A Novel Approach for Improving Generalization in Federated Learning Through Corrective Gradient Weights

Ali Mousazadeh
polytechnic university of turin
Turin, Italy
ali.mousazadeh@studenti.polito.it

Hossein Khodadadi
polytechnic university of turin
Turin, Italy
hossein.khodadadi@studenti.polito.it

Sohrab Salehi
polytechnic university of turin
Turin, Italy
s289037@studenti.polito.it
github.com/AliMousazadeh/FedOptz

July 3, 2023

Abstract

Federated Learning (FL) is a framework for training a machine learning model in which the data exists in clients and cannot be accessed directly. The model is sent to these clients and is trained locally. Then, these trained models are aggregated by a server into a global model. In this scenario, the data amongst clients may vary greatly in terms of distribution, which could pose a challenge in the aggregation phase since the global model will have a tendency to lack proper generalization. This could lead to poor performance on previously unseen data. In order to address this problem, we propose a method which takes advantage of the gradients of clients with respect to their local data in order to estimate the possible effect of each client's model on other clients. We formulate a constrained optimization problem to find the optimal correction terms to be added to the FedAvg baseline model, which adjusts the global model such that clients that have a better estimated loss reduction on other clients will have an increased say on the direction the global model will take. We test the performance of our method against the FedAvg baseline on the Federated EMNIST dataset. We find that our method outperforms the FedAvg baseline both in terms of convergence speed and mean accuracy.

1 Introduction

While new learning techniques emerge, they usually become more data-hungry. Therefore lack of sufficient data is felt more than before to the extent that data is called the new oil of our era. Unfortunately, due to privacy issues, all the potential data is not accessible. Most of the useful data is distributed as distinct islands including cross-silo data e.g. data distributed across hospitals, companies, or a large number of devices. Federated learning introduced by McMahan et al. [14] trains the data with decentralized access. Models are trained iteratively across many federated rounds. For each round, every participating device receives an initial model from a server, performs optimization on its local training data, and sends back the trained model. The server then aggregates all the models from the participating clients and updates the starting model [7]. Despite exten-

sive efforts to tackle the challenges of FL, there is still room for progress on the following fronts:

1.1 Domain Generalization

In FL, distributed clients collect their local data independently, so the data-set of each client may naturally form a distinct source domain. In practice, the model trained over multiple source domains may have poor generalization performance on unseen target domains [22].

1.2 Data heterogeneity

Devices frequently generate and collect data in a non-identically distributed manner across the network. For example, mobile phone users have varied use of language in the context of a next word prediction task. Moreover, the number of data points across devices may vary significantly, and there may be an underlying structure present that captures the relationship amongst devices and their associated distributions. This data generation paradigm violates frequently-used independent and identically distributed (IID) assumptions in distributed optimization, increases the likelihood of stragglers, and may add complexity in terms of modeling, analysis, and evaluation [11].

2 Related Work

Federated learning is a fast-growing research field which has attracted the attention of many researchers. Relevant works have pursued the following directions.

2.1 Data Heterogeneity

In the field of federated learning, McMahan et al. [14] proposed a framework that enables joint training on unbalanced and non-IID local data-sets while minimizing communication costs. This work laid the foundation for federated learning optimization, which involves local client training stages and global server updates. [6] shows that performance degrades as distributions of the client data differ more, and propose a mitigation strategy via server momentum. [7] also develops two new algorithms (FedVC, FedIR) that intelligently resample and reweight over the client pool, bringing large improvements in accuracy and training stability while facing non-IID data. [3] links the model's lack of generalization capacity to the sharpness of the loss surface. Through seeking parameters in neighborhoods having uniform low loss, the model tries to find flatter minima. Finally [13] realizes there exists a greater bias in the classifier than other layers in the client models, and the classification performance can be significantly improved by post-calibrating the classifier after federated training. In practical scenarios [10] challenges such as local inconsistency and client drifts caused by heterogeneous over-fitting emerges that will be discussed in following sections.

2.2 Local Consistency

To address the issue of local inconsistency, Sahu et al. [16] studied non-vanishing biases and introduced a prox-term regularization. FedProx utilizes bounded local updates and penalizes parameters to ensure consistency. Wang et al. [20] introduced local correction techniques. Durmus et al. [1] (FedDyn) proposes a dynamic regularizer for each device at each round, so that in the limit the global and device solutions are aligned. [17] FedSpeed applies the prox-correction term on the current local updates to efficiently reduce the biases introduced by the prox-term.

