IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received 11 March 2024, accepted 24 March 2024, date of publication 11 April 2024, date of current version 26 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3387453

RESEARCH ARTICLE

Accelerating Federated Learning via Sequential Training of Grouped Heterogeneous Clients

ANDREA SILVI, ANDREA RIZZARDI, DEBORA CALDAROLA, BARBARA CAPUTO, AND MARCO CICCONE
Dipartimento di Automatica e Informatica (DAUIN), Politecnico di Torino, 10129 Turin, Italy

Corresponding author: Debora Caldarola (debora.caldarola@polito.it)

This work was supported by the FAIR—Future Artificial Intelligence Research and European Union (EU) Next-GenerationEU [Piano Nazionale di Ripresa e Resilienza (PNRR)—Missione 4 Componente 2, Investimento 1.3—D.D. 1555 11/10/2022] under Grant PE00000013 Computational resources were partly provided by HPC@POLITO.

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 required to reach centralized performance. As a solution, we propose *FedSeq*, a novel framework leveraging the sequential training of subgroups of heterogeneous clients, *i.e.*, *superclients*, to learn more robust models before the server-side averaging step. 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. Our method can be easily integrated with other approaches available in the literature, and empirical results show that combining existing algorithms with FedSeq further improves its final performance and convergence speed. We evaluate our method across multiple FL benchmarks, establishing its effectiveness in both i.i.d. and non-i.i.d. scenarios. Lastly, we highlight that the sequential training introduced here does not introduce additional privacy concerns when compared to the de facto standard, FedAvg.

INDEX TERMS Federated learning, distributed learning, privacy-preserving machine learning, statistical heterogeneity, deep learning.

I. INTRODUCTION

Federated Learning (FL) [1] is a Machine Learning (ML) framework designed to train models in decentralized settings while preserving the privacy of participants – the *clients*. Such a paradigm eliminates the need for clients to disclose their private data with a central authority, thus ensuring compliance with regulations in force. Federated training involves multiple communication rounds, during which a shared global model is trained independently on selected devices. The updated parameters are then aggregated by the server into a new model. While usually effective in homogeneous scenarios [1], [2], [3], where clients have access to similar data, in realistic settings they observe data by breaking the conventional assumption of independence and identical distribution (*i.i.d.*) of classic ML systems. In these scenarios, users collect data from an underlined global distribution based on preferences [4], [5] or geographic position [6], [7], [8], forming a heterogeneous distribution of local datasets. Several works have shown that heterogeneity generally results in slow and unstable convergence of FL algorithms [3], [9], [10], hampering final performance [11], [12] because of local gradients diverging towards different minima [1], [3], and the difficulty to merge specialized models, a phenomenon called *client drift* [13]. This results in a biased and suboptimal global solution compared to the actual minimum [14]. In this work, we focus on heterogeneity caused by i) *label skew*, *i.e.* given an instance-label pair $(x, y) \sim P_k(x, y)$, $P_k(y)$ varies across

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

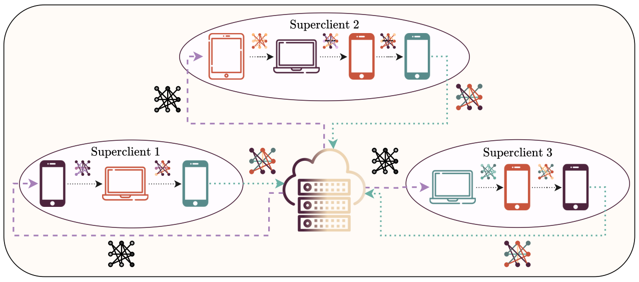


FIGURE 1. **To mitigate statistical heterogeneity in FL, FedSeq forms *superclients* by grouping clients with distinct local data distributions (different colors), creating simulated larger and homogeneous datasets. Sequential training takes place within the selected superclients at each round. The current global model is received by the first client in the chain and sent back by the last one.**

clients k while $P(y|x)$ is identical, ii) *features shift*, *i.e.* $P_k(x)$ varies while $P(y|x)$ is identical, and iii) *different local dataset cardinality*.

To mitigate the effects of the client drift, many methods focus on regularizing the local objective to bring it closer to the global one [10], [13], [15], or on improving the generalization of the learned model [3], [16], [17]. Multitask FL views each local distribution as an individual *task* and aims to train separate but related models concurrently [18], [19], [20], while [21], [22], [23], [24], [25], [26] cluster clients with similar tasks together and assign a specific model each. Data sharing approaches instead leverage fine-tuning of the model over small quantities of public or synthesized i.i.d. data to ensure a more balanced representation of the overall distribution on the server side [11]. However, most of these methods fall short of replicating the performance of centralized scenarios or struggle with extremely skewed heterogeneous distributions [3]. In contrast, **this paper approaches heterogeneity from a different angle, focusing on the training orchestration rather than the training objectives**, to learn more robust models before the server-side averaging step, resulting in reduced noise and achieving centralized performance.

This work presents *Federated Learning via Sequential Superclients Training* (**FedSeq**), a novel approach effectively addressing the issue of statistical heterogeneity in FL. FedSeq employs *sequential* training among clients, carried out in parallel across distinct client groups to harness the distributed setting's parallelism. By allowing the model to access a larger portion of data before the averaging step, the negative effects of data heterogeneity are mitigated, speeding up the training and moving closer to the desired minimum. By grouping clients having diverse local distributions together in a *superclient*, we simulate the existence of a larger, homogeneous dataset while maintaining data privacy, as illustrated in Fig. 1. Clients within the same superclient form a chain and train the received model in a sequential manner. The final updates are sent from the last client to the server and merged there. Intuitively, **this scheme emulates the training dynamics observed on devices with more extensive and evenly distributed datasets**, resulting in a favorable setting for FL.

Communication is known to be the main bottleneck in federated training, *e.g.* due to the clients' unavailability and unreliability [27]. While sequential training provides robustness against data heterogeneity, it can potentially result in slower training progress. This occurs when slower clients end up in the same superclient, leading to increased waiting times on the server side. To overcome this limitation, we present **FedAsyncSeq**, a novel approach that introduces **asynchronous client-server communication by implementing sequential training among superclients**. Rather than merging updates at the end of each training round, FedAsyncSeq allows the model updated by one superclient to be sent directly to another one, enabling faster groups of clients to complete multiple training iterations before merging their updates. At regular intervals of every R rounds, the updates received by the server, potentially stemming from varying numbers of training iterations, are aggregated. This approach not only reduces the number of aggregation and synchronization steps with the server but also allows the model to be trained on a larger number of clients before the averaging step. Consequently, this brings the model closer to a centralized scenario, as ideally, it encounters all superclients before being merged.

A. EXTENSION DETAILS

This work represents an extension of the previous manuscript [28] in several aspects, as summarized in Fig. 2. ① Firstly, we introduce a **novel metric for estimating local distribution similarity without compromising user privacy or needing additional public data**, outperforming the performance of the previously presented approximation techniques. Based on *Task2Vec* [29], it captures both taxonomic and semantic representations of each task, *i.e.*, the local data of each client. In addition, we note that sequential training can suffer from saturation in the later stages of training due to overfitting [30], which can further slow down the overall training process. ② Building upon [30], **FedSeq2Par** is presented as a solution **to increase parallelism as training moves on**. FedSeq2Par dynamically updates the number of superclients and their corresponding client assignments as the rounds progress. It prioritizes sequentiality, *i.e.* larger groups, during the initial stages, and gradually transitions to parallelism, emphasizing smaller groups in the later stages. This allows easier adaptation to varying numbers of devices and distributions.

Empirical analyses demonstrate the superior performance and convergence speed of the introduced methods compared to the current state-of-the-art algorithms in federated scenarios with various data distributions and tasks. ③ To provide a comprehensive evaluation, **the experimental benchmark is extended to include vision datasets that exhibit feature shifts in addition to label skew, as well as Natural Language Processing (NLP) datasets**.

④ Lastly, the robustness of the algorithm against threats posed by potentially malicious participants is a crucial aspect in the FL paradigm [31], [32], [33]. Malevolent clients may attempt to disrupt the training process by manipulating their input data [34], [35], or even try to infer private information of other clients by exploiting the received global model [36], [37]. To assess whether our novel *client-to-client* sequential training approach introduces any privacy vulnerabilities, in this extension we conduct tests against these attacks and observe that **FedSeq often exhibits higher privacy resistance compared to the widely-used FedAvg** [1], considered the de-facto standard algorithm for Federated Learning.

B. CONTRIBUTIONS

To summarize, our main contributions are the following:

- We introduce FedSeq, a new FL algorithm that learns from groups of sequentially-trained clients (*superclients*).
- Several lightweight procedures to compare the clients' probability distributions, analyzing their impact on the creation of superclients. Extending the previous [28], this work proposes to estimate the distribution of the clients' tasks via *Task2Vec*, eliminating the need for external public data.
- Three grouping strategies are evaluated and compared with the naïve random assignment, showing the impact of group quality on the algorithm convergence.
- To speed up training, we introduce FedAsyncSeq to decrease the need for synchronization between superclients and server, and FedSeq2Par to increase parallelism.
- The extensive empirical analyses and tests demonstrate that the developed approaches outperform the state of the art in terms of convergence performance and speed in both i.i.d. and non-i.i.d. scenarios. This paper further extends the benchmark's scope by incorporating both vision and language datasets.
- We prove the resistance of FedSeq to common attacks against clients' privacy, showing its robustness.

C. PAPER STRUCTURE

The paper is structured as follows. Section II discusses the related works. Section III defines the standard FL problem formulation (III-A) and the proposed method, detailing the definition of superclients (III-B) and the sequential training within superclients (III-C), distinguishing between FedSeq, FedAsyncSeq and FedSeq2Par. Experimental results are analyzed and discussed in Section IV, where the introduced approaches are compared with state-of-the-art baselines in terms of final performance, convergence speed and communication costs. Section IV-C presents the ablation studies. Lastly, privacy concerns and experiments are addressed in Section V.

II. RELATED WORKS

A. DATA HETEROGENEITY IN FL

Federated Learning (FL) [1] is a framework to learn a global model distributedly while preserving the users' data privacy [2], [27], [38]. Training [1] is based on rounds of communication between clients and a server, and the global model is built as the weighted average of the updated parameters (FedAvg) obtained via client-side training on local data. In realistic scenarios, a major challenge is posed by the presence of non-i.i.d. and unbalanced clients' data, often referred to as *statistical heterogeneity* [39], [40]. In those settings, the local optimization objectives drift from each other [13], leading to unstable trends [3] and slower convergence rates [6], [10], [39], [41], [42]. This work preserves FedAvg for the server-side updates aggregation while focusing on enhancing convergence performance.

1) CLIENT-SIDE APPROACHES

Several works addressed data heterogeneity issues via client-side training regularization. SCAFFOLD [13] mitigates the effects of the clients' drifts through control variates, while [10], [43] minimize the gap between local and global model parameters to limit the impact of the clients' updates. FedAlign [44] aligns the Lipschitz constants of the models' last blocks to promote smooth optimization and global consistency. Other approaches leverage the *Alternating Direction Method of Multipliers* to asymptotically align local and global objectives [45], [46], [47]. Among those, FedDyn [15] dynamically modifies the local loss function so that local and global stationary points coincide at convergence. Another recent direction studies generalization through the lens of the loss landscape. By seeking flat minima during local training, FedSAM [3], [16], FedSpeed [17], and FedSMOO [48] improve the poor generalization of models in heterogeneous scenarios. Since FedSeq does not modify the client-side training, it can be applied alongside any of these approaches.

