Lay Importance on the Layer: Federated Learning for Non-IID data with Layer-based Regularization

Eleftherios Charteros and Iordanis Koutsopoulos
Department of Informatics
Athens University of Economics and Business, Greece

***Abstract*—We propose Federated Learning with Layer-Adaptive Proximation (FedLap), a new class of algorithms to tackle non-IID data. The novelty is a new regularization term in the local (client) loss function which generalizes that of FedProx and captures divergence between global and local model weights of each client, at the level of Deep Neural Network (DNN) layers. Thus, the weights of different layers of the DNN are treated differently in the regularization function. Divergence between the global and local models is captured through a dissimilarity metric and a distance metric, both applied to each DNN layer. Since regularization is applied per layer and not to all weights as in FedProx, during local updates, only the weights of those layers that drift away from the global model change, while the other weights are not affected. Compared to FedAvg and FedProx, FedLap achieves 3-5% higher accuracy in the first 20 communication rounds, and up to 10% higher accuracy in cases of unstable client participation. Thus, FedLap paves the way for a novel layer-aware class of algorithms in distributed learning.**

I. Introduction

In Federated Learning (FL), a local model is trained on each device (client) with its local data, and then the local model weights are sent to the server. There, the models are combined, and the updated global model is sent back to the devices to initialize a new training round on their local datasets. This exchange is repeated until convergence to a global model. An important challenge in FL model training is that of *statistical heterogeneity of client data*, since datasets of different clients are drawn from different probability distributions.

Given an abstract representation of a data point as $(\mathbf{x}, y)$, where $\mathbf{x}$ denotes the feature (attribute) vector and y denotes a label, there exist various scenarios for non-independent and identically distributed (non-IID) data distributions $P_i(\mathbf{x}, y)$ across different clients i. First, different clients i may have different feature distributions $P_i(\mathbf{x})$ and the same distributions $P_i(y \mid \mathbf{x})$. For example, each client may have photos of a different race or different color of cats. Second, different clients may have different label distributions $P_i(y)$ but roughly the same $P_i(\mathbf{x} \mid y)$. For example, a client may have 80% of the photos in her dataset labeled as dogs and 20% as cats, while another client may have different proportions; however given y, $P_i(\mathbf{x} \mid y)$ is roughly the same. Or, some hospitals may have labels from different diseases and/or different number of patient records on them. Third, different clients i may have the same distribution $P_i(\mathbf{x})$ but different distributions $P_i(y \mid \mathbf{x})$. For example, different employees in the same working environment context have different levels of satisfaction. Finally, clients may have the same distribution $P_i(y)$ and different distributions $P_i(\mathbf{x} \mid y)$; e.g. the building style of the same type of building may differ in different countries.

While Federated Averaging (FedAvg) succeeds with IID data, it struggles with a large number of communication rounds, slow convergence, and poor quality of the model when it comes to Non-IID data [1]. An important improvement over FedAvg is Federated Proximation (FedProx) [1], which introduces an L_2-regularization term in the client loss functions, which prevents the local model weights from diverging from global ones. FedProx, now a widely adopted FL algorithm, inherits the rationale of FedAvg and is lightweight to implement on tiny devices. Further, it comes with no additional communication overhead and computational burden for the server. Some works propose additional mechanisms on top of the basic FL algorithm, such as model compression, client clustering, and client selection, which create additional complexity. Other works assume that a dataset exists at the server, and that it is partly shared with clients, which is a strong assumption. In all works, an important overlooked aspect is that of taking a deeper look into the Deep Neural Network (DNN) model itself, and leveraging a fine-grained view of its structure in model training.

In this work, we introduce a new class of methods for FL training that explicitly take into account the DNN architecture and its layers. We train *a single* global model in a supervised-learning setting, out of non-IID client datasets. The key idea is a novel regularization term in the client training loss functions that consists of a summation of terms. Each such term corresponds to one DNN layer and captures the divergence between the global and local model weights of each client, *at the level of that DNN layer*. The divergence between the global and local models is captured through a dissimilarity metric that emanates from cosine similarity, and through models' Euclidean distance, both applied to each DNN layer.

The new term replaces the one of FedProx and it is a generalization of it. As a result, the weights of different layers are treated differently in the regularization function. Since regularization is applied per layer and not universally to all weights as in FedProx, we can selectively control and change the weights of those layers that drift away from the global model, without affecting the weights of other layers. We verify the superior performance of FedLap over FedAvg and FedProx in terms of accuracy and convergence speed, by experimenting with the MNIST, FashionMNIST, and CIFAR-10 datasets in various settings and cases of unstable client participation.