2.2.1 Client Drifts

Karimireddy et al. [8] highlighted client drifts caused by local overfitting and proposed SCAFFOLD using variance reduction techniques to mitigate drifts. Wang et al. [21] introduced a momentum-based approach to improve generalization performance. Gao et al. [5] introduced a drift correction term to the penalized loss functions. Finally [17] merges the vanilla stochastic gradient with a perturbation computed from an extra gradient ascent step in the neighborhood, thereby alleviating the issue of local overfitting.

2.2.2 Domain Generalization

In the context of federated learning, the focus on domain generalization involves training a model with good performance on unseen target domains. Previous works in domain generalization, like FedProx [11] assume centralized settings with access to data from all source domains. However, these solutions cannot be directly applied to federated learning due to the constraint of not exposing source domain data to the server. FedDG [12] enables each client to exploit multi-source data distributions and transmits the distribution information across clients through an effective continuous frequency space interpolation mechanism. FedADG [22] employs the federated adversarial learning approach to measure and align the distributions among different source domains via matching each distribution to a reference distribution. FedSR [15] learns a simple representation of the data by enforcing an L2-norm and a conditional mutual information regularizer to encourage the model to only learn essential information.

2.2.3 Domain Adaptation

Unsupervised Domain Adaptation (UDA) techniques aim to learn an ML model from one or multiple source domains for performance on a different, but related, target domain [2]. While UDA approaches assume access to unlabeled target domain data, their application in federated learning is limited as the test dataset's participation in the training process is not permitted.

In this work, we propose a novel method that takes advantage of the gradients of clients in order to estimate the similarity of clients to each other. Using these gradients, we define a constrained optimization problem which tries to maximize the reduction in loss of each client given an update which combines the gradients of all clients. Our goal with this method is to address the differences in client models which may result from different distributions of data. The aim is the correct the average model in such a way that more "useful" clients with better generalization capabilities get an increased say in the direction of the global model.

3 Methodology

In this section, we describe the proposed method and argue for its benefits in addressing data heterogeneity issues and achieving better generalization.

Consider the baseline FedAvg scenario in which during each round, some clients are randomly selected. They then receive the global model, train it using their local data, and send their models to the server. Let the loss function for client i be defined as:

$$L_i(W_i, t, y) = \frac{1}{M} \sum_{p=1}^{M} \Lambda(t_p, y_p) \tag{1}$$

where Λ is some non-negative criterion such as cross-entropy and M is the number of samples inside client i. In our method, we ask each client to calculate the gradient of its loss function with respect to its parameters given the local model it has trained during this round and its data. We denote this gradient by $\frac{\partial L_i}{\partial W_i}$ where i indicates the *ith* client selected during this round and W_i is a flattened vector containing all the trainable parameters of the client's model. We then consider a first-order Taylor expansion for the loss of client i in a small neighborhood around W_i:

$$\Delta L_i = \frac{\partial L_i}{\partial W_i} . (W - W_i) = \frac{\partial L_i}{\partial W_i} . \Delta W \tag{2}$$

Where . denotes the dot product. For any ΔW we choose, we can estimate ΔL_i in this manner. However, for the estimation to hold merit, $||\Delta W||$ cannot be large, as (2) is only valid in the neighborhood of W_i. This limits the kinds of corrections we can reliably apply to W_i. Ideally one might want to measure the loss of $W = \sum_{i=1}^{n} x_i W_i$ on all clients such that $\sum_{i=1}^{n} x_i = 1$ and $x_i \geq 0 \ \forall i$. However, this does not work in practice as W will be too far from each client for (2) to apply.

We propose the following ΔW:

$$\Delta W = -\sum_{j=1}^{n} x_j \frac{\partial L_j}{\partial W_j} \tag{3}$$

where x_j is a weight assigned to the gradient of the *jth* client and n is the number of clients per round. Firstly, ΔW is the same across all clients which leads

to an easier optimization problem as we will see later. Secondly, since this correction term depends on the gradients of clients, we do not need the conditions $\sum_{i=1}^{n} x_i = 1 \;\; x_i \geq 0 \; \forall i$ to maintain the scale of W_i. This means we can limit $||\Delta W||$ which may lead to small or even negative weights for a particular client while still being able to use (2) to get a good estimate of the effect of ΔW on the client losses.

