Speeding up Heterogeneous Federated Learning with Sequentially Trained Superclients

Speeding up Heterogeneous Federated Learning with Sequentially Trained Superclients

Riccardo Zaccone*, Andrea Rizzardi*, Debora Caldarola, Marco Ciccone, Barbara Caputo
Politecnico di Torino, Turin, Italy
*Equal contributors and corresponding authors: {name.surname}@studenti.polito.it

***Abstract*—Federated Learning (FL) allows training machine learning models in privacy-constrained scenarios by enabling the cooperation of edge devices without requiring local data sharing. This approach raises several challenges due to the different statistical distribution of the local datasets and the clients' computational heterogeneity. In particular, the presence of highly non-i.i.d. data severely impairs both the performance of the trained neural network and its convergence rate, increasing the number of communication rounds requested to reach a performance comparable to that of the centralized scenario. As a solution, we propose FedSeq, a novel framework leveraging the sequential training of subgroups of heterogeneous clients, *i.e.* *superclients*, to emulate the centralized paradigm in a privacy-compliant way. Given a fixed budget of communication rounds, we show that FedSeq outperforms or match several state-of-the-art federated algorithms in terms of final performance and speed of convergence. Finally, our method can be easily integrated with other approaches available in the literature. Empirical results show that combining existing algorithms with FedSeq further improves its final performance and convergence speed. We test our method on CIFAR-10 and CIFAR-100 and prove its effectiveness in both i.i.d. and non-i.i.d. scenarios.**

I. Introduction

In 2017, McMahan *et al.* [23] introduced Federated Learning (FL) to train machine learning models in a distributed fashion while respecting privacy constraints on the edge devices. In FL, the clients are involved in an iterative two-step process over several communication rounds: (i) independent training on edge devices on local datasets, and (ii) aggregation of the updated models into a shared global one on the server-side. This approach is usually effective in homogeneous scenarios, but fails to reach comparable performance against non-i.i.d. data. In particular, it has been shown that the non-iidness of local datasets leads to unstable and slow convergence [21], suboptimal performance [19], [38] or even model divergence [23]. Several lines of research emerged to address the statistical challenges of FL: client drift mitigation aims at regularizing the local objective in order to make it closer to the global one [1], [14], [21]; multi-task approaches treat each distribution as a *task* and focus on fitting separate but related models simultaneously [30]; FCL integrates Continual Learning (CL) in the FL setting by allowing each client to have a privately accessible sequence of tasks [32]; data sharing approaches use small amounts of public or synthesized i.i.d. data to help build a more balanced data distribution [38].

In this work, we tackle the problems of i) *non identical class distribution*, meaning that for a given pair instance-label $(x, y) \sim P_k(x, y)$, $P_k(y)$ varies across edge devices k while $P(y|x)$ is identical, and ii) *small local dataset cardinality*. Inspired by the differences with the standard centralized training procedure, which bounds any FL algorithm, we introduce Federated Learning via Sequential Superclients Training (FedSeq), a novel algorithm that leverages sequential training among subgroups of clients to tackle statistical heterogeneity. We simulate the presence of homogeneous and larger datasets without violating the privacy constraints: clients having different distributions are grouped, forming a *superclient* based on a dissimilarity metric. Then, within each superclient, the global model is trained sequentially, and the updates are finally combined on the server-side. Intuitively, this scheme resembles the training on devices having larger and less unbalanced datasets, falling into a favorable scenario for FL. To the best of our knowledge, this is the first federated algorithm to employ such a sequential training on clients grouped by their dissimilarity. To summarize, our main contributions are:

- We introduce FedSeq, a new federated algorithm which learns from groups of sequentially-trained clients, namely *superclients*.
- We introduce two lightweight procedures to estimate the probability distribution of a client and analyze how they affect the ability of grouping algorithms to produce better

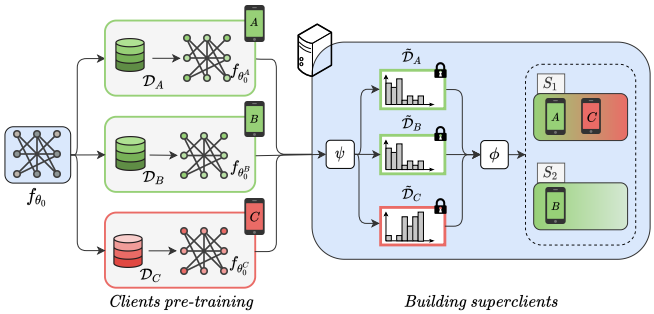


Fig. 1: Building superclients with FedSeq. i) The initial model f_{θ_0} is sent to all K clients, where is trained to fit the local distributions $\mathcal{D}_k$. ii) On the server-side, according to an approximator ψ, the trained models $f_{\theta_0^k}$ are used to obtain an estimate of the clients' distributions $\tilde{\mathcal{D}}_k$. ϕ builds the superclients, grouping together clients having different distributions (A and C), while dividing similar ones (A and B).

superclients. We evaluate two strategies, comparing them with the naïve random assignment, showing the impact of groups quality on the algorithm convergence.
- We show that our method outperforms the state-of-the-art in terms of convergence performance and speed in both i.i.d. and non-i.i.d. scenarios

II. Related works

Recent years have seen a growing interest in Federated Learning [13], [20], [37]. In realistic federated scenarios, a major challenge is posed by the non-i.i.d. and highly unbalanced distribution of the clients' data, also known as *statistical heterogeneity* [10], [22].

FedAvg [23] defines the standard optimization method in FL, where a global model is obtained as a weighted average of local models trained on clients' private data. However, in heterogeneous settings, the local optimization objectives drift from each other, leading to different local models which are hard to be aggregated [14]. Several works demonstrate how the convergence rates of FedAvg get worse with the increase of clients heterogeneity [11], [15], [21], [22], [35]. SCAFFOLD [14] tries to mitigate this issue by introducing control variates, while FedProx [21] adds a proximal term to the local loss function. FedDyn [1] dynamically updates the local objective to ensure the asymptotic alignment of the global and devices solutions. Server-side optimizers [11], [27] have been also introduced for coping with FedAvg lack of adaptivity. In [24], it is showed how fair model aggregation is beneficial when clients observe non-i.i.d. data. While referring mainly to [23] for the aggregation scheme, our work revises the standard framework to account for statistical heterogeneity.

As the learned local model under-represents the deducible patterns from the missing classes, [38] shows how sharing a small set of public data among the clients leads to notable improvements. A similar approach is followed by [18], where the public data enables knowledge distillation. Similarly to [18], we keep the public data on the server-side, with the different purpose of using them to estimate the clients' data distribution in a privacy-compliant way. Unlike [18], [38], such data is never used at training time.

Another line of work tackles the problem from a multitask perspective [6], where each client is treated as a different task [7], [30]. In [4], [5], [29], [36], [16] clients with similar tasks are clustered together and a specialized model is assigned to each cluster. In [5], tasks are identified using a domain classifier learned via knowledge distillation and then addressed by the means of a graph, while in our method, following the same approach of [4], [29], [36], the locally trained model are used to approximate the clients' data distribution. Unlike those works, FedSeq exploits clustering methods to group together clients having distant distributions, in order to obtain an underlying homogeneous dataset within each group, *i.e. superclient*. Our approach also relates to the "*anti-clustering*" literature [25], [26], where the goal is to build similar groups from dissimilar elements [33]. From here on we will refer to such techniques as "*grouping algorithms*".

Finally, FedSeq also relates to peer-to-peer (P2P) methods for FL [12], [28] by sharing models between clients belonging to the same superclient. Unlike such works, we keep the central server as a proxy between clients and prioritize FL's statistical challenges rather than communication costs.

III. Method

A. Problem formulation

In the FL setup, the goal is to learn a global model $f_\theta : \mathcal{X} \to \mathcal{Y}$, parametrized by θ, on data distributed among K clients without sharing local information. Each device $k \in [K]$ has access to n_k samples from a local dataset $\mathcal{D}_k = \{x_i, y_i\}_{i=1}^{n_k}$ where $x \in \mathcal{X}$ is the input and $y \in \mathcal{Y}$ its corresponding label.