II. Related Work

There exist several works that tackle non-IID data in FL [2], as evidenced by surveys [3]. In [4], FedAvg is shown to converge with rate $O(1/T)$, where T is the number of global iterations, for non-IID data, strongly convex local loss functions, and uniform client sampling without replacement.

When clients have non-IID data, the global model is the aggregate of local models whose local optima are far away from each other's. Thus, model averaging makes the global model move away from the true global optimum. This is the *model drift* problem. In order to mitigate it, FedProx [1] adds a convex regularization term in the local loss function, namely the ℓ_2-norm of the difference between the local and global models. The Model Contrastive Learning (MOON) approach [5] employs the representation of a model, namely the feature vector that emerges before the output layer. It enhances the loss in the regularization function with a term so that the agreement between the representation learned by the current local model and that learned by the global model is maximized.

A number of works use *client clustering*. In [6], clients are clustered into groups based on their data distribution. An affinity propagation method is employed, with cosine similarity as distance metric between models of different clients, so that a different model is built for each cluster of clients. In another work [7], clients are clustered based on the similarity of their local models and the global model. Clusters are trained separately and in parallel.

Another line of works use *client selection* to mitigate the effects of non-IID data. The work [8] uses a probabilistic node selection scheme to improve convergence speed of FedAvg for non-IID data. The inner product between the local gradient of each client and the global gradient is used to tune the selection probability for the training process at each round, and thus clients whose gradients are more aligned with the global one are selected to participate with higher probability. The work [9] uses reinforcement learning to learn the subset of clients to participate in each round, having as state the global and local models and as reward the test accuracy.

Other works focus on improvements at the *model aggregation* stage, either directly, or indirectly through presence of some datasets at the server. In FedNova [10], clients run different numbers of local iterations, because they are heterogeneous in their computational power or because their datasets have different sizes. A normalization of the local gradient of each client ensures that the global model is not biased because of clients that run more iterations. The work [11] uses the inverse Euclidean distance between local and global models as aggregation weights at the server to produce the global model out of local ones. The work [12] assumes a dataset at the server that is partially shared with clients to aid in the training process so as to minimize a weighted sum of local-global model divergence and data communication cost.

Relevant but different from the approaches above that learn a single global model, the works on *personalization* aim to learn multiple personalized models [13]. One approach is to train a DNN whose parameters consist of a common part, i.e. the first layers of a DNN (also called global or shared parameters) for all clients, and a client-specific part, i.e. the last very few layers of the model (also called personalized parameters). In FedPer [14], each client updates the global and personalized parameters through joint gradient descent steps and sends back to the server only the locally updated global parameters. Then, the server updates the global parameters through averaging. In [15], the trained model again includes a set of common weights for all clients, and a set of personalized weights that are specific to each client. An exact and unbiased Stochastic Gradient Descent (SGD) step is performed over the full set of weights in a distributed manner, i.e. the updates of personalized weights are performed by the clients, and those of the common ones by the server. Finally, Adaptive Personalized Federated Learning (APFL) [16] learns a mix of local and global parameters for the personalized model of each client so as to both personalize as well as generalize on unseen data.

Our approach belongs in the class of works on client model drift reduction, and it differs from existing works since it uses layer-wise regularization to capture global/local model dissimilarities on a layer basis.

III. Model and notation

A. Federated Learning Model

We consider a supervised-learning scenario in a Federated Learning (FL) setting with a central server and N client nodes. Each client k has a local dataset $\mathcal{D}_k$, $k = 1, \ldots, N$ with D_k data points, denoted as $\mathcal{D}_k = \{(\mathbf{x}_{k,j}, y_{k,j})\}$ for $j = 1, \ldots, D_k$. Let the data points of each dataset $\mathcal{D}_k$ be drawn from probability distribution $P_k(\mathbf{x}, y)$.