Combining (2) and (3), we have:

$$\Delta L_i = -\frac{\partial L_i}{\partial W_i}.(\sum_{j=1}^{n} x_j \frac{\partial L_j}{\partial W_j}) \tag{4}$$

which is an estimation of the change in the loss of client i given the weighted sum of gradients of all clients. The sum of the changes in the losses of all clients therefore becomes:

$$\Delta L = -\sum_{k=1}^{n} (\frac{\partial L_k}{\partial W_k}.(\sum_{j=1}^{n} x_j \frac{\partial L_j}{\partial W_j})) \tag{5}$$

rearranging the terms, we get:

$$\begin{aligned} \Delta L = -\sum_{i=1}^{n} x_i \frac{\partial L_i}{\partial W_i}.(\sum_{j=1}^{n} \frac{\partial L_j}{\partial W_j}) = \\ -\sum_{i=1}^{n} x_i (D_i.D) = -\sum_{i=1}^{n} x_i m_i \end{aligned} \tag{6}$$

by setting $D_i = \frac{\partial L_i}{\partial W_i}$, $D = \sum_{j=1}^{n} \frac{\partial L_j}{\partial W_j}$, and $m_i = D_i.D$. Ideally, our goal is to minimize ΔL or equivalently maximize:

$$J(x) = -\Delta L = \sum_{i=1}^{n} x_i m_i \tag{7}$$

However, since (7) is a linear function of x, its maximum is infinity without some sort of limit on x. Furthermore, it is important to consider the validity of the first order Taylor approximation in the desired neighborhood. If the gradient of each client with respect to its model parameters is bounded and well-behaved, the parameters of the model are bounded, the neighborhood defined by ΔW is sufficiently small, and the loss function in this neighborhood behaves quite linearly with respect to small changes in model parameters, then it would be reasonable to assume that the linear approximation is accurate, although some error would still remain. Of these conditions, the ones concerning the linear and smooth behavior of the loss function within some small neighborhood of the trained state can be addressed to some extent through choosing appropriate training hyper parameters such as batch size [9], learning-rate, and momentum. Tuning these hyper parameters in such a way that would lead to better generalization and smoother minima makes the loss function more well-behaved, which in turn helps the first-order Taylor approximation to be more reliable. If the well-behaved conditions were to be explicitly satisfied, we'd need the loss function of each client to be Lipschitz differentiable [19] having $K \leq ||D_i||$ where K is the Lipschitz constant. This is a rather strong assumption and furthermore, it is not easy to measure the Lipschitz constant of neural networks in practice [18] specially for more complex networks. Generally it's more convenient to set appropriate hyper parameters which lead to better convergence rather than checking if the loss functions are Lipschitz differentiable for each client in advance.

We now consider the condition regarding the neighborhood spanned by ΔW to be small. Ideally, we would like to define a constraint which limits the span of this neighborhood, whilst simultaneously leading to an easy constrained optimization problem that can be solved relatively quickly and preferably in closed form. We should also consider that after finding a solution for a constrained optimization problem, we must also check if the solution is a maximum, minimum, or saddle point. It would be ideal if the chosen constraint in combination with (7) would lead to an easily verifiable maximum as well. Furthermore, it would be desirable if the values of x_i vary between clients and do not remain constant, as constant weights would mean taking simple full-batch steps which typically aren't good for generalization.

After defining an appropriate constraint and finding the optimal values for x_i, the way we create the new global model for the next round is to average the corrected client models which would lead to:

$$W_{global} = W_{FedAvg} - \sum_{i=1}^{n} x_i D_i \tag{8}$$

Initially, one could consider the most direct constraint which enforces the span of the neighborhood, which is:

$$||\Delta W||^2 - c^2 = 0 \tag{9}$$

where c is some chosen small constant. If the conditions for the linear approximation being fairly accurate are satisfied, we have the following inequality which results from (2):

$$||\Delta L_i|| < ||\Delta W||.||D_i|| \tag{10}$$

Therefore given some desired upper bound on the change in loss denoted by ϵ, we have:

$$||\Delta W|| < \frac{\epsilon}{||D_i||} \tag{11}$$

It would be reasonable to choose ϵ in such a way that would scale with the value of the loss function at the point of estimation. Leading to:

$$||\Delta W|| < \frac{\alpha L_i}{||D_i||} \tag{12}$$

where $\alpha \in [0, 1]$ in this case determines the percentage of the desired upper bound of change in loss considering the loss of clients at the point of estimation. Since this must be considered for each client, we can pick:

$$c = min \frac{\alpha L_i}{||D_i||} \ \forall i \ \ , c = \alpha c^* \tag{13}$$

setting $c^* = min \frac{L_i}{||D_i||} \ \forall i$.