FedAvg [23] follows an iterative approach based on T communication rounds with the goal of solving

$$\arg\min_{\theta \in \mathbb{R}^d} \sum_{k \in \tilde{C}} \frac{n_k}{n} L_k(\theta),\ d \in \mathbb{N}^+ \tag{1}$$

where $L_k(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}[\ell_k(f_\theta;(x,y))]$ is the local empirical risk, ℓ_k the cross-entropy loss, and $n = \sum_k n_k$ the total amount of training data. At each round $t \in [T]$, the server sends θ_t to a fraction of $\tilde{C}$ randomly selected clients. Each client $k \in \tilde{C}$ computes its update θ_{t+1}^k using $\mathcal{D}_k$ by minimizing the local objective and sends it back to the server. The updated weights are then aggregated by the server into a new global model $f_{\theta_{t+1}}$ as:

$$\theta_{t+1} \leftarrow \sum_{k \in \tilde{C}} \frac{n_k}{n} \theta_{t+1}^k \tag{2}$$

However, in realistic scenarios, there is no guarantee that local datasets from different clients are drawn independently from the same underlying distribution. , *i.e.* given two clients i and j, $\mathcal{P}(\mathcal{D}_i) \neq \mathcal{P}(\mathcal{D}_j)$. More in general, $f_{\theta^k} \neq f_\theta\ \forall k$ clients [4]. In this work, we mitigate the issue of statistical heterogeneity in classification tasks by introducing FedSeq, an algorithm for FL that leverages sequential training among a sub-sample of clients $\tilde{C}_S$, grouped together according to their data distribution. Specifically, clients observing different data are grouped into a *superclient* S obtaining an approximation of the underlying uniform distribution over all N_c classes, *i.e.* $\bigcup_{k\in\tilde{C}_S} \mathcal{D}_k \sim \mathcal{U}_{[N_c]}$. Intuitively, thanks to the sequential training inside superclients, local models can accumulate knowledge on the majority of the classes even if single clients heavily heterogeneous.

B. Building superclients

Our goal is to build a superclient S from users having different local distributions without breaking the privacy constraints, *i.e.* without directly accessing the clients' data (Figure 1). We propose different grouping criteria G_S as an ensemble of i) a *client distribution approximator* $\psi_{(.)}$ providing statistics regarding the local distribution in a privacy-preserving way, ii) a *metric* τ for evaluating the distance between the estimated data distributions and iii) a *grouping method* $\phi_{(.)}$ to assemble dissimilar clients, *i.e.* $G_S := \{\psi_{(.)}; \tau; \phi_{(.)}\}$.

*1) **Client distribution approximator**:* We split the model f_θ into a deep feature extractor $h_{\theta_{\text{feat}}} : \mathcal{X} \to \mathcal{Z}$ and a classifier $g_{\theta_{\text{clf}}} : \mathcal{Z} \to \mathcal{Y}$, where $\theta = (\theta_{\text{feat}}, \theta_{\text{clf}})$ is the entire set of model parameters. The classification output is given by $g \circ h : \mathcal{X} \to \mathcal{Y}$, where we drop the subscripts to ease the notation.

FedSeq exploits a *pre-training stage* to estimate the clients' data distribution, during which each client k produces a model $f_{\theta_0^k}$ by training on its local dataset for e epochs starting from the same random initialization θ_0. We propose two strategies based on i) the parameters of the local classifier $\theta^k_{0,\text{clf}}$ or ii) its predictions on a server-side public dataset $\mathcal{D}_{pub}$ $\{f^k(z) = g^k(h^k(z)),\ z \in \mathcal{D}_{pub}\}$, respectively ψ_{clf} and ψ_{conf}.

For ψ_{clf}, we hypothesize the weights of the classifier can be representative of the local distribution [2] of each client and directly feed them to the grouping method $\phi_{(.)}$.

For ψ_{conf}, we test each $f_{\theta_0^k}$ on a public *"exemplar set"* $\mathcal{D}_{pub} = \bigcup_{c=1}^{N_c} \mathcal{D}_c$, where $\mathcal{D}_c$ contains J samples for class $c \in [N_C]$. Then, we average the predictions by class as $p_{k,c} = \frac{1}{J} \sum_{x \in \mathcal{D}_c} f_{\theta_0^k}(x)$, and define the k-th client's *confidence vector* as:

$$p_k := \text{softmax}(\{p_{k,1}, ..., p_{k,N_C}\}) \in [0,1]^{N_C} \tag{3}$$

In the following sections, we indicate as $\tilde{\mathcal{D}}_k$ the estimate provided by $\psi_{(.)}$ for the k-th device's data distribution.

*2) **Grouping metrics**:* Starting from client k's data approximation $\tilde{D}_k$, we build similar superclients from users having different distributions, *i.e.* we aim at minimizing the inter-superclients distance while maximizing the intra-superclient one. To do so, given $\tilde{\mathcal{D}}_i$ and $\tilde{\mathcal{D}}_j$, we need a metric $\tau(\tilde{\mathcal{D}}_i, \tilde{\mathcal{D}}_j) : \mathbb{R}^{N_C \times N_C} \to \mathbb{R}$ to measure the distance between the two distribution estimates. We compare the weights of the clients' classifier using the cosine and Euclidean distance, but other popular metrics can be used [34]. When $\tilde{\mathcal{D}}_k$ as the form of an actual probability distribution given by the confidence vector, we also adopt two *disomogeneity* measures, the Gini index [8] and the Kullback-Leibler (KL) divergence [17].

*3) **Grouping method**:* We first define $\mathcal{D}_S = \bigcup_{k \in \tilde{C}_S} \mathcal{D}_k$ as the union of the data from the clients $\tilde{C}_S$ belonging to a superclient S. Our aim is to find the maximum amount of superclients N_S satisfying the following constraints: i) minimum number of samples $|\mathcal{D}_S|_{min}$ and ii) maximum number of clients $K_{S,max}$. We introduce three strategies to find an approximation of the maximization problem, given the chosen $\psi_{(.)}$ and τ. The first, ϕ_{rand}, is a naïve yet practical approach where clients are randomly assigned to superclients until the defined stopping criterion is met. The second one, ϕ_{kmeans}, is based on the K-means algorithm [31]: first, K-means is applied to obtain N_S homogeneous clusters; then, each superclient is formed by iteratively extracting one client at a time from each cluster, until the number of samples $|\mathcal{D}_S|$ in each superclient S is at least $|\mathcal{D}_S|_{min}$ and the number of clients $K_S \leq K_{S,max}$ (detailed algorithm in Appendix A). Lastly, ϕ_{greedy} follows a greedy methodology to produce superclients. Initially, one random client k_i is assigned to the current superclient S, $i \in [K]$. Then, the second client k_j is chosen so as the distance between k_i and k_j is maximized, *i.e.* $\max_{j \in [K]} \tau(\tilde{\mathcal{D}}_{k_i}, \tilde{\mathcal{D}}_{k_j})$. The process is repeated until the established maximum number of clients $K_{S,max}$ and the minimum number of samples $|\mathcal{D}_S|_{min}$ are reached by iteratively maximizing $\tau(\tilde{\mathcal{D}}_j, \frac{1}{|S|} \sum_{i \in |S|} \tilde{\mathcal{D}}_i)$, with $|S|$ being the cardinality of S until that point (see Appendix A).

Algorithm 1: FEDSEQ and FEDSEQINTER

Require: f_{θ_0}, G_S, $K_{S,max}$, $|\mathcal{D}_S|_{min}$. Epochs e, E_k, E_S. T rounds. Clients K. Fraction C of superclients selected at each round.

$S \leftarrow$ CREATESUPERCLIENTS(f_{θ_0}, G_S, e, $K_{S,max}$, $|\mathcal{D}_S|_{min}$, K)
$N_S \leftarrow |S|$
$\Theta \leftarrow [\theta_0, \dots, \theta_0]_{1 \times CN_S}$, $w \leftarrow [0, \dots, 0]_{1 \times CN_S}$
for $t = 0$ to T **do**
  $S^t \leftarrow$ Subsample fraction C of N_S superclients
  for $S_i \in S^t$ **in parallel do**
    Shuffle clients in S_i
    $\theta_t^{S_i,0} \leftarrow \theta_t$ $\theta_t^{S_i,0} \leftarrow \Theta[i]$
    for $e_S = 1$ to E_S **do**
      $\theta_{t+1}^{S_i} \leftarrow$ SEQUENTIALTRAINING($\theta_t^{S_i,0}$, E_k)
    end for
    $\Theta[i] \leftarrow \theta_{t+1}^{S_i}$, $w_i \leftarrow w_i + |\mathcal{D}_{S_i}|$
  end for
  $\theta_{t+1} \leftarrow$ FEDAVG($\{\theta_{t+1}^{S_i},\ \forall S_i \in S^t\}$)
  if $t \mod N_S = 0$ **then**
    $\theta_{t+1} \leftarrow \sum_i \frac{w_i}{w} \Theta[i]$, $w = \sum_i w_i$
    $\Theta \leftarrow [\theta_{t+1}, \dots \theta_{t+1}]$, $w \leftarrow [0, \dots, 0]$
  end if
end for

C. Sequential training

*1) **FedSeq**:* Within each superclient S_i, with $i \in [N_S]$, training is performed in a sequential way, meaning that S_i is considered as a sequence of clients $k_{i,1}, \dots, k_{i,|S_i|}$. The server sends the global model f_{θ_t} to the first device $k_{i,1}$, which trains it for E_k epochs on $\mathcal{D}_{k_{i,1}}$. The obtained parameters $\theta_{t+1}^{k_{i,1}}$ are sent to the next client $k_{i,2}$. Such training procedure continues until the last client $k_{|S_i|}$ updates the received model, possibly repeating for E_S times following a ring communication strategy. Then, the last client sends its update to the server, where all the superclients updates are averaged according to Eq. 1. The details of FedSeq are summarized in Algorithm 1.