Without loss of generality, we assume that a global DNN model is trained for a classification problem with M classes. We denote by $\mathbf{W}^g$ the vector of parameters of the global model. Model learning emerges from the minimization of a training loss function over all client datasets:

$$\mathcal{L}(\mathbf{W}^g) = \frac{1}{\sum_{i=1}^{N} D_i} \sum_{i=1}^{N} \sum_{j=1}^{D_i} \ell(\mathbf{x}_{i,j}, y_{i,j}; \mathbf{W}^g)\,. \tag{1}$$

This global loss function can be equivalently written as:

$$\mathcal{L}(\mathbf{W}^g) = \sum_{i=1}^{N} \beta_i \ell_i(\mathbf{W}^g)\,, \tag{2}$$

where scalar $\beta_i = D_i / (\sum_{j=1}^{N} D_j)$ is the data proportionality of client i, and $\ell_i(\mathbf{W}^g)$ is the client-specific loss over dataset $\mathcal{D}_i$. The form of each data item-individual loss $\ell(y_{i,j}, \mathbf{x}_{i,j}; W)$ depends on whether the task is a regression or a classification one. In multi-class classification, an input feature vector $\mathbf{x}$ is assigned a class label $y \in \{1, \ldots, M\}$, and the subsets of classes across different clients can be mutually exclusive or partially overlap. We consider the cross-entropy loss function,

$$\ell(y_{i,j}, \mathbf{x}_{i,j}; \mathbf{W}^g) = -\sum_{m=1}^{M} \mathbf{1}(y_{i,j} = m) \log \Pr(\hat{y}_{i,j} = m \,|\mathbf{x}_{i,j}; \mathbf{W}^g) \tag{3}$$

where $\mathbf{1}(A)$ is the indicator function of event A, $\hat{y}_{i,j}$ is the output prediction of model $\mathbf{W}^g$ for input $\mathbf{x}_{i,j}$, and the class probability is modeled by the Softmax function, i.e.

$$\Pr(\hat{y}_{i,j} = m \,|\, \mathbf{x}_{i,j}; \mathbf{W}^g) = \frac{e^{a_m}}{\sum_{k=1}^{M} e^{a_k}} \tag{4}$$

where $\mathbf{a} = (a_1, \ldots, a_M)$ is the output of the last layer of the DNN, before Softmax takes effect.

B. DNN model and weight similarity

The DNN model consists of L layers and an output layer. Let n_ℓ denote the number of neurons of layer ℓ, for $\ell = 1, \ldots, L$. The last $(L+1)$-th layer consists of M units, one for each class, i.e. $n_{L+1} = M$, and their values make up vector $\mathbf{a}$ defined above. Let $\mathbf{W}^g = (\mathbf{W}^g_1, \ldots, \mathbf{W}^g_L)$ denote the ensemble of DNN weights of the global model for all layers, where $\mathbf{W}^g_\ell$ is the $(n_\ell \times n_{\ell+1})$ matrix of weights of the global model between layers ℓ and $(\ell+1)$.

Let vector $\mathbf{w}^g_{\ell,j}$ denote the j-th row of $\mathbf{W}^g_\ell$, for $j = 1, \ldots, n_\ell$. This is the vector of dimension $n_{\ell+1}$ of the weights between neuron j of layer ℓ and all neurons of layer $\ell+1$. This vector is written in terms of its components as $\mathbf{w}^g_{\ell,j} = (w^g_{\ell,j}(1), w^g_{\ell,j}(2), \ldots, w^g_{\ell,j}(n_{\ell+1}))$. We refer to $\mathbf{w}^g_{\ell,j}$ as the *layer statistics* of neuron j of layer ℓ for the global model.

Similarly, let $\mathbf{W}^k = (\mathbf{W}^k_1, \ldots, \mathbf{W}^k_L)$ denote the ensemble of DNN weights of the local model of client k, for $k = 1, \ldots, N$, where $\mathbf{W}^k_\ell$ is the $(n_\ell \times n_{\ell+1})$ matrix of weights of the local model of client k between layers ℓ and $\ell+1$. Let $\mathbf{w}^k_{\ell,j}$ denote the j-th row of this matrix, for $j = 1, \ldots, n_\ell$. This is written in terms of its components as $\mathbf{w}^k_{\ell,j} = (w^k_{\ell,j}(1), w^k_{\ell,j}(2), \ldots, w^k_{\ell,j}(n_{\ell+1}))$.