2) SERVER-SIDE APPROACHES

While client-side regularization is effective for mitigating client drift and enhancing local model learning, server-side aggregation is not to be overlooked. Using server-side optimizers [6], [40], [49], [50] allows coping with FedAvg's lack of adaptivity. Server-side momentum [40], [51], [52] addresses the bias towards recently observed clients by leveraging the memory of past updates. Other studies utilize global momentum to guide local updates [53], [54], [55], [56]. Additionally, [3] and [57] suggest to aggregate global updates across rounds to increase the robustness of the model to distribution shifts. The proposed revised orchestration of the training process can effortlessly integrate with any of these techniques.

As the learned local model under-represents the deducible patterns from the missing classes, [11] shows how sharing a small set of public data among the clients leads to notable improvements. A similar approach is followed by [58], where the public data enables knowledge distillation. Reference [59] leverages public unlabelled data to learn a general representation robust to domain shifts. Similarly, FedSeq keeps the public data on the server side, with the different purpose of using it to estimate the clients' data distribution in a privacy-compliant way. Unlike [11], [58], and [59], such data is never used for training.

3) FEDERATED MULTI-TASK LEARNING

Another line of works tackles the problem from a multitask perspective [60], where each client with its own data distribution is seen as a different task [18], [61]. In [21], [22], [23], [24], and [25], clients with similar tasks are clustered together and a specific model is assigned to each cluster. Reference [8] leverages the local style information to identify and group clients having similar distributions. Other works build clusters based on the edge systems complexity and available resources [26], [62]. Following the same approach of [21], [24], and [23], FedSeq approximates the clients' data distribution via the locally trained models, which is later used to build groups of dissimilar clients. Reference [19] uses the relatedness among clients' tasks to improve weight aggregation. Here, Task2Vec [29] embeddings, based on the Fisher information matrix (FIM) of fine-tuned local models, are leveraged to capture similarities among the clients' tasks without needing external public datasets.

4) ANTI-CLUSTERING FL

FedSeq uses clustering techniques to effectively group clients with distant distributions, resulting in a homogeneous underlying dataset within each group. This approach relates to the "*anti-clustering*" literature [63], [64], whose goal is to build similar groups from dissimilar elements [65]. Building upon the strategy of FedGSP [30], which dynamically expands the number of client groups in each round to enhance parallelism, we propose merging this approach with the data estimation strategies and grouping techniques unique to FedSeq. This combination results in FedSeq2Par, an approach that further increases parallelism.

B. BEYOND SYNCHRONOUS SERVER-CLIENT FEDERATED LEARNING

1) PEER-TO-PEER FL

Peer-to-peer (p2p) FL [66], [67] is a decentralized approach where clients communicate directly with each other in order to learn global models, eliminating the need for a central server. In particular, FedSeq shares some common traits with [68], which introduces two network topologies based on cyclic model parameters exchange among clients to enhance performance in heterogeneous scenarios, namely FedCyclic and FedStar. Similarly, our approach enables model sharing among clients within the same superclient. Unlike such works, FedSeq retains the central server as a proxy between clients, while ensuring communication costs equivalent to those of FedAvg.

2) ASYNCHRONOUS FL

Asynchronous FL was introduced to handle stragglers and heterogeneous latency [69], [70]. In this scenario, the server does not wait for all devices to send back their updates but keeps aggregating the models as they arrive. Several approaches rely on a parameter accounting for staleness [69], [71], leverage gradient compression techniques to reduce the communication latency [72], or store the early updates for a given timeframe and discard the late ones [73]. Similarly to these methods, the server in FedAsyncSeq does not wait for all superclients to return their updates but allows faster ones to continue training for multiple rounds before the averaging step. Differently from the standard asynchronous approach, updates are not averaged as they come, but after a fixed window of rounds so that the exchanged models are trained on a larger portion of data.

C. PRIVACY IN FEDERATED LEARNING

A primary objective within the FL framework is to ensure the algorithm's resilience against potential threats posed by malicious participants. However, it's essential to recognize that the FL paradigm, in its current form, is not entirely impervious to threats [74], [75]. Malevolent clients may *poison* the training process by altering the input data [34], [35], worsening the capacity of the global model to acquire new useful knowledge. An attacker might *reconstruct clients' private data* by exploiting the incoming update model [36]. As an example, [76] leverages a GAN (Generative Adversarial Network) [77] to reconstruct other users' personal data, while [78] uses a GAN to ensure the quality of the data that the attacker aims to reconstruct and [79] tries to infer characteristics of the clients with ad-hoc classifiers. Additionally, a malicious server might put in place *label and feature fishing* attacks by intentionally modifying some parameters of the global model [80]. For an in-depth discussion on threats and attacks in FL, we refer the readers to [32] and [33]. Common defense techniques involve using differential privacy [81], [82], or mixing fragments of the local updates before sending them to the server [83]. This work explores some of those attacks against FedSeq and shows that the proposed novel *client-to-client* sequential training approach is robust in terms of privacy compared to FedAvg.

III. METHOD

This section details the problem formulation (Sec. III-A), and the components of the proposed method, distinguishing between FedSeq, FedAsyncSeq and FedSeq2Par (Secs. III-B and III-C). The overall procedure is outlined in Fig. 2, highlighting the extension's additions, while the used symbols are summarized in Table 1.

A. PROBLEM FORMULATION

The objective of FL is to learn a global model $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output space respectively, and $\theta \in \mathbb{R}^d$ the model parameters. The server communicates with a subset of active clients sampled uniformly at random from a set of users C across T rounds. In cross-device settings [27], the number of clients $K := |C|$ is in the order of millions. Each client $k \in C$ has access to a local private dataset $\mathcal{D}_k$ of the form $\{x_i, y_i\}_{i=1}^{n_k}$ where $x_i \in \mathcal{X}$ is the input data point, $y_i \in \mathcal{Y}$ its label and $n_k = |\mathcal{D}_k|$ the local

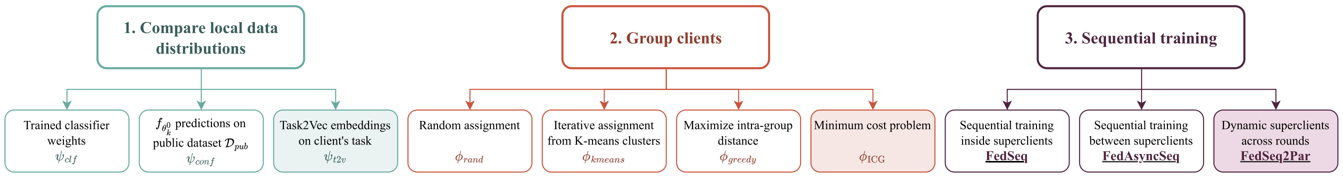

1. Compare local data distributions
2. Group clients
3. Sequential training
Trained classifier weights ψ_{clf}
$f_{\theta_0^k}$ predictions on public dataset $\mathcal{D}_{pub}$ ψ_{conf}
Task2Vec embeddings on client's task ψ_{t2v}
Random assignment ϕ_{rand}
Iterative assignment from K-means clusters ϕ_{kmeans}
Maximize intra-group distance ϕ_{greedy}
Minimum cost problem ϕ_{ICG}
Sequential training inside superclients FedSeq
Sequential training between superclients FedAsyncSeq
Dynamic superclients across rounds FedSeq2Par

FIGURE 2. Summary of the method components. Extensions with respect to [28] are highlighted with colored boxes. Best seen in colors.

TABLE 1. Summary of main introduced symbols.

Notation	Description
f	Global model
k	Client index
θ	Global model parameters
θ_t	Global model parameters at round t
θ_t^k	k-th client updated model parameters
C	Set of clients
C_t	Set of clients selected at round t
K	Number of clients ($\|C\|$)
C	Fraction of clients selected at each round
$\mathcal{D}_k$	k-th client local dataset
$\tilde{\mathcal{D}}_k$	Estimate of k-th client local dataset
n_k	Number of local samples ($\|\mathcal{D}_k\|$)
T	Number of rounds
E_k	Local training epochs
i	Superclient index
$\mathcal{S}$	Set of superclients
$\mathcal{S}_t$	Set of superclients selected at round t
N_S	Number of superclients ($\|\mathcal{S}\|$)
S_i	i-th superclient
C_S	Set of clients belonging to superclient S
K_{S_i}	Number of clients in superclient S_i ($\|C_S\|$)
$K_{S,max}$	Maximum number of clients in each superclient
ψ	Clients distribution estimator
ϕ	Grouping method
τ	Distance metric
e	Pretraining number of epochs
E_S	Loops within each superclient

dataset cardinality. Due to communication constraints [2], only a fraction C_t of clients is randomly selected without replacement for training at round $t \in [T]$. Each client $k \in C_t$ receives the current global model parameters θ_t from the server and computes the update θ_{t+1}^k by minimizing the local empirical risk $L_k(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_k}[\ell_k(f_{\theta_t};(x,y))]$ where ℓ_k is the loss function of the k-th client, *e.g.* the cross-entropy loss. The updates $\{\theta_{t+1}^k\}_{k\in C_t}$ are then sent to the server to be aggregated into the global model $f_{\theta_{t+1}}$. The global training objective is

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

where $n = \sum_{k\in C_t} n_k$. The de-facto standard algorithm for solving the FL objective in Eq. 1 is FedAvg [1], which computes a weighted average of the clients' updates as $\theta_{t+1} \leftarrow \sum_{k\in C_t} {}^{n_k}/_{n}\, \theta_{t+1}^k$. As noted by [49], this is equivalent to performing one step of stochastic gradient descent (SGD) with unitary learning rate as $\theta_{t+1} \leftarrow \theta_t - \sum_{k\in C_t} {}^{n_k}/_{n} (\theta_t - \theta_{t+1}^k)$, where the difference between the local model and the round initialization acts as pseudo-gradient for the client's direction.

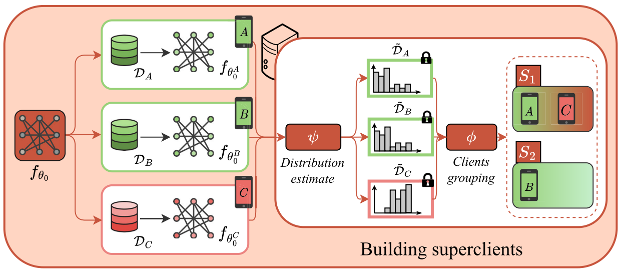


FIGURE 3. FedSeq pre-training phase to build superclients. a) The initial random global model f_{θ_0} is sent to all the clients, which train it using their local data $\mathcal{D}_k\ \forall k \in C$. b) The local data distributions are estimated (ψ) using the clients' updates while preserving their privacy. c) Based on the grouping strategy ϕ, clients are assigned to N_S superclients. Best seen in colors.

As shown in [6], in realistic scenarios, clients likely do not draw data from the same underlying distribution, namely $\mathcal{P}(D_i) \neq \mathcal{P}(D_j)\ \forall i \neq j \in C$, resulting in slower and unstable convergence [39]. More in general, $f_{\theta_\cdot^k} \neq f_{\theta_\cdot}\ \forall k \in C$ [23]. To address the challenges arising from data heterogeneity and speed up convergence, FedSeq introduces modifications to the training orchestration process. Specifically, clients with diverse data distributions are grouped into *superclients* $\{S_i\}_{i=1}^{N_S}$, aiming to minimize the divergence in distribution among superclients. Sequential training is then performed by clients within the same group. Intuitively, this approach allows local models to accumulate knowledge from the overall data distribution, even when client datasets exhibit significant heterogeneity.