*2) **FedSeqInter**:* Sequentiality can be also exploited at a superclient level. At each round t, every selected superclient S_i receives the model $\theta_t^{S_j}$ from another previously involved superclient S_j, initially θ_0. Every N_S rounds the models are averaged, weighted by the number of examples on which each model was trained on. The insight behind this approach is that it might be useful to merge models only after they have been trained on a larger portion of the dataset. Statistically, after N_S rounds, each model is likely to have been trained on the entire dataset, thus getting closer to a centralized scenario. This strategy requires far fewer aggregation and synchronization steps with the server: the possibility to go out of sync accounts for variance in clients' delays, allowing faster superclients not to be slacken by slower ones.

Dataset	Algorithm	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.5$	Centr.
CIFAR-10	FedAvg	71.41	76.82	77.98	85.72
	FedProx	71.41	76.84	77.98	
	SCAFFOLD	79.02	76.47	78.25	
	FedDyn	**83.26**	81.74	82.41	
	FedSeq	82.21	82.20	82.23	
	FedSeqInter	82.65	**82.79**	**83.32**	
	FedSeq + FedProx	82.14	82.16	82.49	
	FedSeq + FedDyn	82.90	**83.55**	**83.95**	
	FedSeqInter + FedProx	82.95	82.95	83.52	
	FedSeqInter + FedDyn	**83.11**	83.06	83.33	
CIFAR-100	FedAvg	42.66	48.02	48.89	55.13
	FedProx	42.66	48.20	48.88	
	SCAFFOLD	42.04	51.04	51.20	
	FedDyn	-	**54.41**	**54.99**	
	FedSeq	46.00	49.55	49.82	
	FedSeqInter	**50.27**	51.60	51.94	
	FedSeq + FedProx	46.02	49.71	49.62	
	FedSeq + FedDyn	50.45	50.23	50.80	
	FedSeqInter + FedProx	**51.13**	**51.54**	52.33	
	FedSeqInter + FedDyn	51.06	51.04	**52.68**	

TABLE I: Comparison with SOTA FL algorithms and centralized scenario.

IV. EXPERIMENTS

We evaluate FedSeq on image classification tasks from CIFAR-10 and CIFAR-100, widely used as benchmarks in FL. In order to set up a heterogeneous scenario, the local class distribution is sampled from a Dirichlet distribution with $\alpha \in \{0, 0.2, 0.5\}$ [10]. Implementation details can be found in Appendix E. We evaluate our results in terms of global accuracy on the test set (Tables I and III) and convergence rates (Table II). All reported results are averaged over the last 100 rounds.

A. Comparison with state-of-the-art FL algorithms

We compare our method with the state-of-the-art (SOTA) algorithms FedAvg [23], FedProx [21], SCAFFOLD [14] and FedDyn [1]. The analysis is presented both in terms of convergence performance (Figure 2, Table I) and speed (Table II). Taking into account both the convergence performance and rates, the best configuration chosen for FedSeq is based on the greedy grouping algorithm with KL-divergence applied on confidence vectors, *i.e.* $G_S = \{\psi_{\text{conf}}, \phi_{\text{greedy}}, \tau_{KL}\}$. In addition, all results are compared with FedSeqInter, which adds the inter-superclient sequential training to this configuration, and is shown to outperform any configuration of FedSeq.

*1) **Results at convergence**:* Table I shows how FedSeq reaches consistently better results than other methods not only when addressing extreme data heterogeneity, but also when faced with less severe conditions. This behavior reflects equally on both datasets. In particular, FedProx seems unable to address extreme scenarios, maintaining performances comparable to FedAvg. SCAFFOLD proves itself effective in addressing the most unbalance case ($\alpha = 0$), with $+7\%$ at convergence on CIFAR-10, but fails at improving the results achieved by FedAvg both in more moderate scenarios and on CIFAR-100. We found FedDyn to be the best current state-of-the-art algorithm, reaching the target accuracies for all configurations except CIFAR-100 with $\alpha = 0$. FedSeq successfully address the challenge of extremely unbalanced clients on both datasets, outperforming FedAvg, FedProx and SCAFFOLD both in terms of final performance and convergence speed, being on par with FedDyn in the average case (Figure 2). FedSeqInter - although initially slower - reaches the highest accuracy value, close to that of the centralized scenario Acc_{centr}: in the most challenging setting, the achieved value corresponds to $96.4\% \cdot Acc_{centr}$ on CIFAR-10 and $91.2\% \cdot Acc_{centr}$ on CIFAR-100. That tells us that aggregating every N_S rounds not only leads to less frequent synchronization between clients and server with a consequent speed up of the training process, but also improves the accuracy reached.

*2) **Integrating FedSeq with state-of-the-art**:* Since FedSeq keeps the same logic of FedAvg both in the local training and the server-side aggregation, it can be easily integrated with other approaches modifying those parts of the algorithm. In particular, we evaluate the performance of FedProx [21] and FedDyn [1] on top of FedSeq, since changes to the local objective are straightforward to transfer in our sequential training framework. FedProx adds a proximal term μ to the local objective to improve stability and regularize the distance between the local and global models. We can repurpose FedProx to be used in our sequential framework by adding a proximal term to retain the information learned by the previous client rather than the global model, with potential benefits in the most challenging settings. Similarly FedDyn can be integrated in FedSeq by adding both linear and quadratic penalty terms to the loss function, using the model trained by the previous client in place of the server's last model (see Appendix F for the details). Results in Table I show that integrating FedSeq with FedProx makes the algorithm converge slightly faster only in the most unbalanced scenario, while performances are on par in the remaining the cases.

*3) **Convergence speed analysis**:* In Table II, we report the time (indicated as number of rounds) needed by our best configurations and SOTAs to reach respectively the 70%, 80% and 90% of the centralized accuracy, also indicating the speedup relative to FedAvg. Considering the most challenging situations, on CIFAR-10, FedSeq based on the KL divergence on confidence vectors is 7 times faster than FedAvg and successfully reaches the 90% of the centralized accuracy in less than a third of rounds budget; on CIFAR-100, FedSeqInter is the only algorithm able to reach the 90% of the centralized accuracy, in less than half of the available rounds. Unfortunately, our experiments running FedDyn on CIFAR-100 with $\alpha = 0$ failed to converge, probably due to the extreme imbalance combined with the difficulty of the task.

B. Ablation study

In this Section, we provide information on the ablation studies performed on FedSeq. Specifically, the details regarding the pre-training phase and the construction of superclients are shown, together with the analysis of the different configurations available for FedSeq which led to the choice presented in Section IV-A.

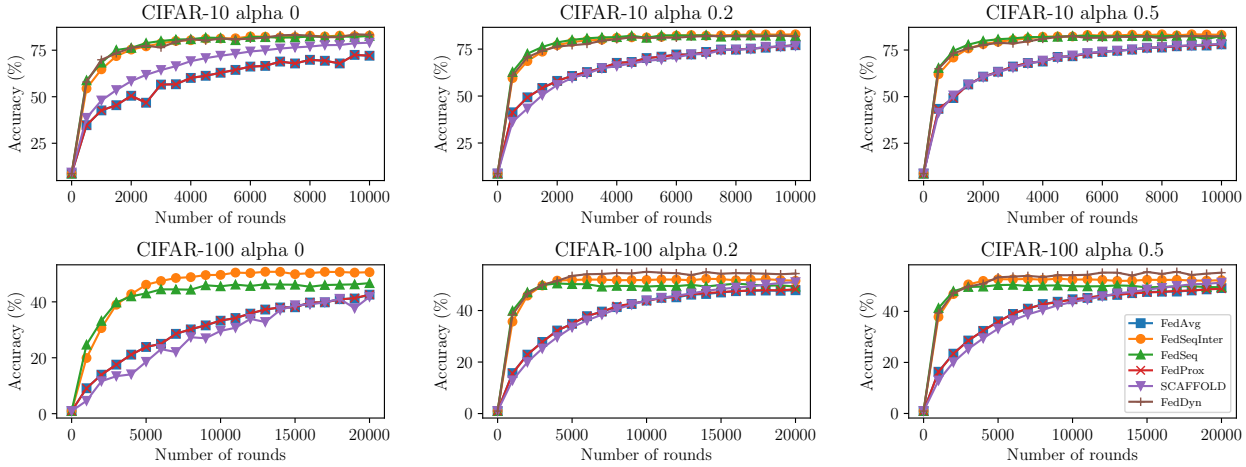


Fig. 2: Comparison between the results of SOTAs and the best configurations of FedSeq and FedSeqInter by varying α and dataset. FedSeqInter performs on par with FedDyn and both outperform the other approaches. Best viewed in color.