The similarity between the local model of client k and the global model for the layer statistics of neuron j of layer ℓ is the cosine similarity of vectors $\mathbf{w}^g_{\ell,j}$ and $\mathbf{w}^k_{\ell,j}$,

$$s^k_{\ell,j} = \cos(\mathbf{w}^k_{\ell,j},\, \mathbf{w}^g_{\ell,j}) = \frac{\langle \mathbf{w}^g_{\ell,j}\,,\, \mathbf{w}^k_{\ell,j} \rangle}{\|\mathbf{w}^g_{\ell,j}\|\, \|\mathbf{w}^k_{\ell,j}\|} \tag{5}$$

where

$$\|\mathbf{w}^g_{\ell,j}\| = \sqrt{\sum_{i=1}^{n_{\ell+1}} {w^g_{\ell,j}}^2(i)} \text{ and } \|\mathbf{w}^k_{\ell,j}\| = \sqrt{\sum_{i=1}^{n_{\ell+1}} {w^k_{\ell,j}}^2(i)} \tag{6}$$

are the ℓ_2-norms of vectors $\mathbf{w}^g_{\ell,j}$ and $\mathbf{w}^k_{\ell,j}$ respectively, and

$$\langle \mathbf{w}^g_{\ell,j}, \mathbf{w}^k_{\ell,j} \rangle = \sum_{i=1}^{n_{\ell+1}} w^g_{\ell,j}(i) w^k_{\ell,j}(i)$$

is their inner product. The *dissimilarity* between the local model of client k and the global model, for the layer statistics of neuron j of layer ℓ is

$$\lambda^k_{\ell,j} = 1 - s^k_{\ell,} \tag{7}$$

where $\lambda^k_{\ell,j} \in [0, 2]$. The corresponding vector of dissimilarities for all neurons of layer ℓ is $\boldsymbol{\lambda}^k_\ell = (\lambda^k_{\ell,1}, \lambda^k_{\ell,2}, \ldots, \lambda^k_{\ell,n_\ell})$.

The squared Euclidean distance between the local model of client k and the global model for the layer statistics of neuron j of layer ℓ is

$$d^k_{\ell,j} = \|\mathbf{w}^k_{\ell,j} - \mathbf{w}^g_{\ell,j}\|^2 = \sum_{i=1}^{n_{\ell+1}} \left(w^k_{\ell,j}(i) - w^g_{\ell,j}(i)\right)^2. \tag{8}$$

The vector of squared Euclidean distances of layer statistics for all neuron of layer ℓ is $\mathbf{d}^k_\ell = (d^k_{\ell,1}, d^k_{\ell,2}, \ldots, d^k_{\ell,n_\ell})$.

IV. The Proposed FedLap algorithm

A. FedLap algorithm

The key idea in our Federated Learning with Layer-Adaptive Proximation (FedLap) algorithm (Algorithm 1) is a new regularization term that generalizes the one in FedProx by capturing *divergence* between the global model (GM) and local model (LM) weights *at the DNN layer level*. Divergence is captured through a dissimilarity metric and a distance metric between the GM and LMs for each DNN layer. Thus, the weights of different layers are treated differently in the regularization term of the local loss function of a client.

In FedProx, the loss function of client k is

$$h^{\text{FedProx}}_k(\mathbf{W}^k) = \ell_k(\mathbf{W}^k) + \frac{\mu}{2}\|\mathbf{W}^k - \mathbf{W}^g\|^2, \tag{9}$$

where $\ell_k(\cdot)$ is the training loss for client k, and $\mathbf{W}^g$ is the current GM sent to clients. Each client k runs local iterations to find LM weights $\mathbf{W}^k$ to minimize $h^{\text{FedProx}}_k(\mathbf{W}^k)$.

In FedLap, we *replace* the fixed proximal term μ with the layer-aware vector of dissimilarities $\boldsymbol{\lambda}^k_\ell$, and we replace the weight regularization term $\|\mathbf{W}^k - \mathbf{W}^g\|^2$ with the layer-aware squared Euclidean distance vector $\mathbf{d}^k_\ell$. For each client k with LM weights $\mathbf{W}^k$ and GM weights $\mathbf{W}^g$ received from the server, the local iterations try to minimize the local loss,

$$h^{\text{FedLap}}_k(\mathbf{W}^k) = \ell_k(\mathbf{W}^k) + \frac{1}{2}\sum_{\ell=1}^{L} \langle \boldsymbol{\lambda}^k_\ell\,,\, \mathbf{d}^k_\ell \rangle, \tag{10}$$

where $\langle \mathbf{x}\,,\, \mathbf{y} \rangle$ is the inner product of vectors $\mathbf{x}$ and $\mathbf{y}$. When a client is selected by the server for a local training iteration, it receives the current GM and trains an LM on its own training data. Each client computes dissimilarity and distance of its current LM from the current GM for each layer. Then, it forms its loss function by incorporating the new regularization term. After some iterations on its local training data towards minimizing its loss function, the client derives a new LM and sends it to the server. The server aggregates all client LMs to issue the next version of GM to distribute to clients. The process above continues until convergence.