B. BUILDING SUPERCLIENTS

This section details how to create a superclient S from users with diverse local distributions while respecting the privacy constraints, *i.e.* without accessing clients' data directly. Assigning clients to equally-sized groups while minimizing distribution distance is a challenging problem similar to the bin packing problem [84], and is NP-hard in nature. Thus, this work proposes using multiple greedy strategies to estimate the local distributions in a privacy-preserving way and solve the clients' clustering problem, being flexible towards dynamic and constantly evolving FL environments. In this Section, we introduce different grouping criteria G_S which are based on *i)* a *client distribution estimator* $\psi_{(.)}$, providing privacy-preserving statistics on the local data distribution, *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.* $G_S := \{\psi_{(.)};\tau;\phi_{(.)}\}$. The approach is depicted in Fig. 3.

1) CLIENTS DISTRIBUTION COMPARISON

The model f_θ can be defined as a combination of a deep feature extractor $h_{\theta_{\text{feat}}} : \mathcal{X} \rightarrow \mathcal{Z}$ and a classifier $g_{\theta_{\text{clf}}} : \mathcal{Z} \rightarrow \mathcal{Y}$, where $\theta = \{\theta_{\text{feat}}, \theta_{\text{clf}}\}$ is the entire set of model parameters and $\mathcal{Z}$ the output feature space. The classification output is given by $g \circ h : \mathcal{X} \rightarrow \mathcal{Y}$, where we drop the subscripts to ease the notation. FedSeq exploits a *pre-training phase* to estimate the users' data distribution. Each client $k \in C$ trains a common random model θ_0 on its dataset $\mathcal{D}_k$ for e epochs, resulting in $f_{\theta_0^k}$, which serves as a starting point for the following distribution estimation approaches. The proposed client distribution estimators are

- ψ_{clf}: as the **model classifier** is biased towards the training data [85], its parameters $\theta^k_{0,\text{clf}}$ serve as a proxy for the client's local data distribution.
- ψ_{conf} relies on the **predictions of the local models on a server-side public dataset**, *i.e.*, $\{f_{\theta_0^k}(z) =: f_0^k,\ z \in \mathcal{D}_{pub},\ k \in C\}$. $\mathcal{D}_{pub}$ contains J samples for each class $c \in [N_C]$. The predictions are averaged by class as $p_{k,c} = \frac{1}{J}\sum_{x\in\mathcal{D}_c} f_0^k(x)$, where $\mathcal{D}_c \subset \mathcal{D}_{pub}$ contains only samples of class c. The k-th client's *confidence vector* is defined as:

$$p_k := \text{softmax}(\{p_{k,1}, \ldots, p_{k,N_C}\}) \in [0,1]^{N_C} \tag{2}$$

  Since the k-th model's predictions are favorable towards the majority of the classes seen in $\mathcal{D}_k$ [86], p_k is an acceptable privacy-preserving representation of $\mathcal{D}_k$.

Although ψ_{conf} has proven to be the most effective approach [28], it relies on the availability of a public dataset that accurately reflects the overall data distribution — a requirement that can be challenging to meet. Therefore, extending [28], this study introduces an alternative method to overcome this limitation, without introducing any privacy liabilities.

- ψ_{t2v}: based on *Task2Vec* [29], which extracts **vectorial representations of given tasks** based on an approximation of the Fisher Information Matrix (FIM), defined as

$$F := \mathbb{E}_{(x,y)\sim f_\theta(x,y)}[\nabla_\theta \log f_\theta(y|x) \nabla_\theta \log f_\theta(y|x)^T]. \tag{3}$$

  The FIM serves as a metric for the information content of a parameter regarding the joint distribution $f_\theta(x,y)$. If it has limited influence on the classification performance for a specific task, its corresponding entry in the FIM will be low. Thus, the FIM represents the task itself, here corresponding to each client's local dataset. Starting from a pre-trained set of weights θ_0, the classifier is fine-tuned on $\mathcal{D}_k$ and the FIM is computed on the feature extractor parameters. The resulting representations are demonstrated to capture taxonomic and semantic similarities between tasks.

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

2) GROUPING CLIENTS

$\mathcal{D}_S = \bigcup_{k\in C_S} \mathcal{D}_k$ is defined as the union of the data from the clients $C_S \subset C$ belonging to a superclient S. The 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 $K_{S,max}$ per superclient. Given $\psi_{(.)}$ and τ, FedSeq [28] approximates the solution of the problem using:

- ϕ_{rand}, a naïve yet practical method that **randomly** assigns clients to superclients until the stopping criterion is met.
- ϕ_{kmeans}: **K-means** [87] is first applied to obtain N_S homogeneous clusters. Each superclient is formed by iteratively extracting one client at a time from each cluster, until $|\mathcal{D}_S| \geq |\mathcal{D}_S|_{min}$ and $K_S \leq K_{S,max}\, \forall S$.
- ϕ_{greedy} initially assigns one random client $k_i \in C$ to the superclient S. The next $k_j \in C \setminus \{k_i\}$ is chosen so as **the distance between** k_i **and** k_j **is maximized**, *i.e.* $\max_{j\in[K]} \tau(\mathfrak{D}_{k_i}, \mathfrak{D}_{k_j})$. The process is repeated by iteratively maximizing $\tau(\mathfrak{D}_j, \frac{1}{|S|}\sum_{i\in|S|}\mathfrak{D}_i)$, with $|S|$ being the cardinality of S, until the defined constraints are met.
- While highly effective [28], the best-performing ϕ_{greedy} is hindered by its iterative nature, leading to slower execution. In response, this work adapts the faster ***Inter-Cluster Grouping*** (ICG) algorithm from [30] to our approach (ϕ_{ICG}). Differently from ICG that requires superclients of equal size, FedSeq relaxes this constraint by redistributing the unassigned clients.

At the end of this procedure, we obtain a set $\mathcal{S}$ of N_S superclients, where each superclient S_i includes $K_{S_i} := |C_{S_i}| \leq K_{S,max}$ clients and $|\mathcal{D}_{S_i}| \geq |\mathcal{D}_S|_{min}$ data points, with $\sum_{S_i\in\mathcal{S}} K_{S_i} = K$.

C. SEQUENTIAL TRAINING

This section introduces three alternatives to leverage *sequential training* within the created superclients, namely FedSeq, FedAsyncSeq and FedSeq2Par. The approaches are summarized in Algorithm 1.

1) FEDSEQ

As showed in Fig. 4, the clients $\{k_{i,1}, \ldots, k_{i,|S_i|}\} \in C$ belonging to the superclient $S_i\ \forall i \in [N_S]$ form a chain performing sequential training. At each round t, the server selects a subset of $S_t \in \mathcal{S}$ superclients. Within each superclient S_i, the first device $k_{i,1}$ receives the global model f_{θ_t} from the server and locally trains it for E_k epochs on $\mathcal{D}_{k_{i,1}}$. The updated parameters $\theta_{t+1}^{k_{i,1}}$ are sent to the next client of the sequence $k_{i,2}$. Such training procedure continues until the last client $k_{|S_i|}$ updates the received model and sends it back to the server. The passage over all the clients of the chain can be repeated E_S times allowing a ring communication strategy. However, $E_S \neq 1$ leads to an increase in communication as multiple messages have to be exchanged between clients, which is why we discourage this approach and set $E_S = 1$ in our experiments. On the server-side, the superclients updates are averaged following Eq. 1. Intuitively, by training the model over multiple clients' data before the averaging step, we simulate the existence of a larger, homogeneous dataset.

Algorithm 1 FedSeq , FedAsyncSeq and FedSeq2Par

```
Require: $f_{\theta_0}$, $G_S$, $K_{S,max}$, $|\mathcal{D}_S|_{min}$. Epochs $e$, $E_k$, $E_S$. $T$ rounds. Clients $C$.
    Fraction $C$ of superclients selected at each round.
    Growth function and parameters $f_{gr}$, $\alpha_{gr}$ and $\beta_{gr}$.
 1: $\mathcal{S} \leftarrow$ CREATE_SUPERCLIENTS$(f_{\theta_0}, G_S, e, K_{S,max}, |\mathcal{D}_S|_{min}, K,$
    $f_{gr}(\alpha_{gr}, \beta_{gr}; t))$
 2: $N_S \leftarrow |\mathcal{S}|$
 3: $\Theta \leftarrow [\theta_0, \dots, \theta_0]_{1\times CN_S}$, $w \leftarrow [0, \dots, 0]_{1\times CN_S}$
 4: for $t = 1$ to $T$ do
 5:     if $f_{gr}(t, \alpha_{gr}, \beta_{gr}) > |N_S|$ then
 6:         $\mathcal{S} \leftarrow$ CREATE_SUPERCLIENTS $(f_{\theta_0}, G_S, e, K, f_{gr}(\alpha_{gr}, \beta_{gr}; t))$
 7:         $N_S \leftarrow |\mathcal{S}|$
 8:     end if
 9:     $\mathcal{S}_t \leftarrow$ Subsample fraction $C$ of $N_S$ superclients
10:     for $S_i \in \mathcal{S}_t$ in parallel do
11:         Shuffle clients in $S_i$
12:         $\theta_t^{S_i,0} \leftarrow \theta_t$ { FedSeq and FedSeq2Par }
13:         $\theta_t^{S_i,0} \leftarrow \Theta[i]$
14:         for $e_S = 1$ to $E_S$ do
15:             $\theta_{t+1}^{S_i} \leftarrow$ SEQUENTIAL_TRAINING$(\theta_t^{S_i,0}, E_k)$
16:         end for
17:         $\Theta[i] \leftarrow \theta_{t+1}^{S_i}$, $w_i \leftarrow w_i + |\mathcal{D}_{S_i}|$
18:     end for
19:     $\theta_{t+1} \leftarrow$ FedAvg$(\{\theta_{t+1}^{S_i}, \forall S_i \in \mathcal{S}_t\})$ { FedSeq and FedSeq2Par }
20:     if $t$ mod $N_S = 0$ then
21:         $\theta_{t+1} \leftarrow \sum_i \frac{w_i}{w}\Theta[i]$, $w = \sum_i w_i$
22:         $\Theta \leftarrow [\theta_{t+1}, \dots, \theta_{t+1}]_{1\times CN_S}$, $w \leftarrow [0, \dots, 0]_{1\times CN_S}$
23:     end if
24: end for
```

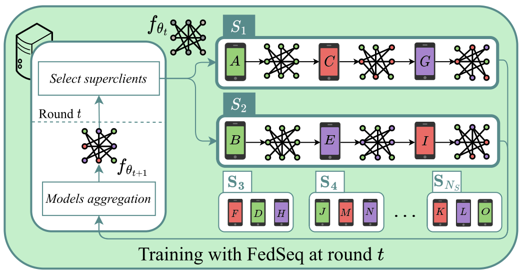


FIGURE 4. Sequential training with FedSeq. At each round t, a subset of superclients (here S_1 and S_2) is selected and receives θ_t, which is trained sequentially by the clients. Final updates are sent back to the server, where they are aggregated with FedAvg. Best seen in colors.