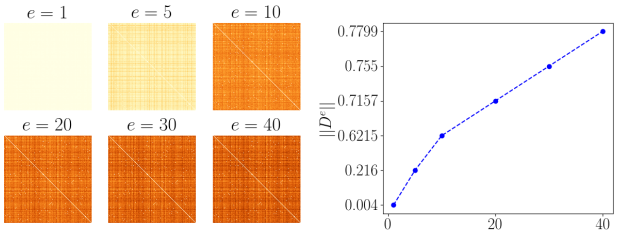
Dataset	Method	$\alpha = 0$			$\alpha = 0.2$			$\alpha = 0.5$		
		70%	80%	90%	70%	80%	90%	70%	80%	90%
CIFAR-10	FedAvg	4036 (1x)	7649 (1x)	- (–)	2384 (1x)	4507 (1x)	- (–)	1945 (1x)	3749 (1x)	8791 (1x)
	FedProx	4036 (1x)	7649 (1x)	- (–)	2384 (1x)	4507 (1x)	- (–)	1946 (1x)	3753 (1x)	8786 (1x)
	SCAFFOLD	2229 (1,81x)	3914 (1,95x)	8043 (–)	2554 (0,93x)	4771 (0,94x)	- (–)	1934 (1,01x)	3761 (1x)	8453 (1,04x)
	FedDyn	**563 (7,17x)**	**954 (8,02x)**	**2131 (–)**	**450 (5,3x)**	**797 (5,65x)**	**2059 (–)**	**374 (5,2x)**	**634 (5,91x)**	**1648 (5,33x)**
	FedSeq[1]	594 (6,79x)	991 (7,72x)	2047 (–)	407 (5,86x)	746 (6,04x)	1682 (–)	325 (5,98x)	619 (6,06x)	1358 (6,47x)
	FedSeq[2]	873 (4,62x)	1502 (5,09x)	3677 (–)	387 (6,16x)	720 (6,26x)	1543 (–)	323 (6,02x)	620 (6,05x)	1409 (6,24x)
	FedSeq[1] + FedProx	594 (6,79x)	991 (7,72x)	2046 (–)	407 (5,86x)	746 (6,04x)	1682 (–)	325 (5,98x)	619 (6,06x)	1358 (6,47x)
	FedSeq[1] + FedDyn	**345 (11,7x)**	**581 (13,17x)**	**1341 (–)**	**253 (9,42x)**	**447 (10,08x)**	**1113 (–)**	**232 (8,38x)**	**403 (9,3x)**	**933 (9,42x)**
	FedSeqInter[1]	762 (5,3x)	1305 (5,86x)	2492 (–)	**538 (4,43x)**	1004 (4,49x)	2099 (–)	**433 (4,49x)**	**814 (4,61x)**	1805 (4,87x)
	FedSeqInter[1] + FedProx	735 (5,49x)	1264 (6,05x)	2388 (–)	544 (4,38x)	1000 (4,51x)	2084 (–)	436 (4,46x)	825 (4,54x)	**1747 (5,03x)**
	FedSeqInter[1] + FedDyn	**733 (5,51x)**	**1262 (6,06x)**	**2344 (–)**	**533 (4,47x)**	**959 (4,7x)**	**2061 (–)**	**425 (4,58x)**	**796 (4,71x)**	1750 (5,02x)
CIFAR-100	FedAvg	14412 (1x)	- (–)	- (–)	6409 (1x)	10253 (1x)	- (–)	5879 (1x)	9331 (1x)	- (–)
	FedProx	14412 (1x)	- (–)	- (–)	6363 (1,01x)	10277 (1x)	- (–)	5918 (0,99x)	9250 (1,01x)	- (–)
	SCAFFOLD	14483 (1x)	- (–)	- (–)	7088 (0,9x)	10191 (1,01x)	17200 (–)	6951 (0,85x)	10373 (0,9x)	16744 (–)
	FedDyn	- (–)	- (–)	**- (–)**	**1031 (6,22x)**	**1603 (6,4x)**	**2634 (–)**	**868 (6,77x)**	**1433 (6,51x)**	**3018 (–)**
	FedSeq[1]	3009 (4,79x)	5780 (–)	- (–)	901 (7,11x)	1421 (7,22x)	**3436 (–)**	854 (6,88x)	1264 (7,38x)	2812 (–)
	FedSeq[2]	3968 (3,63x)	9378 (–)	- (–)	922 (6,95x)	1396 (7,34x)	3924 (–)	843 (6,97x)	1266 (7,37x)	2713 (–)
	FedSeq[1] + FedProx	2946 (4,89x)	6005 (–)	- (–)	898 (7,14x)	1397 (7,34x)	3033 (–)	843 (6,97x)	1298 (7,19x)	2833 (–)
	FedSeq[1] + FedDyn	**1914 (7,53x)**	**3293 (–)**	**7511 (–)**	**556 (11,53x)**	**987 (10,39x)**	**2014 (–)**	**541 (10,87x)**	**957 (9,75x)**	**1912 (–)**
	FedSeqInter[1]	3028 (4,76x)	4333 (–)	8494 (–)	1177 (5,45x)	1734 (5,91x)	3004 (–)	1034 (5,69x)	**1524 (6,12x)**	2675 (–)
	FedSeqInter[1] + FedProx	3027 (4,76x)	4310 (–)	**7149 (–)**	**1163 (5,51x)**	**1721 (5,96x)**	**2915 (–)**	1033 (5,69x)	1525 (6,12x)	2616 (–)
	FedSeqInter[1] + FedDyn	**2964 (4,86x)**	**4183 (–)**	7539 (–)	1180 (5,43x)	1757 (5,84x)	3018 (–)	**1031 (5,7x)**	1538 (6,07x)	**2547 (–)**

TABLE II: Convergence rates for the best configurations of FedSeq (1: $\{\psi_{\text{conf}}, \phi_{\text{greedy}}, \tau_{KL}\}$, 2: $\{\psi_{\text{clfAll}}, \phi_{\text{greedy}}, \tau_{cosine}\}$) and SOTAs. We report the round in which the 70%, 80% and 90% of centralized accuracy is reached ("–" if the target accuracy was not reached), together with the speedup relative to FedAvg ("–" if FedAvg did not reach the target accuracy).

*1) **Clients pre-training**:* All the grouping criteria introduced in Section III-B rely on the clients' data approximation $\tilde{\mathcal{D}}_k$, produced by the *approximator* ψ. Regardless of the choice of ψ, the first step required for building superclients is a pre-training phase, local to every client. The randomly initialized model f_{θ_0} is trained by each device for e epochs and is then exploited for estimating the data distribution without breaking the privacy constraints. Intuitively, e should be large enough for the model to fit the local training set and at the same time as small as possible so as not to cause a computational burden on the clients. Hence we expect models trained on similar distributions to be more alike than those that have seen different ones. We tested $e \in \{1, 5, 10, 20, 30, 40\}$. For each of those values, we obtain the similarity matrix $D^e := \{D^e_{ij} = \frac{\theta^i_e \cdot \theta^j_e}{||\theta^i_e|| \, ||\theta^j_e||}\}$, representing the cosine distance between $f_{\theta^i_e}$ and $f_{\theta^j_e}$, where θ^i_e and θ^j_e are respectively the parameters of client i and j models trained for e local epochs, $\forall (i, j) \in (K \times K)$. Figure 4a shows those matrices as heatmaps for CIFAR-100 (see Appendix B for CIFAR-10). In Figure 4b, the trend of $||D^e||$ for each value of e is reported: we can notice how 5 epochs are sufficient for the models to be significantly different and after 10 epochs of pre-training the change rate of the models is reduced. Therefore, looking for the trade-off between the informative value of the trained models and the performance overhead, we choose $e = 10$ as

Method	ψ	ϕ	τ	$\alpha = 0$	$\alpha = 0.2$	$\alpha = 0.5$
			CIFAR-10			
	-	random	-	81.90	82.09	82.12
	clf	K-means	Euclidean	**82.30**	81.78	82.48
	conf	K-means	Euclidean	82.04	81.99	82.37
FedSeq	conf	greedy	KL	82.21	**82.20**	82.22
	conf	greedy	Cosine	82.09	81.85	82.71
	clf	greedy	Cosine	79.95	82.06	**82.83**
FedSeqInter	conf	greedy	KL	**82.65**	**82.79**	**83.32**
			CIFAR-100			
	-	random	-	**46.39**	48.62	49.44
	clf	K-means	Euclidean	44.91	48.74	49.60
	conf	K-means	Euclidean	43.55	49.43	49.79
FedSeq	conf	greedy	KL	45.97	**49.56**	**49.82**
	conf	greedy	Cosine	45.79	48.98	49.61
	clf	greedy	Cosine	45.22	48.92	49.62
FedSeqInter	conf	greedy	KL	**50.27**	**51.60**	**51.94**

TABLE III: FedSeq baselines: comparison of grouping criteria by varying ϕ, ψ and τ. Results in terms of accuracy (%).