B. Discussion

1) Dissimilarity and distance metrics: Dissimilarity $\boldsymbol{\lambda}^k_\ell$ and distance $\mathbf{d}^k_\ell$ between the LM of client k and the GM vary during the server-client weight exchange process, as they depend on instantaneous GMs and LMs. By bringing LM-GM dissimilarity into the regularization term, we focus on those layers for which the LM weights are not similar to

Algorithm 1 FedLap algorithm. T is the number of global epochs; C is the fraction of participating clients in each training round; N is the total number of clients; E is the number of local epochs at each training round; B is the local minibatch size; L is the number of layers of local and global models; η is the learning rate.

function FEDLAPSERVER
initialize $\mathbf{W}^g$
for each global round $t = 1, 2, \cdots, T$
$m \leftarrow \max\{C \cdot N, 1\}$
(number of clients participating in this round)
$S_t \leftarrow$ random set of m clients
Send $\mathbf{W}^g$ to clients
  for each client $k \in S_t$ in parallel **do**
  $\mathbf{W}^k_{t+1} \leftarrow$ CLIENTUPDATE(k, $\mathbf{W}^g$)
  (server receives local weights from client k)
  end for
$\mathbf{W}^g_{t+1} \leftarrow \frac{1}{|S_t|} \sum_{k \in S_t} \mathbf{W}^k_{t+1}$
(Model aggregation at the server, combining LMs)
$\mathbf{W}^g \leftarrow \mathbf{W}^g_{t+1}$
end for
end function

function CLIENTUPDATE($k, \mathbf{w}$) (runs on client k)
Receive $\mathbf{W}^g$; $\mathbf{w} \leftarrow \mathbf{W}^g$
for each local epoch i from 1 to E
  Compute vectors $\mathbf{d}^k_\ell$, $\boldsymbol{\lambda}^k_\ell$, for layers $\ell = 1, \ldots, L$
  for each batch $b \in \mathcal{D}_k$ of size B
  $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla h^{\text{FedLap}}_k(\mathbf{w})$
  (where $h^{\text{FedLap}}_k(\cdot)$ is given by (10))
  end for
end for
$\mathbf{W}^k_{t+1} \leftarrow \mathbf{w}$
Send $\mathbf{W}^k_{t+1}$ back to server
end function

those of the GM, or that they were increased most during the update, so as to bring them closer to the GM weights, without changing the weights of layers that were not updated. On the other hand, distance metrics adjust the magnitude of the regularization term. The inner product $\langle \boldsymbol{\lambda}^k_\ell , \mathbf{d}^k_\ell \rangle$ helps to carefully regularize the model.

2) Evolving algorithm behavior during training: The impact of regularization changes in the course of training. As training progresses, FedLap gradually becomes similar to FedAvg. This is because, as the GM converges, the layer-wise similarity between the LMs and the GM increases and approaches 1. Thus, $\boldsymbol{\lambda}^k_\ell$ approach zero; hence the effect of regularization is weakened, and ultimately no regularization is applied, as in FedAvg. Furthermore, since the GM converges and fits all client data, LMs make ever smaller weight updates since they already fit their data to the GM. Thus, $\mathbf{d}^k_\ell$ approach zero as well. Therefore, at the first iteration rounds, the fast convergence of FedLap is exploited, while at the final stages of training we revert to FedAvg to leverage the positive effects of weight averaging to reach an optimum.

While FedProx applies regularization from the first step of local training, in FedLap the adaptation of the regularization term allows LMs to explore different directions from the GM before regularization takes effect. Thus, LMs make larger steps at the first rounds and then slow down so that the weights are close to those of the GM.This is because a client k updates its LM with global weights and when they start local training, the $\boldsymbol{\lambda}^k_\ell$'s are zero. As clients' local updates progress, regularization takes effect due to increasing dissimilarity and distance.

V. DATA EXPERIMENTS

A. Experiment setup and hyperparameters

For the MNIST, FashionMNIST, and CIFAR10 datasets and 100 clients, we keep 10,000 images for testing and the rest for training and validation. For MNIST and FashionMNIST, we use a two-layer neural network with ReLu activation for the hidden layer, and a Softmax activation for the output layer. For CIFAR10, we use the architecture from the Tensorflow Tutorial which consists of 3 convolutional layers, followed by 2 fully connected layers, all with ReLu activation functions and Softmax at the output layer.