Therefore, the constrained optimization task becomes:

$$max \ \ f(x) = \sum_{i=1}^{n} x_i m_i \tag{14}$$

such that:

$$g(x) = ||\Delta W||^2 - c^2 = 0 \tag{15}$$

The Lagrangian is:

$$\mathcal{L} = f(x) - \lambda g(x) = \sum_{i=1}^{n} x_i m_i - \lambda(||\Delta W||^2 - c^2) \tag{16}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial x_i} &= m_i - 2\lambda D_i(\sum_{j=1}^{n} x_j D_j) \\ &= D_i(\sum_{j=1}^{n} D_j) - 2\lambda D_i(\sum_{j=1}^{n} x_j D_j) = 0 \end{aligned} \tag{17}$$

Setting $x_j = \frac{1}{2\lambda}$ for all j, we get:

$$\frac{\partial \mathcal{L}}{\partial x_i} = D_i(\sum_{j=1}^{n} D_j) - D_i(\sum_{j=1}^{n} D_j) = 0 \tag{18}$$

Therefore:

$$x_i = x_j = \frac{1}{2\lambda} \tag{19}$$

Inserting this into (15), we get:

$$\frac{1}{4\lambda^2} \sum_{k=1}^{n} \sum_{p=1}^{n} D_k.D_p = \frac{1}{4\lambda^2} D.D = c^2 \tag{20}$$

$$\lambda = \pm \frac{\sqrt{D.D}}{2c} \tag{21}$$

Inserting this into (19) leads to:

$$x_i = \pm \frac{c}{\sqrt{D.D}} \tag{22}$$

where the positive and negative answers lead to the maximum and minimum of (15) respectively. Therefore:

$$x_i = \alpha \frac{c^*}{\sqrt{D.D}} \tag{23}$$

Due to the linear nature of the optimization function and the quadratic constraint, the solutions are mirrored by their sign which denote the maximum and the minimum. Therefore we do not need to check for saddle points. Technically the constraint (15) does not need to be an equality and in fact, it's < 0 since any point for which $||\Delta W||$ is sufficiently small can be a possible solution. However, since this is a maximization problem, the objective function is linear, the constraint is convex, and λ is

positive at the maximum, that means relaxing the constraint would lead to a higher maximum. This in turn means the maximum value of the objective function for any $c_0^2 < c^2$ is going to be less than the maximum of the objective function given c^2. Therefore, we do not need to consider any point on the interior of the condition and we can replace < 0 with $= 0$ as we have done in (15).

While the solutions found in this formulation of the problem are valid, they give equal weight to all the clients regardless of the similarity of their gradients. This happens due to the nature of the constraint (15). This would effectively mean correcting the FedAvg model by the sum of the full-batch gradients of clients, which isn't desirable. We would like to keep the quadratic nature of the constraint since it is more likely to lead to an easily solvable problem, but we would also like the constraint to lead to a solution which considers the effects of the clients on each other.

We consider a different constraint of the following form:

$$g(x) = (\sum_{i=1}^{n} x_i^2 ||D_i||^2) - c^2 = 0 \tag{24}$$

where c is the same as in (13). Unlike (15), this constraint no longer enforces $||\Delta W||$ to be small enough such that the changes in the loss are of the order αL_i. It still limits $||\Delta W||$, however the hyper parameter α has lost the clear interpretation it had in the previous constraint. This constraint encourages giving smaller weights to clients with bigger local gradient norms, therefore it pushes the correction term in favor of the clients with harder to minimize losses. Furthermore, since it involves the term x_i^2, it allows the weight for a client to be negative as well. As we will see, this has the interesting outcome that if the estimated bad effect a client has on others outweighs the estimated good effect it has on its local loss, it will be given a negative weight in the correction term to run back a portion of its contribution to the FedAvg model.