2) FedAsyncSeq

In realistic scenarios, synchronous federated training can become impractical, especially when considering factors such as the latency of slower devices. The delays can be further exacerbated by FedSeq since there is no control over the capabilities of clients' systems, and multiple slow clients may be grouped together within the same superclient, leading to a significant increase in server-side waiting time. To mitigate the challenges posed by latency and ensure efficient training, we propose **FedAsyncSeq**, which **leverages sequentiality at the superclient level** while allowing asynchronous updates to be merged (Fig. 5). Instead of aggregating the models after *every* round, the server does so every R rounds. During this time frame, the model received by each superclient S_i is instantly sent to another random superclient S_j (where $i \neq j$). This approach allows the fastest chains to continue with additional training iterations instead of waiting for slower ones. After every R rounds, the most recent updates from each chain of superclients, which may originate from distinct rounds, are combined. This approach shares similarities with asynchronous settings, where models are aggregated as soon as they become available. It is worth noting that R can potentially equal T, meaning that the updated models are only averaged at the end of training. This strategy reduces the number of aggregation and synchronization steps with the server while enabling the model to potentially observe the entire dataset before averaging. This brings us closer to the centralized setting while still leveraging FL's parallelism.

3) FedSeq2Par

Sequential training of the model through a long chain of clients with highly diverse data distributions may lead to catastrophic forgetting [88]. This refers to the model forgetting the knowledge acquired from the initial users while becoming overly specialized to the most recently seen datasets [30]. Additionally, maintaining a static sequence of clients within each superclient may result in the model learning information based on the order of the clients, introducing biases or unintended patterns. Building upon [30], this extension introduces the *Sequential-to-Parallel* (STP) approach in FedSeq2Par to overcome these issues. Sequential training is exploited during the initial stages, facilitating information exchange among heterogeneous clients. **Parallelism is gradually introduced by incrementally increasing the number of superclients** and reducing their sizes, promoting faster convergence. The dynamic creation of superclients allows accounting for new clients, while eliminating potential biases related to the static nature of superclients. Formally, in each training round t, STP dynamically constructs an increasing number of superclients $\{S_i\}_{i=1}^{f_{gr}(\alpha_{gr}, \beta_{gr}; t)}$, where $f_{gr}(\cdot, \cdot; \cdot)$ is a non-decreasing growth function dependent on the training round t, and the hyperparameters $\alpha_{gr}, \beta_{gr} \in \mathbb{R}^+$ control the growth rate and the initial number of superclients respectively. The choice of f_{gr} is critical for the behavior of STP. As in [30], we consider three possible growth functions: *linear*, allowing for a smooth growth; *logarithmic*, promoting an initial faster dynamic; *exponential*, with slower changes at first,

$$f_{linear}(\alpha_{gr}, \beta_{gr}, t) = \beta_{gr}[\alpha_{gr}(t-1)+1], \tag{4}$$

$$f_{log}(\alpha_{gr}, \beta_{gr}, t) = \beta_{gr}[\alpha_{gr}\ln t + 1], \tag{5}$$

$$f_{exp}(\alpha_{gr}, \beta_{gr}, t) = \beta_{gr}(1+\alpha_{gr})^{t-1}. \tag{6}$$

IV. EXPERIMENTS

This section introduces the empirical evaluations of the proposed approaches, comprising ablation studies (Sec. IV-C) and comparison with the state of the art (Sec. IV-B).

A. DATASETS

FedSeq, FedAsyncSeq and FedSeq2Par are evaluated on both vision and NLP datasets. Additionally to the Cifar10 and Cifar100 [89] datasets proposed in our previous [28], this extension aims to enlarge the paper's scope and introduces

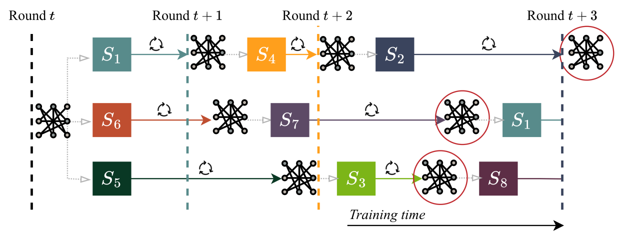


FIGURE 5. Training with FedAsyncSeq with $R = 3$. At round t, the global model is sent to the selected superclients $\{S_1, S_5, S_6\}$, having varying latency (*full arrows*). The first superclient to complete training (S_1) marks the start of the new round $t+1$. As soon as the server receives the updated model, it sends it to another superclient (*e.g.*, from S_1 to S_4). Within the time it takes for S_5 to finish training, the faster S_1 and S_4 can also complete theirs. After R rounds, the latest updates from each chain of superclients (*circled in red*) are combined, and the process begins anew. Best seen in colors.

Femnist [90] for image classification, Shakespeare [90] for next character prediction and StackOverflow [91] for next word prediction, all widely used as FL benchmarks [10], [15], [49]. It is important to note that *all* datasets exhibit heterogeneous distributions concerning label skew. Differently from the CIFAR datasets, FEMNIST also presents notable feature shifts attributed to diverse calligraphy styles depicting the same letter or number. Moreover, the local dataset cardinality significantly varies across clients in both FEMNIST and the NLP datasets.

In order to set up a heterogeneous scenario for Cifar10 and Cifar100, the local class distribution is sampled from a Dirichlet distribution with $\alpha \in \{0, 0.2, 0.5\}$ [40]. Both Cifar datasets are divided into 500 clients with 100 images each. The IID and non-IID ("NIID" for short) data distributions of Femnist introduced in [90] follow the writers' ownership, *i.e.* each client is a distinct writer. The NIID split accounts for both label skew and feature shift. The IID and NIID distributions of Shakespeare [90] reflect some Shakespearean characters, with 100 clients owning around 3,743 samples each, while the implementation of StackOverflow follows [49]. Femnist and StackOverflow better represent realistic cross-device settings thanks to a larger number of clients: $K = 3{,}500$ and $40k$ respectively. Additional information can be found in Appendix A of the supplementary material. As proposed by previous works [3], [15] accounting for the learning trends instability, the results are averaged over the last 25 rounds on Shakespeare and 100 rounds on Cifar10/100 and Femnist. As done in [49], due to its prohibitively large number of clients and examples, testing on StackOverflow full dataset is performed only at the end and a subsample is used during training.

B. COMPARISON WITH STATE-OF-THE-ART FL ALGORITHMS

1) ALGORITHMS

To validate the effectiveness of the proposed approaches, this study conducts a comparison with state-of-the-art (SOTA) algorithms for heterogeneous FL. In addition to the standard FedAvg [1] described in Sec. III, FedSeq and its variants are tested against FedProx [10], which adds a regularization term in the local loss function to encourage proximity between clients' and server's model parameters, Scaffold [13], addressing the client drift via stochastic variance reduction, FedDyn [15], that aligns local and global stationary points, and FedCyclic [68], which cyclically exchanges models across clients without relying on a central server. FedCyclic can be seen as the extreme case of FedSeq with $N_S = 1$ and no server-side aggregation. FedCyclic was chosen over FedStar (from the same paper [68]) due to the latter's impractical communication overhead in realistic scenarios. FedStar indeed requires each client to send its updates to *all* the other participants, leading to an exponential increase in communication costs. To ensure a fair comparison, FedCyclic is not trained on all clients at each round but randomly selects a fraction C of the available ones. The same applies to all the other approaches and, most importantly, we ensure that the number of model updates within rounds is the same for all the compared methods. FedSeq and FedSeq2Par build superclients using ψ_{t2v} and the best corresponding $\phi_{(\cdot)}$ (see Sec. IV-C). Local pre-training runs for 10 epochs chosen from $\{1, 5, 10, 20, 30, 40\}$ (see Appendix B of the supplementary material for details). We select $R = N_S$ in FedAsyncSeq.

2) FINAL PERFORMANCE

Table 2 shows that FedSeq and its extensions are either competitive or outperform other SOTA algorithms on all tasks and datasets, especially on severe heterogeneous data distributions. We point out that both Scaffold and FedDyn require stateful clients, and Scaffold doubles the size of the communicated message, differently from this paper's approaches. Being the extreme case of FedSeq with $N_S = 1$, FedCyclic reaches similar performances on most of the datasets, implying that having one single superclient containing all clients does not dramatically increase performances and instead hugely increments the amount of training time, as each client needs to wait for the previous one's update. Focusing on the extensions of FedSeq, both FedAsyncSeq and FedSeq2Par improve the baseline's performances on all tasks. We note that aggregating superclients updates every N_S rounds not only requires less frequent synchronization between clients and server, but also improves the reached accuracy. FedSeq2Par achieves the best results in most cases, exploiting the benefits of sequential training and parallelism, being the second best to FedDyn only in the case of $\alpha = 0.2$ and $\alpha = 0.5$ in CIFAR100. However, differently from FedSeq2Par, FedDyn relies on stateful clients, posing a significant challenge for real-world deployments with billions of edge devices [3], [92]. In such large-scale settings, individual devices are unlikely to be called upon for multiple training rounds. This transience renders their local states obsolete quickly, compromising their effectiveness in subsequent training iterations. The results obtained by FedDyn on the more realistic FEMNIST (3,500 clients) and STACKOVERFLOW ($40k$ clients) datasets underscore this point. On FEMNIST, FedDyn outperforms the baseline FedAvg by only ≈ 0.4 points in accuracy. In contrast,

TABLE 2. Comparison with state-of-the-art FL algorithms. Color coding: first, second and third best results.