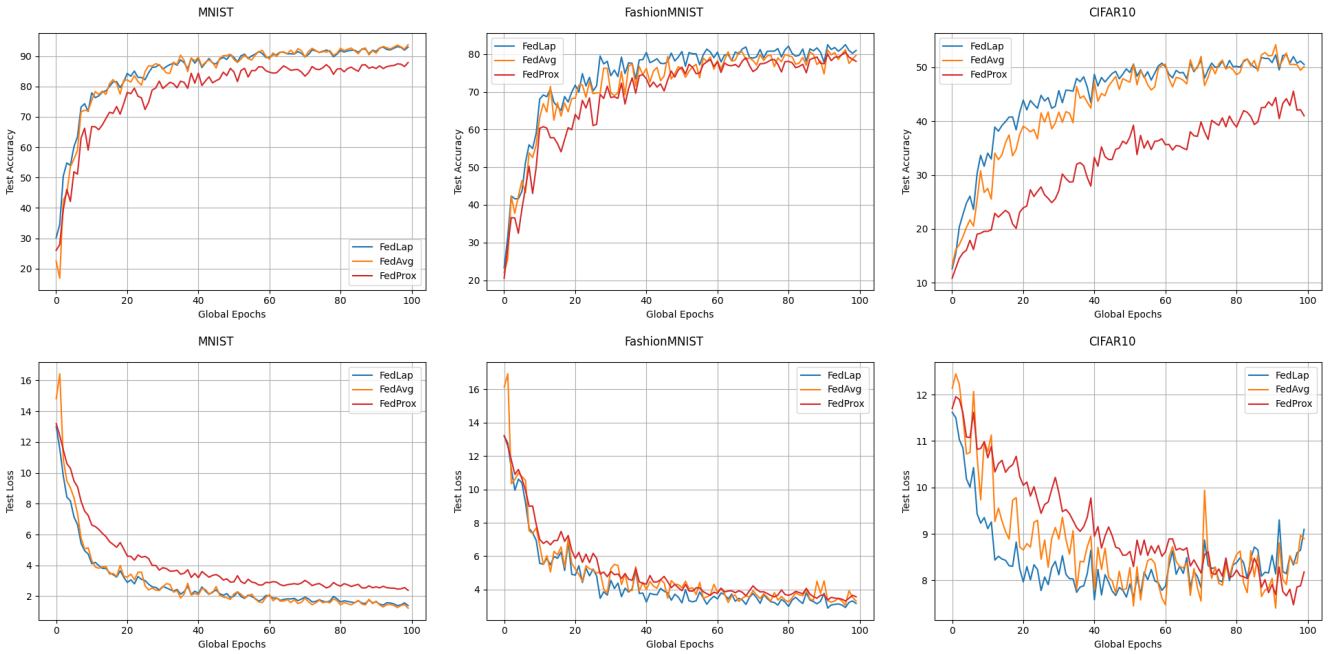
To model non-IID data, we sort data by the class label, we divide it into 200 chunks of size 300, 300, and 200 respectively for MNIST, FashionMNIST, and CIFAR10, and we give 2 chunks to each client so that most clients have 2 of the 10 classes. For each round, we pick a random 10% of clients. We use the SGD optimizer as local solver, with a learning rate 0.01 and momentum 0.9. All experiments were conducted with PyTorch version 1.6.0 and NVIDIA CUDA version 10.2. For a fair comparison among FedLap, FedProx and FedAvg, we introduce an importance parameter $q \in [0, 1]$ in the regularization in (10). For $q = 0$ and $q = 1$, FedLap becomes FedAvg and FedProx. We found that the best accuracy is obtained when the batch size B and number of epochs E are 10, while q does not affect convergence, thus we set $q = 0.5$.

B. Comparison of FedLap vs. FedProx and FedAvg

For the MNIST dataset (Fig. 1a), FedLap starts from a higher accuracy than FedAvg and FedProx, and ultimately it performs similarly to FedAvg and better than FedProx. The performance difference is not large compared to FedAvg, but it can be observed that FedLap follows a smoother learning curve than FedAvg which shows more spikes. For the FashionMNIST dataset (Fig. 1b), the performance gain of FedLap becomes clear, as it converges to higher accuracy than FedProx and FedAvg, while for the CIFAR10 dataset (Fig. 1c), which is harder than MNIST and FashionMNIST, the positive effect of adaptive regularization becomes more visible. FedLap manages to achieve a larger accuracy level much faster, before converging and slowly transforming to FedAvg.