Given (24), The Lagrangian becomes:

$$\mathcal{L} = \sum_{i=1}^{n} x_i m_i - \lambda(\sum_{i=1}^{n} x_i^2 ||D_i||^2 - c^2) \tag{25}$$

We have:

$$\frac{\partial \mathcal{L}}{\partial x_i} = m_i - 2\lambda x_i ||D_i||^2 = 0 \tag{26}$$

$$x_i = \frac{1}{2\lambda} \frac{m_i}{||D_i||^2} \tag{27}$$

Inserting (27) into (24) we get:

$$\lambda = \pm \frac{1}{2c} \sqrt{\sum_{j=1}^{n} (\frac{m_j}{||D_j||})^2} \tag{28}$$

Similar to the previous constraint, the positive value for λ is the one leading to a maximum. This leads to the following value for x_i:

$$x_i = \alpha \frac{c^*}{\sqrt{\sum_{j=1}^{n} (\frac{m_j}{||D_j||})^2}} \frac{m_i}{||D_i||^2} \tag{29}$$

Through inspecting (29) we see that for a solution to exist, we must have $\forall i : \ ||D_i|| \neq 0$. We also see that x_i relates linearly to α, therefore the hyper parameter α will linearly scale the weights of clients. Setting $\alpha = 0$ will reduce the correction term to 0 and therefore the global model will be updated with the FedAvg model. The appropriate value for α should be set in the same manner as with any other hyper parameter, which would be through checking the convergence on the train data or the accuracy on the validation set.

As briefly mentioned earlier, we have:

$$sign(x_i) = sign(m_i) = sign(D_i.D) \tag{30}$$

Therefore the sign of a client's weight is determined by the dot product of its gradient with the sum of the gradients of all clients. This means the contribution of a client to the FedAvg model is reversed to some extent determined by α if its gradient is dissimilar to the average gradients of the sampled clients during each round.

4 Experiments

The following section of our work is divided to two subsections: preliminary tests are done to study the

FedAvg Experiment			
Epochs	clients/R	NIID Acc	IID Acc
1	5	83%	85.3%
1	10	83%	86%
1	20	82.9%	84.9%
5	5	84.3%	85.2%
10	5	84.1%	84.8%

Table 1: FedAvg Table

effect of different hyper-parameters to analyse their influence on accuracy of the server model on test clients. All the tests are performed on 100% of data except for IID version of experiments which was done on 25% of data. The second subsection will study the performance of our proposed method in practice. Note that the definition of the dataset and the CNN model utilized in this study are provided in Appendix B.

4.1 Preliminary Experiments

This subsequent showcases the results of one preliminary task, while the entire collection of experiments can be found in Appendix A.

4.1.1 FedAvg Experiment

According to the experimental results presented in Table 1, when we increase the number of epochs or the number of clients per round in the training process, the model's accuracy initially improves. However, after a certain point, increasing these factors begins to negatively impact the model's generalization across all clients data. This effect is particularly prominent when dealing with non-IID data. Considering the 86.89% accuracy of the centralized baseline model at the best case scenario the achieved accuracy for non-IID is 2.59% and for IID is 1.43% lower. Note that the accuracy of the model was evaluated on the last call of the test function on test clients after 1000 rounds of communications. The corresponding plots analysing the influence of number of epochs and client per round can be found in Figures 1 and 2.

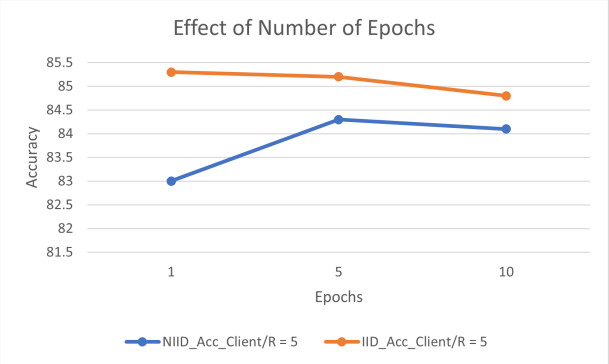

Figure 1: Effect of number of epochs on the accuracy of model in IID and NIID setting

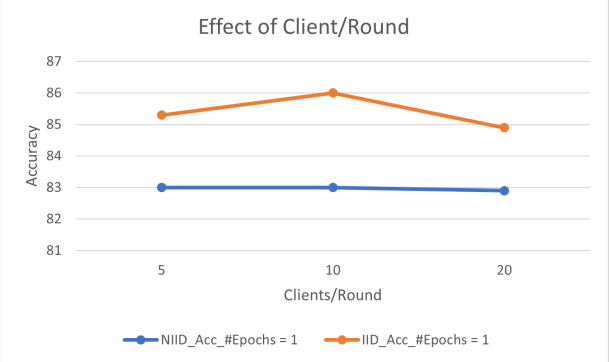

Figure 2: Effect of number of clients per round on the Accuracy of model in IID and NIID setting