Algorithm	CIFAR10			CIFAR100			FEMNIST		SHAKESPEARE		STACKOVERFLOW
	$\alpha=0$	$\alpha=0.2$	$\alpha=0.5$	$\alpha=0$	$\alpha=0.2$	$\alpha=0.5$	NIID	IID	NIID	IID	-
Centralized		85.64$_{\pm0.07}$			54.97$_{\pm0.19}$		87.52$_{\pm0.27}$		52.00$_{\pm0.02}$		28.50$_{\pm0.20}$
FedAvg	71.27$_{\pm0.29}$	76.32$_{\pm0.36}$	77.39$_{\pm0.43}$	42.68$_{\pm0.22}$	48.79$_{\pm0.55}$	49.51$_{\pm0.61}$	81.55$_{\pm0.11}$	83.06$_{\pm0.12}$	48.68$_{\pm0.12}$	48.50$_{\pm0.07}$	24.68$_{\pm0.15}$
FedProx	71.52$_{\pm0.08}$	76.21$_{\pm0.50}$	77.38$_{\pm0.57}$	42.83$_{\pm0.18}$	48.84$_{\pm0.65}$	49.44$_{\pm0.49}$	81.55$_{\pm0.05}$	83.07$_{\pm0.05}$	48.73$_{\pm0.12}$	48.51$_{\pm0.15}$	24.59$_{\pm0.19}$
SCAFFOLD	78.82$_{\pm0.15}$	78.02$_{\pm1.13}$	78.51$_{\pm0.24}$	42.17$_{\pm0.10}$	51.06$_{\pm0.03}$	51.03$_{\pm0.12}$	82.56$_{\pm0.07}$	82.70$_{\pm0.01}$	50.91$_{\pm0.19}$	50.91$_{\pm0.12}$	26.10$_{\pm0.15}$
FedDyn	**83.31$_{\pm0.15}$**	82.31$_{\pm0.41}$	82.97$_{\pm0.40}$	**50.35$_{\pm0.27}$**	**53.50$_{\pm0.76}$**	**54.32$_{\pm0.63}$**	81.95$_{\pm0.42}$	82.00$_{\pm0.13}$	**51.77$_{\pm0.03}$**	**51.94$_{\pm0.09}$**	25.51$_{\pm0.02}$
FedCyclic	82.45$_{\pm0.18}$	**82.61$_{\pm0.27}$**	**83.49$_{\pm0.08}$**	47.46$_{\pm0.42}$	49.93$_{\pm0.16}$	50.47$_{\pm0.27}$	85.46$_{\pm0.05}$	**87.47$_{\pm0.09}$**	50.25$_{\pm0.03}$	50.68$_{\pm0.03}$	26.44$_{\pm1.68}$
FedSeq (ours)	81.89$_{\pm0.28}$	82.19$_{\pm0.26}$	82.77$_{\pm0.12}$	45.87$_{\pm0.45}$	49.26$_{\pm0.40}$	49.63$_{\pm0.42}$	**87.11$_{\pm0.03}$**	**87.48$_{\pm0.03}$**	51.70$_{\pm0.13}$	51.82$_{\pm0.00}$	**28.91$_{\pm0.21}$**
FedAsyncSeq (ours)	**83.03$_{\pm0.31}$**	**83.17$_{\pm0.27}$**	**83.57$_{\pm0.22}$**	**50.23$_{\pm0.11}$**	**51.39$_{\pm0.18}$**	**51.27$_{\pm0.24}$**	**86.96$_{\pm0.07}$**	87.20$_{\pm0.01}$	**51.82$_{\pm0.13}$**	**51.88$_{\pm0.04}$**	**27.44$_{\pm0.35}$**
FedSeq2Par (ours)	**83.68$_{\pm0.14}$**	**84.21$_{\pm0.40}$**	**84.26$_{\pm0.06}$**	**51.46$_{\pm0.23}$**	**51.44$_{\pm0.02}$**	**51.72$_{\pm0.10}$**	**87.58$_{\pm0.15}$**	**87.95$_{\pm0.05}$**	**52.75$_{\pm0.20}$**	**52.71$_{\pm0.05}$**	**29.79$_{\pm0.33}$**

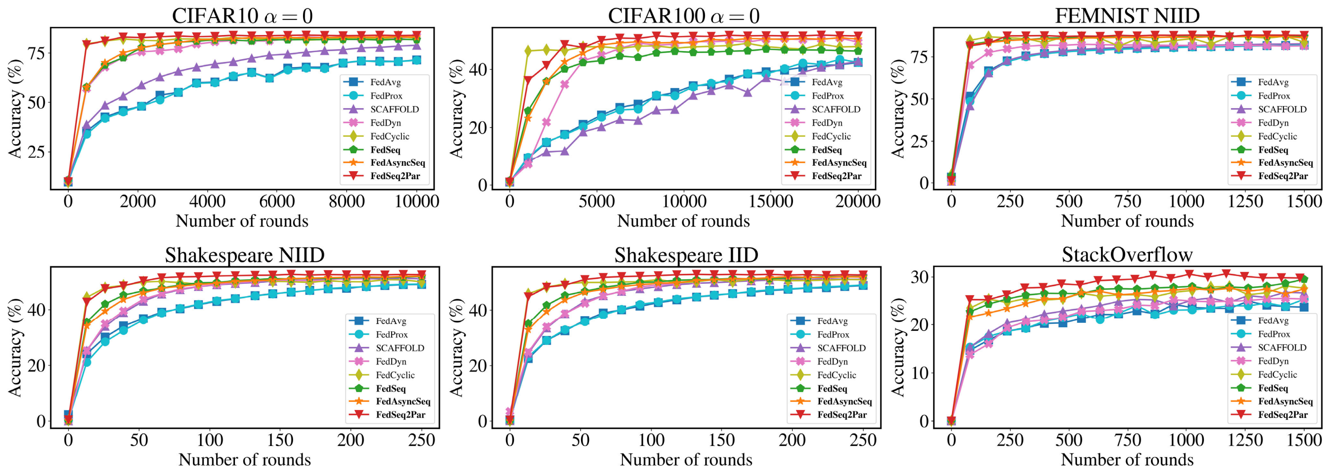


FIGURE 6. Accuracy convergence plots of FedSeq, FedSeq2Par, FedAsyncSeq (in bold) and SOTA algorithms on vision and NLP datasets. On average, FedSeq2Par is the best-performing algorithm. All the proposed approaches can be distinguished for their improved speed. Best seen in colors. Full results are reported in the Appendix E of the supplementary material.

FedSeq's variants, especially FedSeq2Par, achieve a significant improvement of +6 points over FedAvg. On the even larger STACKOVERFLOW dataset, FedDyn shows a loss of ≈ 0.4 points compared to the baseline, while FedSeq2Par exhibits a gain of ≈ 1.3 points. These results confirm the limitations of FedDyn in real-world cross-device scenarios, where its reliance on stateful clients becomes a significant disadvantage.

3) CONVERGENCE SPEED

Fig. 6 evidently shows that FedSeq and its extensions not only achieve superior results but also exhibit accelerated performance. Fig 7 compares the rounds necessary to each algorithm to reach 70% and 90% of the centralized accuracy on Cifar10/100 and Femnist. Table E.1 in Appendix E of the supplementary material integrates the results for the other datasets. FedSeq consistently demonstrates significant improvements in convergence speed across all tasks, achieving a speed-up factor of over 18x on Femnist and 10x on StackOverflow, presenting high cross-device variability. Achieving superior overall accuracy, asynchronous training, and reduced latency compared to FedSeq does not compromise the convergence speed of FedAsyncSeq. FedSeq2Par notably improves convergence speed on all tasks and datasets through its STP approach. FedDyn enhances the convergence rates of FedAvg but experiences parameter explosion in highly imbalanced settings [92], requiring gradient clipping techniques. Among the considered methods, FedCyclic achieves comparable or better results than FedSeq2Par. However, it is important to note that FedCyclic, as an extreme case of FedSeq with $N_S = 1$, eliminates any form of parallelism inherent in distributed and FL settings. The results highlight the significance of sequential training for rapid convergence in initial rounds, while the superior performance of FedSeq2Par in later stages underlines the role of parallelism in achieving both improved final performance and convergence speed up.

4) COMMUNICATION COST

Communication is the main bottleneck of federated training [93], due to the overload of the networks and message size. Thus, when comparing the performance of FL algorithms, their impact on the communication cost is of the utmost importance. As already shown in Sec. IV-B3, our methods speed up the convergence, implying that fewer communication rounds are needed to reach a target performance. This study additionally compares the number of client-server exchanges required by the proposed method (FedSeq) with leading SOTA algorithms. Table 3 demonstrates that FedSeq achieves **less network communication** thanks to its client-to-client approach. Similar analyses can be easily extended to FedSeq2Par and FedAsyncSeq. Given the total number of clients K, the fraction selected at each round C, the total number of superclients N_S, and the rounds T, we first analyze the case in which all superclients are equally sized,

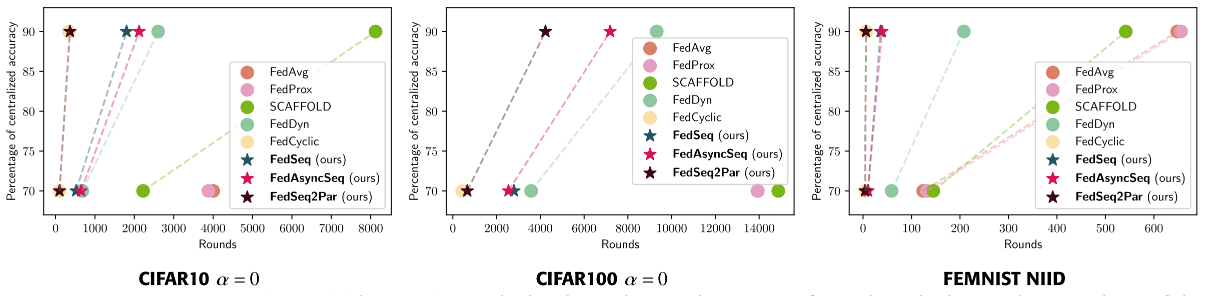


FIGURE 7. Convergence rates in non-i.i.d. scenarios. Each plot shows the rounds necessary for each method to reach 70% and 90% of the centralized accuracy. Not all the algorithms reach the 90% target (missing line). Our methods (*in bold, stars*) outperform the others in all settings. Best seen in colors.

TABLE 3. Number of communication exchanges from server to client (C2S), client to server (S2C) and client to client (C2C) at each round t and across all rounds T.

Method	S2C	C2S	C2C	Total @ round	Total T rounds
FedAvg	CK	CK	0	$2CK$	$2TCK$
FedProx	CK	CK	0	$2CK$	$2TCK$
FedDyn	CK	CK	0	$2CK$	$2TCK$
SCAFFOLD	$2CK$	$2CK$	0	$4CK$	$4TCK$
FedCyclic	1	1	$CK-1$	$CK+1$	$T(CK+1)$
FedSeq $K_{S_i} = K_{S_j}$	CN_S	CN_S	$\left(\frac{K}{N_S}-1\right)CN_S$	$C\,(N_S+K)$	$TC(N_S+K)$
FedSeq $K_{S_i} \neq K_{S_j}$	CN_S	CN_S	$\sum_{S_i\in\mathcal{S}_t} K_{S_i} - CN_S$	$CN_S +$ $+\sum_{S_i\in\mathcal{S}_t} K_{S_i}$	$TCN_S +$ $+\sum_{t\in[T]}\sum_{S_i\in\mathcal{S}_t} K_{S_i}$

i.e., $K_{S_i} = K_{S_j} = {}^{K}/_{N_S} =: K_S \, \forall i \neq j$. In FedAvg, the server sends the global model to the $C \cdot K$ selected clients, which then send back the updated version. As summarized in Table 3, this process accounts for $2C \cdot K$ exchanges over the network. The same goes for FedProx and FedDyn. SCAFFOLD requires double the communication. In FedSeq with equal K_S instead, the server-to-client (S2C) and client-to-server (C2S) communication only happens between the first and last clients of the chain of each superclient respectively. Since the server selects $C \cdot N_S$ superclients, the process sums up to $2C \cdot N_S$ exchanges. Moreover, within each superclient, the clients exchange messages following the chain, for a total of $K_S - 1$ transmissions $\forall S$. If we consider all $C \cdot N_S$ groups involved, this is equivalent to

$$(K_S - 1)\, C \cdot N_S = \left(\frac{K}{N_S} - 1\right) C \cdot N_S = C \cdot K - C \cdot N_S. \quad (7)$$

By summing everything up, the total is $2C \cdot N_S + C \cdot K - C \cdot N_S = C(N_S + K)$. Since $N_S < K$, **the overall communication cost of FedSeq is smaller than FedAvg, FedProx, FedDyn and SCAFFOLD**. If superclients are not equally sized, the client-to-client (C2C) cost is $\sum_{S_i \in \mathcal{S}_t} (K_{S_i} - 1) = \sum_{S_i \in \mathcal{S}_t} K_{S_i} - C \cdot N_S$, where $|\mathcal{S}_t| = C \cdot N_S$, and the total becomes $C \cdot N_S + \sum_{S_i \in \mathcal{S}_t} K_{S_i}$, *i.e.*, depends on the size of the selected superclients. However, to ensure a fair comparison, we select $K_{S,max}$ s.t. ${}^{K}/_{K_{S,max}} \approx N_S$, *i.e.*, most of the superclients are of the same size, falling back to the first scenario. This implies that **FedSeq always has the lowest cumulative cost**. Lastly, FedCyclic is a limit case of FedSeq with one superclient made of $C \cdot K$ clients, with a total cost of $T \cdot C(K + {}^{1}/_{C})$.