C. Performance comparison for unstable client participation

Next, we study the performance benefit of FedLap when the clients' participation in FL training is unstable, because



(a) MNIST (b) FashionMNIST (c) CIFAR10

Figure 1: Test dataset accuracy for FedLap, FedProx and FedAvg.

either (*i*) clients lack computational power and cannot perform all local updates by the time the server asks them to return their LMs, or (*ii*) they have unstable connection and cannot communicate with the server, or (*iii*) their connection is slow and the server cannot afford to wait for their updates. In FedProx, it was proposed to take into account those LMs and allow unstable clients to send whatever LMs they have learned until the moment when the server asks them. We simulate this behavior by randomly selecting clients (stragglers) from those that participate in the training round. Stragglers perform a random number of local updates in $[1, E]$ and send back to the server the LMs they learned up to that time.

For the MNIST dataset (Fig. 2), FedLap achieves high accuracy much faster than FedAvg and FedProx for unstable client connection. For 50% and 90% of stragglers, FedLap achieves 80% accuracy in less than 20 rounds. Similar trends can be observed in Figs. 3 and Fig. 4 for FashionMNIST and CIFAR10. FedLap achieves 50% accuracy with both 50% and 90% stragglers in around 50 communication rounds.

The *main takeaways* from our experiments are as follows:

- FedLap achieves higher accuracy and reaches this value in fewer communication rounds compared to FedAvg and FedProx, especially in more difficult datasets.
- FedLap significantly outperforms FedAvg and FedProx in terms of accuracy and convergence speed in the presence of unstable client participation.
- FedLap convergence speed and accuracy improves when the number of local computations is increased.
- FedLap approximates FedAvg when the GM converges.
- FedLap performance is not negatively affected by uncertain client participation.

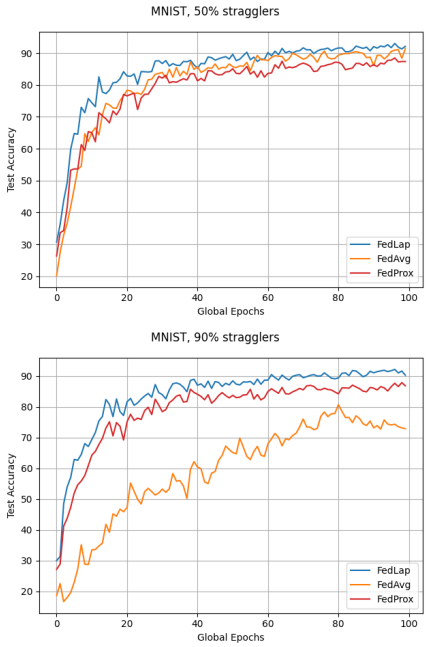


Figure 2: Test dataset accuracy for FedLap, FedProx and FedAvg on MNIST with 50% (top) and 90% (bottom) of clients selected as stragglers.

VI. Conclusion

The main novelty of FedLap is the introduction of a layer-wise regularization term in the local loss function of

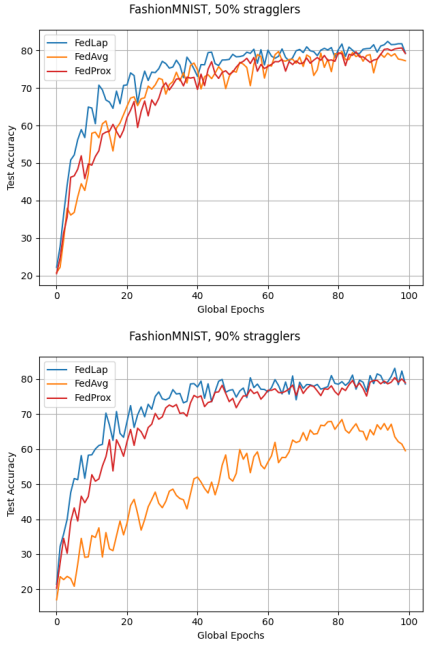


Figure 3: Test dataset accuracy for FedLap, FedProx and FedAvg on FashionMNIST with 50% (top) and 90% (bottom) of clients selected as stragglers.