(a) (b)

Fig. 3: Effect of pre-training $K = 500$ local models for $e \in \{1, 5, 10, 20, 30, 40\}$ epochs on CIFAR-100. (a) Heatmaps of the similarity matrix D^e. (b) Trend of $||D^e||$. After $e = 10$ the slope of the curve decreases.

default value for the clients pre-training.

*2) **Estimating clients' data distribution**:* We can extract an estimate of the distribution of local datasets from the clients' pre-trained models via an approximator ψ (see Sec. III-B1. We compare ψ_{clf} and ψ_{conf}, based respectively on the pre-trained classifier weights and on the confidence vectors (Eq. 3). As for the *classifier* approximator, we test three different scenarios: we use all three fully connected layers of the network, the last two or only the last. To mitigate the *curse of dimensionality* [3], we apply PCA [9] on the parameters, keeping 90% of the explained variance. Our key findings are that the percentage of preserved components: i) decreases with the complexity of the dataset, *i.e.* less components are needed for CIFAR-10, and ii) increases directly proportional to e, except for $\alpha = 0$ (more details in Appendix C). We deduce that 10 local epochs are already sufficient to capture the polarization of the dataset in its extreme imbalance. As for ψ_{conf}, we retain 10 images per class from the test set on the server-side ($\mathcal{D}_{pub}$) for testing the pre-trained models and computing the *confidence vectors* as described in Section III-B1. Once $\mathcal{D}_{pub}$ has served its purpose, it is not used again.

*3) **Comparison of grouping criteria**:* Here we provide the experimental results of the different combinations of grouping criteria G_S. As for the implementation of the grouping method ϕ_{kmeans}, a reasonable value of K is the number of classes of the dataset. In order to evaluate how *homogenous* the superclients' overall data distribution is, we use the following measures:

- *balance ratio* $:= \frac{\min_{c \in [N_C]} N_c}{\max_{c \in [N_C]} N_c}$, where N_c is the number of samples for the class c
- *covered classes* $:= \frac{1}{N_C} \sum_{c=1}^{N_C} \mathbb{1}_{P(y=c)>0}$.

It should be noted that the percentage of classes covered is a less discriminatory measure, as the class is accounted for as present even if only one of its samples is in the superclient, while a low deviation from the mean of the samples per class is necessary to have a higher balance ratio, making the latter more reliable. In Appendix C, Table I shows the results varying by ψ, ϕ and τ. The first consideration is that the random assignment strategy (ϕ_{rand}) has surprisingly good indices, especially if compared with more clever algorithms. The reason lies in statistical considerations on the setting: when $\alpha = 0$, there are multiple clients (*i.e.* 50 clients in CIFAR-10 and 5 in CIFAR-100) having samples belonging to the same class; therefore, a random choice in unlikely to group only those clients with the same data distribution. As α grows, each client has a more homogeneous distribution, so every clustering criterion leads to a similar result. ϕ_{kmeans} is the best performing algorithm when $\alpha = 0$, with zero variance on the number of clients in the same set. ϕ_{greedy} shows better performances in most cases, hence it is our algorithm of choice. Figures in Appendix D show examples of superclients built with different ϕ. As for the approximators, it is possible to see that, fixed the choice of ϕ_{greedy}, the use of ψ_{clf} mostly leads to higher balance ratio, especially when τ_{cosine} is adopted, while Table III shows that ψ_{conf} brings towards higher accuracy. As for the metrics, the speedup with τ_{KL} is more prominent (Table II). So our approximator of choice is ψ_{conf} with τ_{KL}.

V. CONCLUSION

In this work we address statistical heterogeneity in FL introducing FedSeq, the first approach exploiting sequential training of clients grouped by data dissimilarity (*superclients*). We evaluate different stategies for grouping clients, based on privacy-preserving approximations of their local distributions, and show that FedSeq is robust to suboptimal solutions. We extend sequential training to superclients to reduce the impact of slow devices (FedSeqInter) and find that the convergence performances improve. Our comparative analysis with the state-of-art shows that FedSeq largely outperforms FedAvg, FedProx and SCAFFOLD in terms of convergence accuracy and speed on both extreme and less severe non-i.i.d. scenarios, while performing on par with FedDyn on average. Finally, empirical results show that combining existing algorithms with FedSeq further improves its final performance and convergence speed.

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. *International Conference on Learning Representations*, 2021.

[2] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6430–6439, 2019.

[3] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[4] Christopher Briggs, Zhong Fan, and Peter Andras. Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

[5] Debora Caldarola, Massimiliano Mancini, Fabio Galasso, Marco Ciccone, Emanuele Rodolà, and Barbara Caputo. Cluster-driven graph federated learning over multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2749–2758, 2021.

[6] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[7] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.

[8] Frank A Farris. The gini index and measures of inequality. *The American Mathematical Monthly*, 117(10):851–864, 2010.

[9] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[10] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *NeurIPS Workshop*, 2019.

[11] 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.

[12] Chenghao Hu, Jingyan Jiang, and Zhi Wang. Decentralized federated learning: A segmented gossip approach. *arXiv preprint arXiv:1908.07782*, 2019.

[13] 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. *arXiv preprint arXiv:1912.04977*, 2019.

[14] 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.

[15] Ahmed Khaled, Konstantin Mishchenko, and Peter Richtárik. First analysis of local gd on heterogeneous data. *arXiv preprint arXiv:1909.04715*, 2019.

[16] Kavya Kopparapu and Eric Lin. Fedfmc: Sequential efficient federated learning on non-iid data. *arXiv preprint arXiv:2006.10937*, 2020.

[17] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[18] Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581*, 2019.

[19] Qinbin Li, Yiqun Diao, Quan Chen, and Bingsheng He. Federated learning on non-iid data silos: An experimental study. *CoRR*, abs/2102.02079, 2021.

[20] 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.

[21] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, 2:429–450, 2020.

[22] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[23] 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.

[24] Umberto Michieli and Mete Ozay. Are all users treated fairly in federated learning systems? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2318–2322, 2021.

[25] Sai Lokesh Reddy Y. Srijayanthi Subramanian Sakthivel Ravichandran Mohammed Fayaz A., Neethimani S. M. Comparative analysis of anti-clusters formed using various distance metrics and k-medoids algorithm. *International Journal of Advanced Science and Technology*, 29(06):7705–7717, Jun. 2020.

[26] Martin Papenberg and Gunnar W Klau. Using anticlustering to partition data sets into equivalent parts. *Psychological Methods*, 26(2):161, 2021.

[27] Sashank Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečnỳ, Sanjiv Kumar, and H Brendan McMahan. Adaptive federated optimization. *International Conference on Learning Representations (ICLR)*, 2021.

[28] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[29] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE transactions on neural networks and learning systems*, 2020.

[30] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. *arXiv preprint arXiv:1705.10467*, 2017.

[31] Douglas Steinley. K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34, 2006.

[32] Anastasiia Usmanova, François Portet, Philippe Lalanda, and German Vega. A distillation-based approach integrating continual learning and federated learning for pervasive services. *arXiv preprint arXiv:2109.04197*, 2021.

[33] Ventzeslav Valev. Set partition principles revisited. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition*, pages 875–881, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[34] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[35] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[36] Ming Xie, Guodong Long, Tao Shen, Tianyi Zhou, Xianzhi Wang, Jing Jiang, and Chengqi Zhang. Multi-center federated learning. *arXiv preprint arXiv:2108.08647*, 2021.