4.2 Proposed Method Experiments

In this section we discuss the results of some experiments on our proposed method. We compare the mean test accuracy of our method against the baseline FedAvg across different random seeds and values of α. In particular, we check how well these methods converge improve in terms of mean test accuracy, and how much their mean test accuracy differs when sufficient convergence has been reached.

We see in Table 2 and Table 3 that as we increase the value of α, we typically get better convergence in both the IID and non-IID cases. As the FedAvg ($\alpha = 0$) and the proposed method reach the final

rounds of training, the gap between the two becomes smaller. Our experiments show that the gap between the FedAvg and centralized accuracy on the EMNIST dataset is quite small, therefore it is not easy to measure an improvement for a given federated learning method. Nonetheless, our experiments show that the proposed method does perform better, even if by a small margin.

In the non-IID case, higher values of α can be selected and the correction can be made more aggressively. However it's typically a better idea to go for smaller values of α in the IID case. While we get a boost in mean accuracy as we increase α, it starts to oscillate as we increase α by too much. The value for this hyper parameter depends on several factors, including but not limited to: model and data complexity, the distribution of labels and inputs inside clients, and uniform or non-uniform access to clients. A full study of the effects of all possible factors on α is beyond the scope of this report, however in any case its value can be tuned by checking the performance on the validation set.

Round/α	0	0.1	0.2	0.3	0.4
171-200	78.30 0.25	78.64 0.24	78.87 0.16	**79.11** 0.16	79.04 0.20
901-1000	83.23 0.15	83.22 0.18	83.22 0.20	83.28 0.14	**83.35** 0.08

Table 2: Mean test accuracy of the proposed method on the non-IID FEMNIST dataset for different choices of α. Top number in each cell is the mean test accuracy across 3 different seeds, while the bottom number is the unbiased standard deviation. $\alpha = 0$ turns the method into FedAvg.

Round/α	0	0.025	0.05	0.075	0.1
171-200	80.20 0.76	80.27 0.73	80.33 0.70	80.41 0.67	**80.43** 0.63
901-1000	83.64 0.38	83.69 0.42	83.73 0.40	83.78 0.42	**83.83** 0.47

Table 3: Mean test accuracy of the proposed method on the IID FEMNIST dataset for different choices of α. Top number in each cell is the mean test accuracy across 3 different seeds, while the bottom number is the unbiased standard deviation. $\alpha = 0$ turns the method into FedAvg.

5 Conclusion

In this report, we introduced a novel method which corrects the FedAvg model by a weighted sum of the gradients of clients. These weights are calculated as the solution to an optimization problem which aims to minimize the local losses of clients through this correction, given a condition and hyper parameter which limits the strength of the correction. We have shown that our method performs better than the FedAvg model on the Federated EMNIST dataset in terms of convergence and accuracy.

References

[1] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021. 2

[2] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006. 3

[3] Debora Caldarola, Barbara Caputo, and Marco Ciccone. Improving generalization in federated learning by seeking flat minima. In *European Conference on Computer Vision*, pages 654–672. Springer, 2022. 2

[4] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020. 10

[5] Liang Gao, Huazhu Fu, Li Li, Yingwen Chen, Ming Xu, and Cheng-Zhong Xu. Feddc: Federated learning with non-iid data via local drift decoupling and correction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10112–10121, 2022. 2

[6] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019. 2

[7] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Federated visual classification with real-world data distribution. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 76–92. Springer, 2020. 1, 2

[8] 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*, pages 5132–5143. PMLR, 2020. 2

[9] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. 4

[10] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020. 2

[11] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020. 2, 3

[12] Quande Liu, Cheng Chen, Jing Qin, Qi Dou, and Pheng-Ann Heng. Feddg: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1013–1023, 2021. 3

[13] Mi Luo, Fei Chen, Dapeng Hu, Yifan Zhang, Jian Liang, and Jiashi Feng. No fear of heterogeneity: Classifier calibration for federated learning with non-iid data. *Advances in Neural Information Processing Systems*, 34:5972–5984, 2021. 2