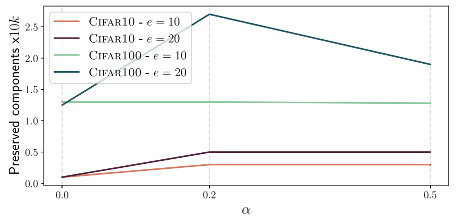


FIGURE 8. CIFAR datasets. Ratio of the preserved components after applying PCA with 90% of explained variance when varying the number of local epochs e.

C. ABLATION STUDIES AND ANALYSES

This section discusses the impact of each method component introduced in Sec. III.

1) ESTIMATING CLIENTS' DATA DISTRIBUTION

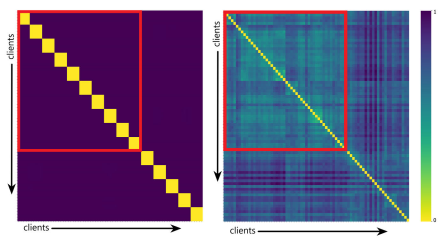
The proposed method utilizes the parameters (ψ_{clf}) or pre-trained model predictions ($\psi_{\text{conf}}, \psi_{\text{t2v}}$) to compute a privacy-preserving estimate of the clients' dataset distribution. To mitigate the *curse of dimensionality* [94] on the classifier parameters in ψ_{clf}, PCA [95] is applied, keeping 90% of the explained variance. Fig. 8 shows that the percentage of preserved components decreases with the complexity of the dataset, *e.g.* fewer components are needed for Cifar10, and increases directly proportional to e. As for ψ_{conf}, not to severely impact the original dataset, $\mathcal{D}_{pub}$ is built using 10 images per class from the test set for computing the *confidence vectors* (Eq. 2). Once $\mathcal{D}_{pub}$ has served its purpose, it is not used again.

Since a public dataset capturing the overall global distribution may not be available in realistic settings, this paper introduces ψ_{t2v}, based on Task2Vec [29], which presents two main advantages: *i*) no external dataset is required, and *ii*) clients only fine-tune the classifier, reducing the latency. Following [29], we use a pre-trained ResNet18 for image classification tasks, and a GPT-2-like [96] language modeling transformer for NLP tasks. To better understand the difference in their behavior, the embeddings of ψ_{t2v} (*right*) and ψ_{conf} (*left*) are compared in Fig. 9a. Specifically, we illustrate the distance between their embeddings computed over the first 75 clients of Cifar100 with $\alpha = 0$. The first 25 clients exclusively have images of aquatic mammals (beavers, dolphins, otters, seals, and whales), the next 25 clients have

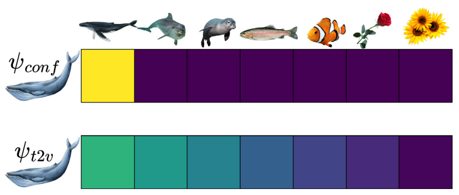
TABLE 4. FedSeq baselines: comparison of grouping criteria by varying ϕ, ψ and τ. Results in terms of accuracy (%).

Method	ψ	ϕ	τ	CIFAR10			CIFAR100			FEMNIST		SHAKESPEARE	
				$\alpha=0$	$\alpha=0.2$	$\alpha=0.5$	$\alpha=0$	$\alpha=0.2$	$\alpha=0.5$	NIID	IID	NIID	IID
FedSeq	-	Random	-	81.90	82.09	82.12	46.39	48.62	49.44	87.07	87.49	51.55	51.84
	clf	K-means	Euclidean	**82.30**	81.78	82.48	44.91	48.74	49.60	87.07	87.45	51.78	51.70
		Greedy	Cosine	79.95	82.06	83.32	45.22	48.92	49.62	87.12	87.42	51.72	51.88
	conf	K-means	Euclidean	82.04	81.99	82.37	43.55	49.43	49.79	87.10	**87.50**	51.79	51.87
		Greedy	KL	82.21	82.20	82.22	45.97	49.56	49.82	87.01	87.46	51.70	51.65
		Greedy	Gini Index	82.09	81.85	82.71	45.79	48.98	49.61	87.05	87.42	51.59	**51.98**
	t2v	Greedy	Norm-Cosine	82.28	82.48	**82.86**	**46.51**	**50.06**	**50.31**	87.11	87.48	51.65	51.82
		ICG	Euclidean	82.05	**82.51**	82.54	46.33	49.13	49.86	**87.18**	87.45	**51.84**	51.77
FedAsyncSeq	*t2v*	*	*	83.40	83.37	83.85	50.38	51.64	51.58	86.96	87.20	51.93	52.02
FedSeq2Par	*t2v*	ICG	Euclidean	$\underline{\mathbf{83.86}}$	$\underline{\mathbf{84.66}}$	$\underline{\mathbf{84.35}}$	$\underline{\mathbf{51.23}}$	$\underline{\mathbf{51.41}}$	$\underline{\mathbf{51.78}}$	$\underline{\mathbf{87.58}}$	$\underline{\mathbf{87.96}}$	$\underline{\mathbf{52.84}}$	$\underline{\mathbf{52.54}}$

(*): (Greedy, Norm-Cosine) for CIFAR10/100; (K-means, Euclidean) for FEMNIST and SHAKESPEARE.



(a) Distance matrices on ψ_{conf} (*left*) and ψ_{t2v} (*right*) tasks embeddings.



(b) ψ_{conf} and ψ_{t2v}'s tasks embeddings similarity.

FIGURE 9. Cifar100, $\alpha = 0$. ***(a)*** **Focus on 75 clients. Each group of 25 clients has access to either images of aquatic mammals, fishes or flowers.** ***(b)*** **Focus on client with images of** ***whales*****. Comparison of embedding distances with clients containing images of progressively different entities. ψ_{t2v} accurately recognizes the similarities between animals, in contrast to ψ_{conf}.**

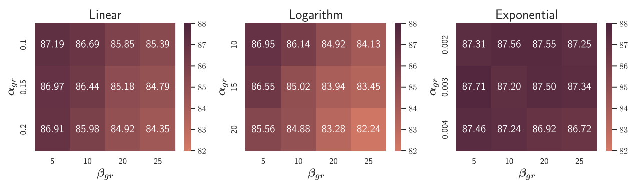
images of fishes (aquarium fishes, flatfishes, rays, sharks, and trouts), and the last 25 clients have images of various flowers. Vectors from ψ_{conf} lack class similarity representation, while the ψ_{t2v} distance matrix reveals that clients with fish and aquatic mammal images (*red square*) cluster together more closely than those with flower images. In Fig. 9b, one client with only whale images is compared in terms of distance with other clients having progressively dissimilar images to whales. Once again, ψ_{t2v} accurately recognizes the similarities between animals, in contrast to ψ_{conf}. To understand this behavior, we note that the embedding of the k-th client with samples belonging to class $c \in [N_c]$ given by ψ_{conf} is $p_k[i] \approx \begin{cases} 1 & if i = c, \\ 0 & otherwise \end{cases}$. This aligns with our expectations, as $f_{\theta_0^k}$ is trained to classify observations with label c. As a result, the embeddings of clients seeing different classes (regardless of the similarity of the depicted subjects) are equally distant. However, this contradicts our intuitive understanding, as we would expect that similarities in data distributions would manifest as closeness in the vector space. In contrast, the distance between ψ_{t2v} embeddings aligns with our intuition on semantic and taxonomic relations among entities. This behavior is evidently reflected in its performance in Table 4, where ψ_{t2v} consistently outperforms the previous best approach, ψ_{conf} [28], for both vision and NLP tasks.

2) GROUPING CRITERIA

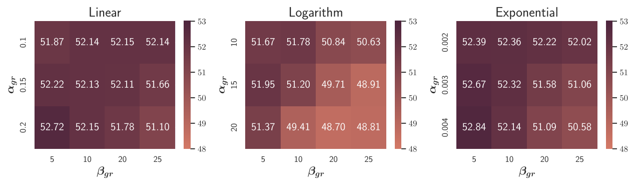
Table 4 compares the different combinations of grouping criteria G_S. As for ψ_{kmeans}, a reasonable value for the number of clusters is N_c, and the Euclidean distance is used to compare the resulting superclients. The confidence vectors extracted by ψ_{conf} have the form of a probability distribution (Sec. III), additionally comparable via disomogeneity measures such as the KL divergence and the Gini Index. The normalized embeddings obtained with Task2Vec are compared with the cosine distance ("Norm-Cosine" in the Table) [29]. Notably, ϕ_{rand} returns groups obtaining competitive results with the other grouping methods. The reason lies in statistical considerations on the cross-device setting: with the number of clients being large in all datasets, a randomly created group is unlikely to contain clients belonging all to the same data distribution. ψ_{t2v} achieves the best performance across all settings, demonstrating effective capture of task similarities. We select ϕ_{icg} as grouping method due to its satisfying results and efficiency, useful in the dynamic creation of superclients especially with several groups.

3) FedSeq2Par

As described in Sec. III-C3, the Sequential-to-Parallel approach used in FedSeq2Par is based on the function $f_{gr}(\alpha_{gr}, \beta_{gr}, t)$ (Eq. 4-6), that defines the number of superclients at each round t. This section aims to understand which growth function better suits the analyzed settings (*linear*, *logarithmic*, or *exponential*) and the effect of the parameters α_{gr} (growth rate) and β_{gr} (initial number of superclients). α_{gr} is chosen so that a fully parallel scenario is reached in the last rounds, while favoring sequential training at the beginning. We test $\beta_{gr} \in \{5, 10, 20, 25\}$ for all datasets except for the larger StackOverflow, for which we select $\beta_{gr} \in \{50, 100, 200, 250\}$. Fig. 10 analyzes the impact of these parameters on the NIID splits of Femnist and Shakespeare. Notably, starting with the smallest number of superclients β_{gr} consistently yields superior performance,



(a) FEMNIST



(b) SHAKESPEARE

FIGURE 10. Sensitivity of FedSeq2Par to f_{gr} and the growth parameters α_{gr} and β_{gr}. Results in test accuracy (%) on the NIID splits.

TABLE 5. Parallelism and test accuracy: FedSeq *vs* FedSeq2Par.

Dataset	N_S	$\overline{N}_S$	Parallelism	Accuracy ($\alpha = 0$ / NIID)	
	FedSeq	FedSeq2Par	↑	FedSeq	FedSeq2Par
CIFAR10	50	**104**	2.09x	81.89	**83.68**
CIFAR100	50	**113**	2.26x	45.87	**51.46**
FEMNIST	175	**383**	2.19x	87.11	**87.58**
SHAKESPEARE	25	**52**	2.09x	51.70	**52.75**
STACKOVERFLOW	1900	**5966**	3.14x	28.91	**29.79**

as it exploits sequentiality more. f_{exp} has the best and most consistent results. Due to the uniformity of these results, the same configuration is maintained across all datasets. We further compare FedSeq2Par with FedSeq in terms of number of superclients and final performance in Table 5. For FedSeq2Par, we compute the average number of superclients $\overline{N}_S$ across rounds. We note that $\overline{N}_S$ is constantly larger than N_S, implying a more parallelized scenario *on average* with FedSeq2Par w.r.t. FedSeq even if $\beta_{gr} \ll N_S$. This behavior positively reflects on the final performance, confirming the efficacy of the STP approach.