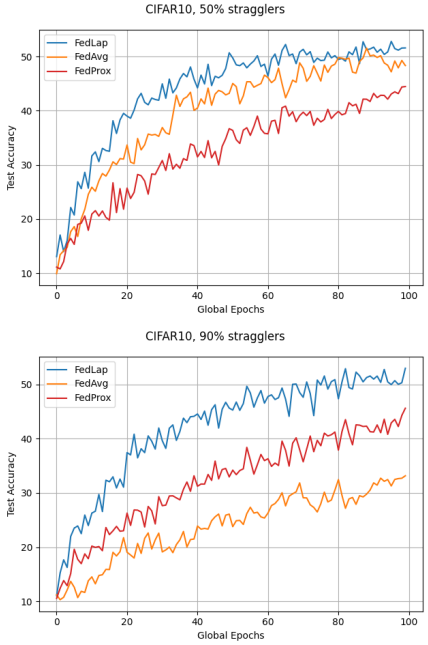


Figure 4: Test dataset accuracy for FedLap, FedProx and FedAvg on CIFAR10 with 50% (top) and 90% (bottom) of clients selected as stragglers.

clients. FedLap achieves much faster convergence and higher test accuracy compared to FedAvg and FedProx. There exist various directions that warrant future investigation. At the experimental front, it makes sense to experiment with other similarity metrics or transformed versions of them and other distance metrics. At the algorithm front, model similarity and distance and layer-wise regularization could be adapted to create a personalized model for each client that gradually integrates the GM in the model, without forgetting previous knowledge if clients do not participate in a training round.

Acknowledgment

This work was supported by the CHIST-ERA grant CHIST-ERA-18-SDCDN-004 (project LeadingEdge, grant number T11EPA4-00056) through the General Secretariat for Research and Innovation (GSRI). It was also supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under Project Number HFRI-FM17-352, Project Title "Wireless Mobile Delay-Tolerant Network Analysis and Experimentation", Project acronym LEMONADE.

References

[1] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, "Federated Optimization in Heterogeneous Networks", ArXiv Abs/1812.06127, 2018.

[2] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-IID data: A survey", *Elsevier Neurocomputing*, vol.465, pp.371-390, Nov. 2021.

[3] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-IID data in Federated Learning", *Elsevier Future Gener. Computing Syst.*, vol. 135, pp.244-258, Oct. 2022.

[4] X. Li, K. Huang, W. Yang, S. Wang and Z. Zhang, "On the Convergence of FedAvg on Non-IID Data", *in Proc. Int. Conf. on Learning Representation (ICLR),* 2020.

[5] Q. Li, B. He and D. Song, "Model-Contrastive Federated Learning," *in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* 2021.

[6] P. Tian, W. Liao, W. Yu and E. Blasch, "WSCC: A Weight Similarity Based Client Clustering Approach for Non-IID Federated Learning," *in IEEE Internet of Things Journal,* vol.9, no.15, pp. 20243-20256, Oct. 2022.

[7] C. Briggs, Z. Fan and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," *in Proc. Int. Joint Conf. on Neural Networks (IJCNN),* 2020.

[8] H. Wu and P. Wang, "Node Selection Toward Faster Convergence for Federated Learning on Non-IID Data," *IEEE Trans. on Network Science and Engineering,* Feb.2022.

[9] H. Wang, Z. Kaplan, D. Niu and B. Li, "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning," *in Proc. IEEE INFOCOM,* 2020.

[10] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization" *in Proc. Advances in Neural Information Processing Systems (NeurIPS)* 2020.

[11] Y. Yeganeh, A. Farshad, N. Navab, and S. Albarqouni, " Inverse Distance Aggregation for Federated Learning with Non-IID Data", *In Domain Adaptation and Representation Transfer, and Distr. Collab. Learning (DART DCL),* 2020, Lecture Notes in Computer Science(), vol 12444. Springer.

[12] Z. Zhao et al., "Federated Learning With Non-IID Data in Wireless Networks," *IEEE Trans. on Wireless Communications,* vol. 21, no. 3, pp. 1927-1942, March 2022.

[13] Y. Mansour, M. Mohri, J. Ro, A. T. Suresh, "Three Approaches for Personalization with Applications to Federated Learning", ArXiv Abs/2002.10619, 2020.

[14] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, S. Choudhary, "Federated Learning with Personalization Layers, ArXiv Abs/1912.00818, 2019.

[15] S. Nikoloutsopoulos, I. Koutsopoulos and M.K. Titsias, "Personalized Federated Learning with Exact Stochastic Gradient Descent", ArXiV Abs/2202.09848, 2022.

[16] Y. Deng, M. M. Kamani, M. Mahdavi, "Adaptive Personalized Federated Learning", ArXiv Abs/2003.13461, 2020.