[14] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017. 1, 2

[15] A Tuan Nguyen, Philip Torr, and Ser Nam Lim. Fedsr: A simple and effective domain generalization method for federated learning. *Advances in Neural Information Processing Systems*, 35:38831–38843, 2022. 3

[16] Anit Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. 12 2018. 2

[17] Yan Sun, Li Shen, Tiansheng Huang, Liang Ding, and Dacheng Tao. Fedspeed: Larger local interval, less communication round, and higher generalization accuracy. *arXiv preprint arXiv:2302.10429*, 2023. 2

[18] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018. 4

[19] Han Wang, Siddartha Marella, and James Anderson. Fedadmm: A federated primal-dual algorithm allowing partial participation. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 287–294. IEEE, 2022. 4

[20] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020. 2

[21] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. Slowmo: Improving communication-efficient distributed sgd with slow momentum. *arXiv preprint arXiv:1910.00643*, 2019. 2

[22] Liling Zhang, Xinyu Lei, Yichun Shi, Hongyu Huang, and Chao Chen. Federated learning with domain generalization. *arXiv preprint arXiv:2111.10487*, 2021. 2, 3

6 Appendix A: Experiments

6.0.1 Centralized Training: Upper Bound

The accuracy of the centralized model of the experiment was fine tuned by performing a grid search on different learning rates, batch sizes, momentums and epochs. The result was retrieved based on the Table 4. The test was repeated on three different seeds on the best achieved hyper-parameters to make sure the result of our study is independent from initial values in the weight parameters of the model. Retrieved accuracy for the best hyper-parameters were 86.89%, 86.32% and 86.11% respectively.

6.0.2 Probability of Selection

In this experiment we ran the FedAvg experiment (both in IID and non-IID scenarios) with 10 clients

Centralized Fine Tuning				
Epochs	l_rate	Momentum	b_size	Acc
10	.001	.9	64	86.71%
10	.001	.9	128	86.39%
10	.001	.95	64	86.71%
10	.001	.95	128	86.64%
10	.001	.99	64	86.50%
10	.001	.99	128	86.78%
10	.0001	.9	64	84.94%
10	.0001	.9	128	83.46%
10	.0001	.95	64	85.83%
10	.0001	.95	128	84.89 %
10	.0001	.99	64	86.50 %
10	.0001	.99	128	86.47%
20	.001	.9	64	86.41%
20	.001	.9	128	86.50%
20	.001	.95	64	86.38%
20	.001	.95	128	86.5%
20	.001	.99	64	84.89%
10	**.001**	**.95**	**32**	**86.89%**
10	.001	.95	16	86.63%

Table 4: Centralized Fine Tuning

per round. We have modified the function selecting the clients involved at each training round and assigned a different probability to each client to be selected based on table 5. We perform comparisons between: **(I) rows of table 5, (II) results of experiments after and before applying probability of selection (table 5 and table 1)**.
(I): In the first row scenario we divide the clients into two groups of proportions 10-90%. We select from the 10% group with probability of 0.5, while choosing the other 90% group with probability 1 - 0.5 = 0.5. This exaggerates the availability of those 10% which makes them influence the training process more. In the second row scenario the groups are divided with the ratio 30-70% with the probability of selection from the 30% group massively reduced to 0.0001. This has the effect of making the 30% group unavailable most of the times during training, greatly reducing their effect on the global model.
(II): Reasonably the accuracy of the model for the

Probability of Selection		
conditions	NIID Acc	IID Acc
10% clients, p = 0.5	82.60 %	83.63 %
30% clients, p = .0001	83.33 %	82.37 %

Table 5: Probability of Selection

conventional FedAvg provided in Table 1 should be higher since in the FedAvg setting each and all of the clients have the same chance to participate in training process, while in the Probability of Selection setting some of them are selected much more or much less than others. Measurements in Table 5 demonstrate this result.

6.0.3 Power Of Choice

Recent works [4] showed how selecting clients in a smarter way, i.e. according to some criterion, can lead to better results in terms of final performance and speed of convergence. For instance, [4] shows that biasing client selection towards clients with higher local loss achieves faster error convergence. We have run the experiment with Power-of-choice in the IID and non-IID scenarios with 10 clients per round varying the value of d, which is the number of clients with higher loss among chosen clients in each round.