[37] Jie Xu, Benjamin S Glicksberg, Chang Su, Peter Walker, Jiang Bian, and Fei Wang. Federated learning for healthcare informatics. *Journal of Healthcare Informatics Research*, 5(1):1–19, 2021.

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

APPENDIX

A. GROUPING ALGORITHMS

Here, we provide details on the grouping algorithms described in Section III.

Algorithm 2: K-means grouping method ϕ_{kmeans}

Require: K clients, $\{\tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_K\}$ clients' approximated distributions, $|\mathcal{D}_S|_{min}$ minimum number of samples per superclient, $K_{S,max}$ maximum number of clients per superclient, grouping metric τ, n_k number of images on kth device

$N =$ number of classes
$C_1, \ldots, C_N$ = K-MEANS($\{\tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_K\}, \tau, N$) {K-means algorithm with $K = N$ returns N homogeneous clusters}
$z \leftarrow 0,\ S = [\,],\ j = 0$ {with S being the set of superclients and z its index}
while $|\bigcup_{i=1}^{N} C_i| > 0$ **do**
  $S_z \leftarrow [\,],\ N_z \leftarrow 0$
  while $N_z < |\mathcal{D}_S|_{min}$ **and** $|S_z| < K_{S,max}$ **do**
    $k \leftarrow$ RANDOM(C_j)
    S_z.ADD(k), C_j.REMOVE(k)
    $j \leftarrow ((j+1) \mod N)$
    $N_z \leftarrow N_z + n_k$
  end while
  S.ADD(S_z)
  $z \leftarrow z + 1$
end while
return S

Algorithm 3: Greedy grouping method ϕ_{greedy}

Require: K clients, $\{\tilde{\mathcal{D}}_1, \ldots, \tilde{\mathcal{D}}_K\}$ clients' approximated distributions, $|\mathcal{D}_S|_{min}$ minimum number of samples per superclient, $K_{S,max}$ maximum number of clients per superclient, grouping metric τ, n_k number of images on kth device

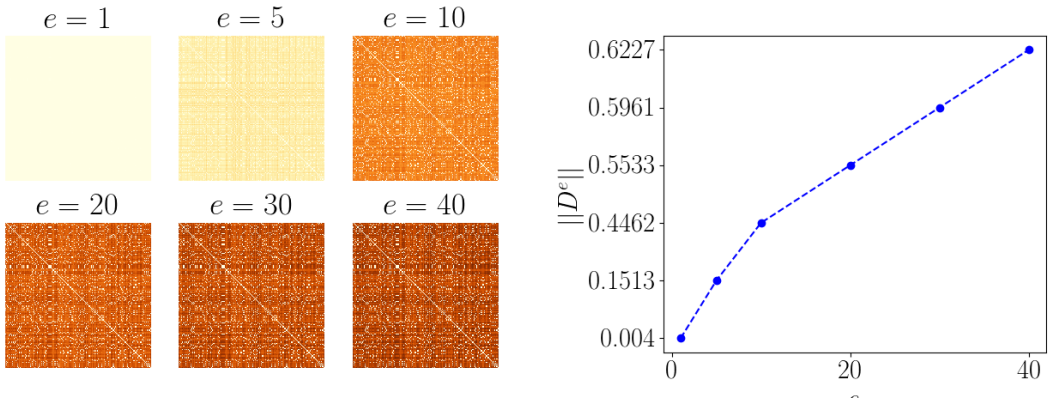
$z \leftarrow 0,\ S = [\,],\ \tilde{K} \leftarrow [k_1, \ldots k_K]$
while $|\tilde{K}| > 0$ **do**
  $S_z \leftarrow [\,],\ N_z \leftarrow 0$
  $k_i \leftarrow$ RANDOM($\tilde{K}$)
  S_z.ADD(k_i), $\tilde{K}$.REMOVE(k_i)
  $\tilde{\mathcal{D}}_{S_z} \leftarrow \tilde{\mathcal{D}}_i,\ N_z \leftarrow N_z + n_i$
  while $N_z < |\mathcal{D}_S|_{min}$ **and** $|S_z| < K_{S,max}$ **do**
    $k_j \leftarrow argmax_j(\tau(\tilde{\mathcal{D}}_j, \tilde{\mathcal{D}}_{S_z}))$
    $\tilde{\mathcal{D}}_{S_z} \leftarrow \frac{1}{2}\tilde{\mathcal{D}}_{S_z} + \frac{1}{2}\tilde{\mathcal{D}}_j$
    $N_z \leftarrow N_z + n_j$
    S_z.ADD(k_j), $\tilde{K}$.REMOVE(k_j)
  end while
  S.ADD(S_z)
  $z \leftarrow z + 1$
end while
return S

B. CLIENTS' PRE-TRAINING ON CIFAR-10

Figure 4 shows the effect of pre-training local models varying the number of local epochs e for CIFAR-10. As shown for CIFAR-100 in the main paper, we find a trade-off between the informative value of the trained models and the performance overhead with $e = 10$. The results obtained are consistent across both datasets, showing that the chosen network is able to correctly fit both of them.

C. COMPARISON OF GROUPING CRITERIA

Table IV shows experimental results of the different combinations of grouping criteria G_S. We remind that the goal of our approach is to group clients with different distributions in the same superclient, in order to obtain heterogeneous ones.



(a) (b)

Fig. 4: Effect of pre-training $K = 500$ local models for $e \in \{1, 5, 10, 20, 30, 40\}$ epochs on CIFAR-10. **(a)** Heatmaps of the similarity matrix D^e. **(b)** Trend of $||D^e||$. After $e = 10$ the slope of the curve decreases.

To this end, our evaluation metrics are the *balance ratio* and *covered classes* (see Section IV-B3 of the main paper), as a way to reflect the heterogeneity of superclient's dataset. The approximators *classifierAll*, *classifierLast2* and *classifierLast* refer respectively to extracting the weights of all, the last two or only the last fully connected layer from our network of choice, LeNet-5. In practice, since we apply PCA on the network parameters (Figure 5), extracting all the classifier's weights does not introduce much additional computational burden. Moreover, *classifierAll* achieves the best performance among the three options. Therefore we choose to always extract all the weights. Results are consistent across the dataset and show that the best combinations are $G_S^a = \{\psi_{\text{clf}}, \phi_{\text{greedy}}, \tau_{cosine}\}$ and $G_S^b = \{\psi_{\text{conf}}, \phi_{\text{greedy}}, \tau_{cosine}\}$. Experimental results on the performance of FedSeq given such grouping criteria show that on the average case $G_S^c = \{\psi_{\text{conf}}, \phi_{\text{greedy}}, \tau_{KL}\}$ leads to best results (see Section IV-B3).

Approximator ψ	Method ϕ	Metric τ	CIFAR-10		CIFAR-100	
			Balance Ratio	Covered Classes	Balance Ratio	Covered Classes
classifierAll	Greedy	Cosine distance	**0.334**	0.886	**0.028**	0.667
		Wasserstein distance	0.081	0.759	0.011	0.651
	K-means	Euclidean distance	0.207	0.902	0.009	0.655
classifierLast2	Greedy	Cosine distance	**0.275**	0.871	**0.034**	0.668
		Wasserstein distance	0.090	0.746	0.009	0.652
	K-means	Euclidean distance	0.203	0.900	0.009	0.654
classifierLast	Greedy	Cosine distance	**0.266**	0.880	**0.043**	0.668
		Wasserstein distance	0.085	0.755	0.010	0.650
	K-means	Euclidean distance	0.204	0.902	0.009	0.655
confidence vectors	Greedy	Cosine distance	**0.311**	0.886	**0.014**	0.658
		Wasserstein distance	0.077	0.784	0.009	0.654
		KL divergence	0.271	0.870	0.011	0.656
		Gini index	0.298	0.876	0.012	0.657
	K-means	Euclidean distance	0.173	0.894	0.009	0.656
-	Random	-	0.068	0.835	0.009	0.655

TABLE IV: Comparison between different clustering methods. Each result is the average of the scores obtained for $\alpha \in [0, 0.2, 0.5]$.