V. PRIVACY

Recent FL literature has highlighted the potential for attackers to reconstruct sensitive information through the clients' updates [36]. Thus, concerns on the potential privacy implications of the *client-to-client* sequential training approach introduced by FedSeq arise. Specifically, this extension poses the question: *does FedSeq's client-to-client sequential training facilitate the retrieve of previous users' personal information by a malicious client?* To answer, FedSeq is evaluated against two famous attacks, namely the **label flipping** [35] (LFA) and the **GAN recovery** attacks (GRA) [76], and study potential private information leakages. The well-known gradient inversion attack [36] is not considered here, as its assumptions do not align with our approach (*e.g.*, access of the attacker to both initial and updated models, knowledge of private labels). Differently, in this case, clients only receive the updated parameters from the previous user and potentially malicious clients are not aware of other users' private labels.

A. LABEL FLIPPING ATTACK

LFA is an *active* privacy attack aiming at deteriorating the global model performances by switching labels at training time. Here, the focus is on models solving the classification task. To mislead the global model classification ability, the set of malicious clients $\mathcal{A} := \{a_i\}_{i=1}^{L \cdot K} \subseteq C$ with $L \in [0, 1]$ willingly swaps the labels of their local data following a set of criteria $\{\gamma_i\}_{i=1}^{L \cdot K}$. The criterion γ_i defines the labels to be swapped during the attack for each attacker a_i. For instance, $\gamma_i = \gamma_j$ implies that the attackers a_i and a_j will swap the same classes. This work tests two possible situations:

1) Different attackers swap distinct classes, *i.e.* each attacker a_i chooses its γ_i independently (γ_{random}),
2) All the attackers swap the same classes, *i.e.* $\gamma_i = \gamma_j \, \forall i, j \in \mathcal{A}$ (γ_{fixed}).

B. GAN RECOVERY ATTACK

GRA is a *passive* privacy attack that aims at reconstructing other clients' private information using GAN architectures [97]. It is important to highlight that the primary objective of GANs is to *generate* samples that closely resemble those found in the training set without direct access to the original ones. GANs rely on interactions with a *discriminative* deep neural network to learn and capture the underlying data distribution [76]. They are trained to mimic the images encountered by the discriminative network, starting from random initialization. However, a potential concern arises when the discriminator is trained on private data, as it can potentially be exploited to train a generator network capable of reconstructing the sensitive data. This poses significant privacy and security concerns. Formally, the GANs' optimization problem [76] is

$$\min_{\theta_G} \max_{\theta_D} \sum_{i=1}^{n} \log f(x_i, \theta_D) + \sum_{j=1}^{n} \log(1 - f(g(z_j, \theta_G), \theta_D)), \tag{8}$$

where $f(x, \theta_D) : \mathcal{X} \to \mathcal{Y}$ is a discriminative network parametrized by θ_D that, given an image, outputs a class label. The generative network $g(z, \theta_G) : \mathcal{X} \to \mathcal{X}$ receives random noise as input and outputs an image. x_i is the original image and $g(z_j)$ is a randomly generated one. In GRA, at round t, an attacker $a_i \in C$ disguised as a client exploits the incoming trained model θ_t as the discriminator of a *GAN*, *i.e.* $\theta_D \leftarrow \theta_t$. The generator g is then trained for E_a epochs to reconstruct inputs similar to the ones previously accessed by θ_t, thus breaking the clients' privacy.

C. TESTING FEDSEQ PRIVACY RESILIENCE

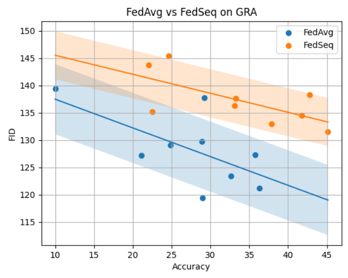
This section provides quantitative results of FedSeq's resilience against the *LFA* and *GRA* attacks. We show that FedSeq not only does not introduce additional privacy liabilities w.r.t. FedAvg, but it learns more robust models.

TABLE 6. Label Flipping Attack experiments after $1k$ rounds. Results in accuracy (%) and drop in accuracy (↓) w.r.t. to the reference. In bold smaller drops in each attack. Symbols: "∘" (negligible or non-existing drops), "Fixed" (γ_{fixed}) and "Random" (γ_{random}).

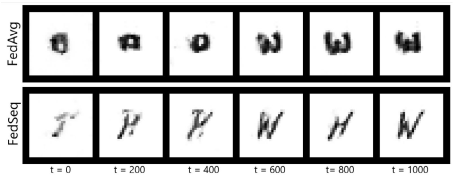
L_S	L	CIFAR10 Swapped Labels	CIFAR10 γ	CIFAR10 FedSeq Accuracy ↑	CIFAR10 FedSeq Drop ↓	CIFAR10 FedAvg Accuracy ↑	CIFAR10 FedAvg Drop ↓	CIFAR100 Swapped Labels	CIFAR100 γ	CIFAR100 FedSeq Accuracy ↑	CIFAR100 FedSeq Drop ↓	CIFAR100 FedAvg Accuracy ↑	CIFAR100 FedAvg Drop ↓
0.1	0.1	0↔2	Fixed	76.06	∘	48.72	∘	*INTRA*$_{SC}$	Fixed	38.81	−0.89	12.57	**−0.28**
	0.5			75.92	**−0.20**	47.55	−1.14			38.87	−0.83	12.35	**−0.50**
0.3	0.1			76.25	**−0.15**	47.74	−0.95			39.66	∘	12.45	−0.40
	0.5			74.75	−1.35	48.30	**−0.39**			39.04	**−0.66**	11.55	−1.30
0.5	0.1			75.65	**−0.45**	47.55	−1.14			39.23	**−0.47**	12.35	−0.50
	0.5			75.50	**−0.60**	47.44	−1.25			38.40	**−1.30**	11.02	−1.83
0.1	0.1	3↔5	Fixed	76.05	∘	47.92	−0.77		Random	39.36	−0.34	13.23	∘
	0.5			75.50	**−0.60**	47.93	−0.76			39.59	∘	13.05	∘
0.3	0.1			75.75	**−0.35**	47.80	−0.89			39.67	∘	13.23	∘
	0.5			75.28	**−0.82**	47.74	−0.95			38.88	−0.82	12.47	**−0.38**
0.5	0.1			76.15	∘	47.93	−0.76			39.68	∘	13.05	∘
	0.5			75.09	**−1.01**	46.94	−1.75			38.98	**−0.72**	11.27	−1.58
0.1	0.1	0↔2 ↔3↔5	Fixed	76.65	∘	47.36	−1.33	*EXTRA*$_{SC}$	Fixed	40.11	∘	12.37	−0.48
	0.5			76.01	∘	48.12	−0.58			40.03	∘	12.20	−0.65
0.3	0.1			75.79	**−0.31**	48.29	−0.40			39.29	−0.41	13.08	∘
	0.5			75.19	**−0.91**	46.42	−2.27			38.73	−0.97	12.34	**−0.51**
0.5	0.1			75.35	−0.75	48.12	**−0.58**			38.95	−0.75	12.20	**−0.65**
	0.5			75.09	**−1.01**	46.94	−1.75			38.46	−1.24	12.30	**−0.55**
0.1	0.1		Random	76.52	∘	48.65	∘		Random	39.17	−0.53	13.12	∘
	0.5			75.71	**−0.39**	47.52	−1.17			39.12	−0.58	12.83	∘
0.3	0.1			75.98	**−0.12**	47.01	−1.68			39.54	∘	13.06	∘
	0.5			74.75	**−1.35**	46.70	−1.99			37.90	−1.80	11.81	**−1.04**
0.5	0.1			75.93	**−0.17**	47.52	−1.17			39.03	−0.67	13.12	∘
	0.5			73.46	−2.64	46.49	**−2.20**			37.76	−1.94	11.94	**−0.91**
		Reference accuracy: **FedSeq** 76.10% - **FedAvg** 48.69%						Reference accuracy: **FedSeq** 39.70% - **FedAvg** 12.85%					

1) FedSeq AGAINST LFA

Table 6 summarizes the results on the different setups proposed to evaluate the robustness of FedSeq to the LFA attack. We distinguish between the fraction of malicious *superclients* L_S and the fraction of malicious clients *within* each malevolent superclient L. The corresponding fraction of attackers in FedAvg becomes $L_S \cdot L$. We test L_S in $\{0.1, 0.3, 0.5\}$ and L in $\{0.1, 0.5\}$. For example, $L_S = 0.1$ and $L = 0.5$ implies that 10% of the superclients are malevolent, and 50% of their clients are attackers. Following [35], the four swapped classes in Cifar10 are: *airplanes* (label `0`) exchanged with *birds* (label `2`), and *dogs* (label `5`) with *cats* (label `3`). We additionally swap all the aforementioned classes (`0`,`2`,`3`,`5`) at the same time. When using Cifar100 instead, the concept of "superclass" proper of the dataset is exploited (*e.g.*, *aquatic mammals*, *flowers*). We either swap 20 classes that do not belong to the same superclass, *i.e.* one class for each superclass (*e.g.*, *dolphins* and *roses*), or exchange pair of 20 labels belonging to the same superclass (*e.g.*, *dolphins* with *whales*). We refer to the former as Extra$_{SC}$, and to the latter as Intra$_{SC}$. To evaluate FedSeq against the easiest scenario for the attacker, all the experiments are run with $\alpha = 100$ on both Cifar10 and Cifar100, meaning that all K clients see all the classes, and the LFA is always feasible. T is set equal to $1k$. Table 6 shows the results of the attack on each proposed configuration, analyzing both the accuracy of the model on the overall test set and the drop w.r.t. the reference experiment without attackers. On average, the *fixed* attacks are more effective than the *random* ones. The reason behind this behavior is intuitive: when using γ_{fixed}, the attackers never let the model learn the correct patterns for classifying the swapped labels, differently from the random acting. Swapping 4 labels in Cifar10 rather than 2 brings on average more damage. For Cifar100, the Extra$_{SC}$ attacks are significantly more effective: the model likely learns some common features for images belonging to the same superclass, leading to a reduced efficacy of the Intra$_{SC}$ attack. Importantly, FedSeq outperforms FedAvg on most scenarios both in terms of accuracy and drop w.r.t. to the reference: this means FedSeq is still able to achieve faster convergence if under attack, and is more robust than FedAvg.



(a) FID scores after GRA attack on FedSeq and FedAvg



(b) GRA attacker images reconstruction

FIGURE 11. (a) GRA attack on global model with different accuracy. The resulting FID scores on FedSeq are consistently higher, implying a less effective attack. (b) Examples of images reconstructed by the GRA attacker at distinct rounds.

