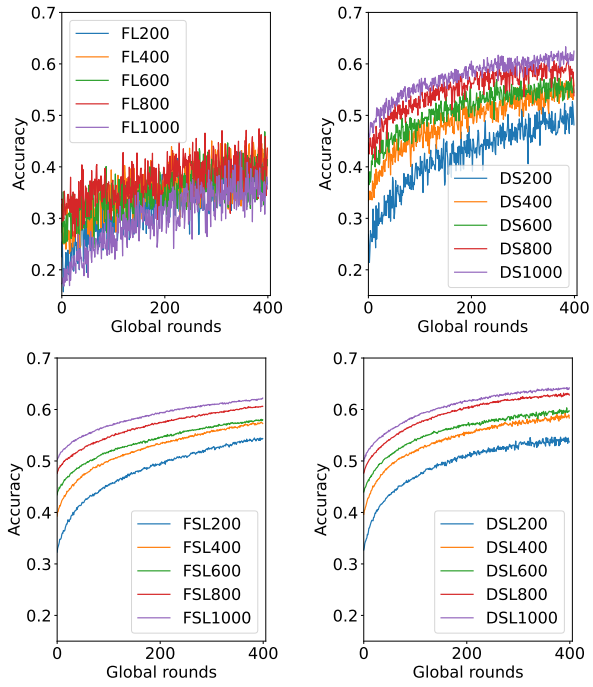

Fig. 5. Plot of accuracy for different server sample sizes $n_0 \in \{200, 400, 600, 800, 1000\}$ with $C = 2$ and $S = 3$.

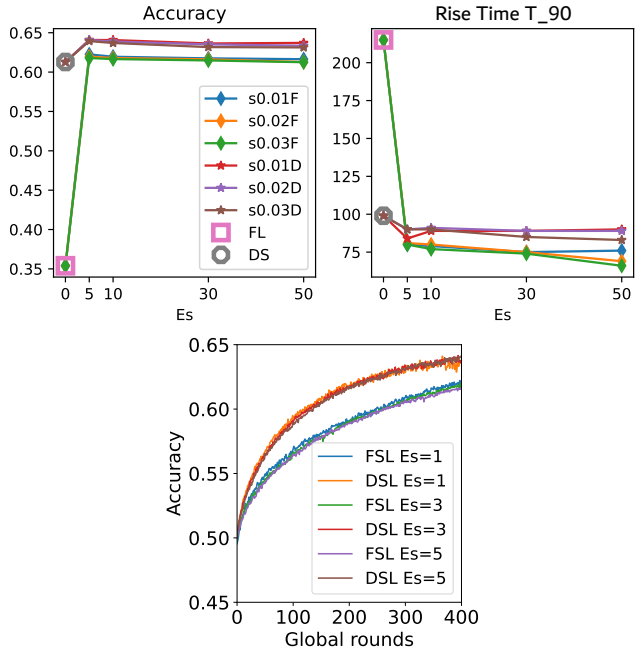

Fig. 6. Plot of accuracy and rise time (T_{90}) as a function of E_s (F = FSL, D = DSL, $N = 10$, $C = 2$ and $S = 3$ with $n_0 = 1{,}000$ server samples).

FSL and DSL approach similar performance over time.

• **Effects of server iterations and learning rate:** Fig. 6 plots the accuracy and rise time for $C = 2$ and $S = 3$ as we vary E_s for three different values of η_0. Again, the results suggest that we can select small E_s when n_0 is small without noticeable performance difference as E_s increases from 1 to 5.

IV. Conclusion

We proposed a new approach to alleviate the performance degradation in FL when clients' samples are non-IDD. Our approach provides a small local dataset to the server to allow it to perform local learning instead of sharing the data among all the clients. Our analysis revealed that this approach can improve the learning process even when the server dataset is very small. Our experimental studies demonstrated that the proposed approach provides significant benefits over conventional FL techniques and indeed only a small dataset is needed for meaningful performance gains. We also showed that our approach has a comparable performance with data sharing, and thus is more preferable in practice.

References

[1] Dimitri P Bertsekas et al. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optim. Mach. Learn.*, 2010(1-38):3, 2011.

[2] Mung Chiang and Tao Zhang. Fog and IoT: An overview of research opportunities. *IEEE Internet Things J.*, 3(6):854–864, 2016.

[3] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *IJCNN*, pages 2921–2926, 2017.

[4] Yuyang Deng, Mohammad Mahdi Kamani, and Mehrdad Mahdavi. Adaptive personalized federated learning. *arXiv:2003.13461*, 2020.

[5] Yihan Jiang, Jakub Konečnỳ, Keith Rush, and Sreeram Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv:1909.12488*, 2019.

[6] 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. *Found. Trends Mach. Learn.*, 14(1–2):1–210, 2021.

[7] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic controlled averaging for federated learning. In *37th ICML*, volume 119, pages 5132–5143. PMLR, 13–18 Jul 2020.

[8] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Tech. Rep.*, 2009.

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Mach. Learn. Syst.*, 2:429–450, 2020.

[10] Van Sy Mai, Richard J. La, and Tao Zhang. Federated Learning with Server Learning: Enhancing Performance for Non-IID Data. *arXiv:2210.02614*, 2022.

[11] 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. PMLR, 2017.

[12] Naoya Yoshida, Takayuki Nishio, Masahiro Morikura, Koji Yamamoto, and Ryo Yonetani. Hybrid-FL for wireless networks: Cooperative learning mechanism using non-IID data. In *ICC*, pages 1–7. IEEE, 2020.

[13] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated learning with non-IID data. *arXiv:1806.00582*, 2018.