D. SUPERCLIENTS ANALYSIS

Figures 6,7,8 show superclients distributions in different settings. Figure 6 represents the distribution of 10 superclients built, from left to right, with ϕ_{greedy}, ϕ_{kmeans} and ϕ_{rand} on CIFAR-10. It is clear that the first two methods are able to build perfectly homogeneous superclients, while ϕ_{rand} struggles in doing so. Figure 7 shows the same configuration on CIFAR-100:

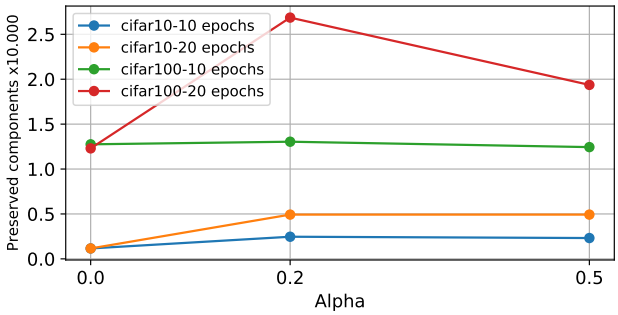


Fig. 5: Ratio of the preserved components after applying PCA with 90% of explained variance.

in this case, the advantage of using ϕ_{greedy} or ϕ_{kmeans} over ϕ_{rand} is not as evident, but the superclient distributions created with the first two clustering methods are still spread more homogeneously over the classes. Figure 8 demonstrates the effect of α (from left to right: 0, 0.2 and 0.5) in the construction of the superclients: the bigger the value of α, the more homogeneous the superclients distributions are, regardless of the clustering method.

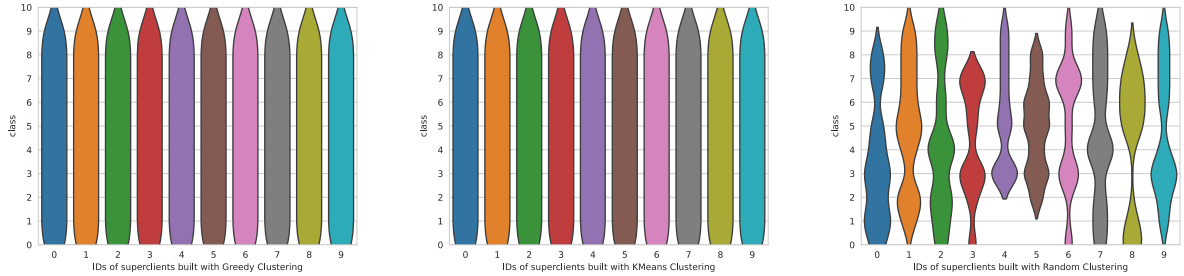


Fig. 6: Example of superclient distributions produced by different grouping algorithms on CIFAR-10 and $\alpha = 0$.

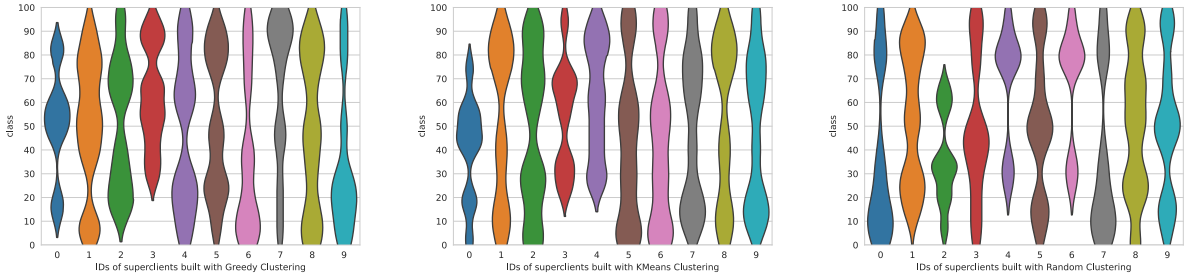


Fig. 7: Example of superclient distributions produced by different grouping algorithms on CIFAR-100 and $\alpha = 0$.

E. IMPLEMENTATION DETAILS

We evaluate FedSeq on image classification tasks on two synthetic datasets widely used as benchmarks in FL, namely CIFAR-10 and CIFAR-100. As for the data partitioning, we follow the protocol described in [10]: the class distribution of every client is sampled from a Dirichlet distribution with varying concentration parameter α. Since our method addresses statistical heterogeneity, in our experiments we use $\alpha \in \{0, 0.2, 0.5\}$ that, combined with the number of clients K among which the dataset is split ($K = 500$), sets up a realistic scenario in which clients have small and very unbalanced datasets.

Accounting for the difficulty of the task, we run the experiments for $T = 10k$ rounds on CIFAR-10 and $20k$ on CIFAR-100. The fraction of clients selected at each round is $C = 0.2$. Following the setup of [11], our model is their proposed version of *LeNet-5*, with a client learning rate of 0.01, weight decay set to $4 \cdot 10^{-4}$, momentum 0 and batch size 64. As for the centralized scenario, we add a momentum of 0.9 and a cosine annealing schedule for the learning rate, training the model for 300 epochs.

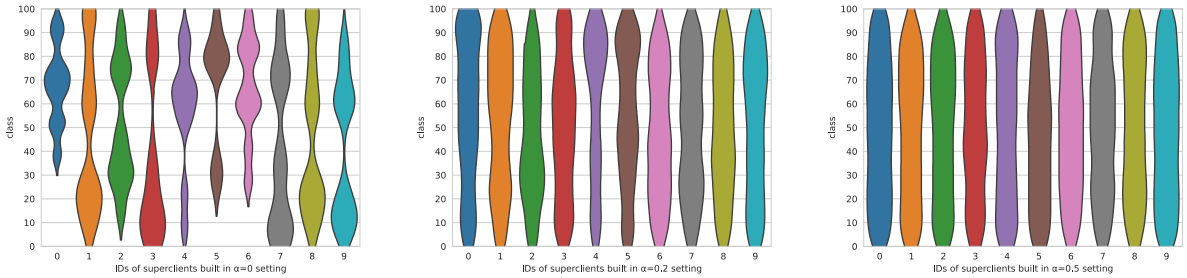


Fig. 8: Example of superclient distributions in different α settings with ϕ_{rand}.

As for the clustering methods, we fix $|\mathcal{D}_S|_{min} = 800$ and $K_{S,max} = 11$. In FedSeq, we fix $E_k = E_S = 1$ and similarly $E = 1$ for FedAvg and the other SOTAs. An analysis on the choice of E_S can be found in Appendix F: we show it is not convenient to perform more than one epoch through a superclient. For FedProx we evaluate $\mu \in [10^{-4}, 10^{-3}, 10^{-2}]$ and choose $\mu = 0.01$, while for FedDyn $\alpha_{dyn} = 0.1$ is chosen from the finetuning interval $[10^{-3}, 10^{-2}, 10^{-1}]$.

Regarding FedDyn, we were unable to obtain the results for CIFAR-100 with $\alpha = 0$: we conjecture that in our setting the amount of local update was not enough to calculate the appropriate h^t server side, and the model diverged. To confirm this intuition we successfully ran the same case using a learning rate of 0.1; the same happens when integrating FedDyn with FedSeq, in which case the models has more updated before returning to the server for the aggregation.

When integrating FedProx in FedSeq, we use $\mu = 0.01$ chosen from $[10^{-4}, 10^{-3}, 10^{-2}]$; when instead we integrate in FedSeqInter, we choose $\mu = 1$ chosen from $[10^{-2}, 10^{-1}, 1, 10]$: the rationale behind having selected higher values is that in a sequential training with loose aggregation it can be beneficial to try to retain more knowledge from the previous client's training. The experimental results in section IV of the main paper confirm this intuition. Similarly when integrating FedDyn in FedSeq, we use $\alpha_{dyn} = 0.1$ chosen from $[10^{-3}, 10^{-2}, 10^{-1}]$, while when integrating in FedSeqInter we choose $\alpha_{dyn} = 1$ from $[10^{-2}, 10^{-1}, 1]$.

F. DETAILS ON THE INTEGRATION OF FEDSEQ WITH STATE-OF-THE-ART

FedProx

As pointed out in section IV-A2, FedProx adds a proximal term μ to the local objective to improve stability and regularize the distance between the local and global models, modifying the local objective function as follows:

$$\theta_k^t = \arg\min_{\theta}(R_k(\theta; \theta^{t-1}) = L_k(\theta) + \frac{\mu}{2}||\theta - \theta^{t-1}||^2) \tag{4}$$

In our setting, incorporating FedProx objecting function into the sequential training means trying to retain the information learned by the previous client rather than the global model, with potential benefits in the most challenging settings. In fact, because when $\alpha = 0$ clients have local dataset with samples belonging only to one class, adding a proximal term could help avoiding the model shift towards the new learned task. In such a case, the objective function becomes:

$$\theta_{S_{k,j}}^t = \arg\min_{\theta}(R_{S_{k,j}}(\theta; \theta_{S_{k,j-1}}^t) = L_{S_{k,j}}(\theta) + \frac{\mu}{2}||\theta - \theta_{S_{k,j-1}}^t||^2) \tag{5}$$

where $\theta_{S_{k,j-1}}^t$ is the model after the training of client $j-1$ belonging to superclient S_k.