Based on Table 6 the accuracy of the model in Power of Choice setting comparing to Probability of Selection setting should be more, since in each train round only those clients which have more loss are chosen. Furthermore, in each train round all the clients have the same chance to participate in training process. If we want to compare results of the Power of Choice with conventional FedAvg, reasonably the accuracy should be higher since the clients are being chosen more deliberately than uniformly at random.

6.0.4 Domain Generalization in FL

In the experiments of this section we have tried to address the following question: **"What happens when new domains, e.g. new users, appear?**

Power of Choice		
d	NIID Acc	IID Acc
2	82.46 %	79.05 %
5	82.75 %	80.51 %
8	84.36 %	84.66 %
10	84.15 %	85.77 %

Table 6: Power of Choice

Is the model able to generalize to unseen target data as well?"

(I) without accounting for generalization: we have built RotatedFEMNIST in which images are rotated counterclockwise with an angle of 0°, 15°, 30°, 45°, 60° and 75° to form six domains, identified as M0, M15, M30, M45, M60, M75. We have selected 1000 random clients and divided them in 6 groups, then we applied to each group one of the 6 rotations. Note that in this experiments 1000 rotated client images were fed to our model in addition to other clients without rotation. The accuracy of the model without generalizing on unseen domains for FedAvg is measured as 79.3% while centralized model accuracy was 86.47%.

(II) with accounting for generalization: following the "leave-one-domain-out" strategy, we chose one domain as the target domain, trained the model on all remaining domains, and evaluate it on the chosen domain. Based on the results in Table 7, the accuracy of the models in all settings initially increases from 0°to 45°of rotation, then it decreases from 45°to 75°. This behaviour is likely due to the fact that when the test domain is M0 and has no rotation, the training dataset consists of domains with rotation angles that are further away from 0. In contrast, when the desired domain is M45, the training data includes rotations like 30 or 60 degrees which are closer to 45 degrees. Therefore the accuracy is higher since these rotations are closer to the desired domain.

6.0.5 FedSR

The FedSR method aims to learn a simple representation of the data for better generalization. In particular, it enforces an L2-norm regularizer on the representation and a conditional mutual information regularizer between the representation and the data given the labels, to encourage the model to only learn essential information. Experiments of the previous section were repeated for FedSR. The results of this task are presented in Table 8. They demonstrates slight improvement in the accuracy of the models in this scenario.

Leave One Domain Out		
T_Domain	C_Acc	FedAvg_NIID
M0	45.24%	53.1%
M15	81.31%	66.1%
M30	75.81%	67.4%
M45	80.97%	67.6 %
M60	76.91%	65.5 %
M75	62.53%	51.9 %

Table 7: Leave One Domain Out

Leave One Domain Out: FedSR		
T_Domain	FedAvg_NIID	FedSR_NIID
M0	53.1%	53.3%
M15	66.1%	67.7%
M30	67.4%	67.9%
M45	67.6%	68.1%
M60	65.5%	66.2%
M75	51.9%	52.4%

Table 8: Leave One Domain Out: FedSR

7 Appendix B

7.1 Dataset

The Federated Extended MNIST (FEMNIST) dataset is a variant of the Extended MNIST dataset specifically designed for federated learning. The dataset includes 62 classes, comprising the digits (0-9) and both uppercase and lowercase English letters (A-Z and a-z). Similar to the original MNIST dataset, the FEMNIST images are typically 28x28 pixels in size, with each pixel representing the grayscale intensity of the image.

7.2 Model

The CNN model utilized in our study, consists of two main components, convolutional layers and fully connected layers. The input to the CNN is a grayscale image with a single channel. The first convolutional layer takes this input and applies a set of 32 filters, each with a size of 5x5. These filters capture different patterns and features in the image. The resulting feature maps go through a ReLU activation function, which introduces non-linearity to the network. After the activation, a 2x2 max pooling layer is applied to reduce the spatial dimensions of the feature maps. The process is repeated with the second convolutional layer, which consists of 64 filters of size 5x5. Again, ReLU activation and max pooling are applied to the output feature maps. Following the convolutional layers, the feature maps are flattened into a vector, allowing them to be processed by the fully connected layers. The first fully connected layer has 2048 neurons. It maps the flattened features to a higher-dimensional space, allowing for more expressive representations. The second and last fully connected layer maps the features to the number of classes in the task (62 classes assuming letters). It performs the final mapping from the learned features to the predicted class probabilities.