FedDyn

In FedDyn the proposed risk objective dynamically modifies local loss functions, so that, if in fact local models converge to a consensus, the consensus point is consistent with stationary point of the global loss [1]. Namely, each device computes:

$$\theta_k^t = \arg\min_{\theta}(R_k(\theta; \theta_k^{t-1}, \theta^{t-1}) = L_k(\theta) + \langle \nabla L_k(\theta_k^{t-1}), \theta \rangle + \frac{\alpha_{dyn}}{2}||\theta - \theta^{t-1}||^2) \tag{6}$$

FedDyn authors point out that for the first order condition for local optima to be satisfied, as $\theta_k^t \to \theta_k^\infty$ and $\nabla L_k(\theta_k^t) \to \nabla L_k(\theta_k^\infty)$, $\theta^t \to \theta_k^\infty$ which implies $\theta_k^\infty \to \theta^\infty$. Then the server side aggregation updates the model such that:

$$\theta^t = \frac{1}{|P_t|}\sum_{k \in P_t} \theta_k^t - \frac{1}{m}\sum_{k \in P_t}(\theta_k^t - \theta^{t-1}) \tag{7}$$

being P_t the subset of client selected at round t. In this way θ^t convergence implies $h^t \to 0$.

When incorporating it in FedSeq, the dynamic regularizer and the first order condition for local optima become:

$$\begin{aligned} R_{S_{k,j}}(\theta;\theta^{t-1}_{S_{k,j}},\theta^{t}_{S_{k,j-1}}) &\triangleq L_{S_{k,j}}(\theta) - \langle \nabla L_{S_k,j}(\theta^{t-1}_{S_{k,j}}),\theta\rangle + \frac{\alpha_{dyn}}{2}||\theta-\theta^{t}_{S_{k,j-1}}||^2) \\ \nabla R_{S_{k,j}}(\theta;\theta^{t-1}_{S_{k,j}},\theta^{t}_{S_{k,j-1}}) &= L_{S_k,j}(\theta^{t}_{S_{k,j}}) - \nabla L_{S_k,j}(\theta^{t-1}_{S_{k,j}}) + \alpha_{dyn}(\theta-\theta^{t}_{S_{k,j-1}}) \end{aligned} \tag{8}$$

Applying the same reasoning of FedDyn, as $\theta^{t}_{S_{k,j}} \to \theta^{\infty}_{S_{k,j}}$ and $\nabla L_{S_k,j}(\theta^{t}_{S_{k,j}}) \to \nabla L_{S_k,j}(\theta^{\infty}_{S_{k,j}})$, this implies $\theta^t \to \theta^{\infty}_{S_{k,j}}$. Analogously the server side aggregation becomes:

$$\theta^t = \frac{1}{|P_t|}\sum_{k\in P_t}\theta^t_{S_k} - \frac{1}{m}\sum_{k\in P_t}(\theta^t_{S_k}-\theta^{t-1}) \tag{9}$$

Because of the sequential training of the models, the term $\theta^t_{S_k}$ can be rewritten as the sum of the gradients computed by each client inside a superclient, leading to the following equation:

$$\begin{aligned} \theta^t_{S_k} &= \theta^{t-1} + \sum_{j}^{|S_k|}\nabla L_{S_{k,j}}(\theta^t_{S_{k,j}}) \Rightarrow \sum_{k\in P_t}(\theta^t_{S_k}-\theta^{t-1}) = \sum_{k\in P_t}\sum_{j}^{|S_k|}\nabla L_{S_{k,j}}(\theta^t_{S_{k,j}}) \\ \theta^t &= \frac{1}{|P_t|}\sum_{k\in P_t}\theta^t_{S_k} - \frac{1}{m}\sum_{k\in P_t}\nabla L_{S_k}(\theta^t_{S_k}) \quad \text{where} \quad \nabla L_{S_k}(\theta^t_{S_k}) \triangleq \sum_{j}^{|S_k|}\nabla L_{S_{k,j}}(\theta^t_{S_{k,j}}) \end{aligned} \tag{10}$$

In this way θ^t convergence implies $\sum_{k\in P_t}\nabla L_{S_k}(\theta^t_{S_k}) \to 0$: indeed the definitions of $h^t \triangleq \sum_k \nabla L_k(\theta^t_k)$ in FedDyn and $h^t \triangleq \sum_k \nabla L_{S_k}(\theta^t_{S_k})$ in FedSeq are analogous.

G. Analysis on the superclients' local epochs E_S

In analogy with the number of client's local epochs E_k, we analyse what happens increasing the superclient's epochs E_S. The intuition behind this study is that, since superclients are built on top of heterogeneous data distributions, more loops on their dataset could produce more robust models, not biased towards a single class. Increasing E_S while decreasing the global round number T does not impact the communication steps, but reduces the number of aggregations and accounts for a more loose synchronization. To compare fairly, Figure 9 shows the results for $E_S \in \{1,2,4\}$, and T decreased by the corresponding factor: to ease the visulization we compare the accuracies along *equivalent rounds*, meaning that the actual round r_{abs} for each line in the graph is scaled by the number of E_S, formally $r_{\text{eq}} = \frac{r_{\text{abs}}}{E_S}$. It is possible to notice that, comparing models with the same amount of training, increasing the number of sequential rounds among superclients' clients does not improve the performance accordingly. As increasing E_S does not decrease communication cost, there is no advantage performing more than one epoch. Differently, using the strategy of FedSeqInter, we obtain a similar effect in that models are more trained before the aggregation step, but:

- **The dataset the model is sequentially trained on is broader:** indeed the aggregation period N_S is choosen such that statistically the models encounters the whole global dataset before the aggregation step;
- **We do not add any computation per round:** even better, aggregation every N_S rounds requires less sync.

In Section IV of the main paper we empirically demonstrate that the latter approach ultimately leads to better convergence performances.

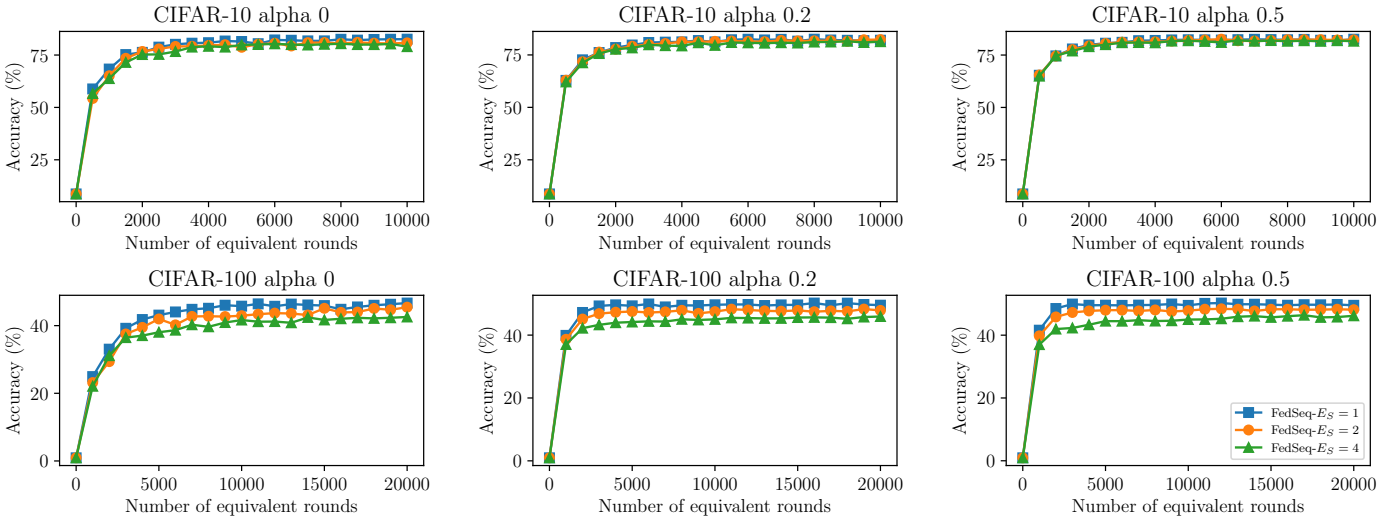


Fig. 9: FedSeq varying $E_S \in \{1, 2, 4\}$. Results show that, on equal effort, increasing the amount of computation through superclients' clients does not improve the performance. Best viewed in color.