2) FedSeq AGAINST GRA

The *Fréchet inception distance* (FID) [98] assesses the quality of the images created by a generative model. Given a dataset $\mathcal{D}$ and its reconstruction $\hat{\mathcal{D}}$, the FID measures the distribution of their features, extracted using an InceptionV3 network [99], using the *Fréchet distance* [98]. A lower score indicates better-quality images. Within the context of an attack, the FID has to be as large as possible, signifying the attacker's inability to reconstruct private data effectively. Unlike the approach in [76], we refrain from incorporating a "fake" class in the classifier, deeming it unrealistic. Instead, we allow the attacker to utilize an additional binary dense layer on top of the model to distinguish between "fake" and "real" data. Fig. 11a resulting from attacks on models with varying levels of accuracy. It is clear that the attack conducted on FedSeq consistently yields higher FID scores in comparison to FedAvg, underscoring its enhanced privacy characteristics. Fig. 11b shows some examples of images reconstruction at different rounds.

VI. CONCLUSION

This work addresses the issues arising from the inherent statistical heterogeneity in Federated Learning (FL) introducing FedSeq. FedSeq leverages sequential training among groups of heterogeneous clients (*superclients*) to obtain more robust models before the server-side averaging step. Various strategies are proposed to effectively group clients according to their data distribution. To reduce the waiting time due to the latency of the slowest superclients, we develop FedAsyncSeq, which allows asynchronous communication between clients and server. Lastly, to exploit sequentiality and parallelism at their best, FedSeq2Par dynamically changes the number of superclients at each round. The extensive experiments on multiple FL benchmarks prove the efficacy of our approaches, in terms of final performances, convergence speed and privacy resilience. include a deeper analysis of FedSeq's convergence properties. Theoretical and empirical studies on large-scale vision datasets could provide valuable insights. Furthermore, extending FedSeq's application beyond classification tasks would be a promising avenue for future research. Finally, addressing the potential for catastrophic forgetting inside superclients due to client heterogeneity is crucial. Developing techniques to mitigate this issue and preserve knowledge across clients would significantly enhance the effectiveness of FedSeq.

ACKNOWLEDGMENT

This manuscript reflects only the authors' views and opinions, neither the European Union nor European Commission can be considered responsible for them.

(Andrea Silvi, Andrea Rizzardi, and Debora Caldarola contributed equally to this work.)

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, 2017, pp. 1273–1282.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[3] D. Caldarola, B. Caputo, and M. Ciccone, "Improving generalization in federated learning by seeking flat minima," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 654–672.

[4] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018, *arXiv:1811.03604*.

[5] L. Yang, B. Tan, V. W. Zheng, K. Chen, and Q. Yang, "Federated recommendation systems," in *Federated Learning: Privacy and Incentive*. New York, NY, USA: Springer, 2020, pp. 225–239.

[6] T.-M. H. Hsu, H. Qi, and M. Brown, "Federated visual classification with real-world data distribution," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K. New York, NY, USA: Springer, 2020, pp. 76–92.

[7] L. Fantauzzo, E. Fanì, D. Caldarola, A. Tavera, F. Cermelli, M. Ciccone, and B. Caputo, "FedDrive: Generalizing federated learning to semantic segmentation in autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2022, pp. 11504–11511.

[8] D. Shenaj, E. Fanì, M. Toldo, D. Caldarola, A. Tavera, U. Michieli, M. Ciccone, P. Zanuttigh, and B. Caputo, "Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 444–454.

[9] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-IID data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2019.

[10] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, 2020, pp. 429–450.

[11] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," 2018, *arXiv:1806.00582*.

[12] Q. Li, Y. Diao, Q. Chen, and B. He, "Federated learning on non-IID data silos: An experimental study," in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 965–978.

[13] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, Jul. 2020, pp. 5132–5143.

[14] T. Zhou, Z. Lin, J. Zhang, and D. H. K. Tsang, "Understanding and improving model averaging in federated learning on heterogeneous data," 2023, *arXiv:2305.07845*.

[15] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *Proc. Int. Conf. Learn. Represent.*, 2021.

[16] Z. Qu, X. Li, R. Duan, Y. Liu, B. Tang, and Z. Lu, "Generalized federated learning via sharpness aware minimization," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 18250–18280.

[17] Y. Sun, L. Shen, T. Huang, L. Ding, and D. Tao, "FedSpeed: Larger local interval, less communication round, and higher generalization accuracy," in *Proc. Int. Conf. Learn. Represent.*, 2023.

[18] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[19] H. Jamali-Rad, M. Abdizadeh, and A. Singh, "Federated learning with taskonomy for non-IID data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 8719–8729, Nov. 2023.

[20] C. T. Dinh, T. T. Vu, N. H. Tran, M. N. Dao, and H. Zhang, "A new look and convergence rate of federated multitask learning with Laplacian regularization," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–11, Dec. 2022.

[21] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3710–3722, Aug. 2021.

[22] K. Kopparapu and E. Lin, "FedFMC: Sequential efficient federated learning on non-iid data," 2020, *arXiv:2006.10937*.

[23] 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. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–9.

[24] G. Long, M. Xie, T. Shen, T. Zhou, X. Wang, J. Jiang, and C. Zhang, "Multi-center federated learning: Clients clustering for better personalization," 2021, *arXiv:2108.08647*.

[25] D. Caldarola, M. Mancini, F. Galasso, M. Ciccone, E. Rodolà, and B. Caputo, "Cluster-driven graph federated learning over multiple domains," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2743–2752.

[26] Y. Yan, X. Tong, and S. Wang, "Clustered federated learning in heterogeneous environment," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–14, Apr. 2023.

[27] P. Kairouz et al., *Advances and Open Problems in Federated Learning*. Hanover, MA, USA: Now Publishers, 2021.

[28] R. Zaccone, A. Rizzardi, D. Caldarola, M. Ciccone, and B. Caputo, ''Speeding up heterogeneous federated learning with sequentially trained superclients,'' in *Proc. 26th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2022, pp. 3376–3382.

[29] A. Achille, M. Lam, R. Tewari, A. Ravichandran, S. Maji, C. Fowlkes, S. Soatto, and P. Perona, ''Task2Vec: Task embedding for meta-learning,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6429–6438.

[30] S. Zeng, Z. Li, H. Yu, Y. He, Z. Xu, D. Niyato, and H. Yu, ''Heterogeneous federated learning via grouped sequential-to-parallel training,'' in *Proc. Int. Conf. Database Syst. Adv. Appl.* New York, NY, USA: Springer, 2022, pp. 455–471.

[31] L. Lyu, H. Yu, and Q. Yang, ''Threats to federated learning: A survey,'' 2020, *arXiv:2003.02133*.

[32] L. Lyu, H. Yu, X. Ma, C. Chen, L. Sun, J. Zhao, Q. Yang, and P. S. Yu, ''Privacy and robustness in federated learning: Attacks and defenses,'' *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2022.

[33] J. Wang, A. Pal, Q. Yang, K. Kant, K. Zhu, and S. Guo, ''Collaborative machine learning: Schemes, robustness, and privacy,'' *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–18, 2022.

[34] L. Muñoz-González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, ''Towards poisoning of deep learning algorithms with back-gradient optimization,'' in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 27–38.

[35] V. Tolpegin, S. Truex, M. E. Gursoy, and L. Liu, ''Data poisoning attacks against federated learning systems,'' in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2020, pp. 480–501.

[36] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, ''Inverting gradients-how easy is it to break privacy in federated learning?'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 16937–16947.

[37] J. Jeon, K. Lee, S. Oh, and J. Ok, ''Gradient inversion with generative image prior,'' in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 29898–29908.

[38] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, ''Federated learning for healthcare informatics,'' *J. Healthcare Informat. Res.*, vol. 5, no. 1, pp. 1–19, Mar. 2021.

[39] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, ''On the convergence of FedAvg on non-iid data,'' in *Proc. Int. Conf. Learn. Represent.*, 2020.

[40] T.-M. H. Hsu, H. Qi, and M. Brown, ''Measuring the effects of non-identical data distribution for federated visual classification,'' in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, 2019.

[41] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, ''Adaptive federated learning in resource constrained edge computing systems,'' *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[42] A. Khaled, K. Mishchenko, and P. Richtárik, ''First analysis of local GD on heterogeneous data,'' 2019, *arXiv:1909.04715*.

[43] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, ''FedDC: Federated learning with non-IID data via local drift decoupling and correction,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10102–10111.

[44] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, ''Local learning matters: Rethinking data heterogeneity in federated learning,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 8397–8406.

[45] Y. Chen, R. S. Blum, and B. M. Sadler, ''Communication efficient federated learning via ordered ADMM in a fully decentralized setting,'' in *Proc. 56th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2022, pp. 96–100.

[46] Y. Gong, Y. Li, and N. M. Freris, ''FedADMM: A robust federated deep learning framework with adaptivity to system heterogeneity,'' in *Proc. IEEE 38th Int. Conf. Data Eng. (ICDE)*, May 2022, pp. 2575–2587.

[47] H. Wang, S. Marella, and J. Anderson, ''FedADMM: A federated primal-dual algorithm allowing partial participation,'' in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 287–294.

[48] Y. Sun, L. Shen, S. Chen, L. Ding, and D. Tao, ''Dynamic regularized sharpness aware minimization in federated learning: Approaching global consistency and smooth landscape,'' in *Proc. Int. Conf. Mach. Learn.*, 2023, pp. 32991–33013.

[49] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, ''Adaptive federated optimization,'' in *Proc. Int. Conf. Learn. Represent.*, 2021.

[50] Y. Wang, L. Lin, and J. Chen, ''Communication-efficient adaptive federated learning,'' in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 22802–22838.

[51] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, ''Mime: Mimicking centralized stochastic algorithms in federated learning,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2021.

[52] J. Wang, V. Tantia, N. Ballas, and M. Rabbat, ''SlowMo: Improving communication-efficient distributed sgd with slow momentum,'' in *Proc. Int. Conf. Learn. Represent.*, 2020.

[53] E. Ozfatura, K. Ozfatura, and D. Gündüz, ''FedADC: Accelerated federated learning with drift control,'' in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 467–472.

[54] J. Xu, S. Wang, L. Wang, and A. C.-C. Yao, ''FedCM: Federated learning with client-level momentum,'' 2021, *arXiv:2106.10874*.

[55] R. Das, A. Acharya, A. Hashemi, S. Sanghavi, I. S. Dhillon, and U. Topcu, ''Faster non-convex federated learning via global and local momentum,'' in *Proc. Uncertainty Artif. Intell.*, 2022, pp. 496–506.

[56] G. Kim, J. Kim, and B. Han, ''Communication-efficient federated learning with accelerated client gradient,'' 2022, *arXiv:2201.03172*.

[57] D. Caldarola, B. Caputo, and M. Ciccone, ''Window-based model averaging improves generalization in heterogeneous federated learning,'' in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2023, pp. 2263–2271.

[58] D. Li and J. Wang, ''FedMD: Heterogenous federated learning via model distillation,'' in *Proc. Adv. Neural Inf. Process. Syst. Workshop*, 2019.

[59] W. Huang, M. Ye, and B. Du, ''Learn from others and be yourself in heterogeneous federated learning,'' in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10133–10143.

[60] R. Caruana, ''Multitask learning,'' *Mach. Learn.*, vol. 28, no. 1, pp. 41–75, 1997.

[61] A. Fallah, A. Mokhtari, and A. Ozdaglar, ''Personalized federated learning: A meta-learning approach,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2020.

[62] X. Zhou and X. Wang, ''Memory and communication efficient federated kernel k-means,'' *IEEE Trans. Neural Netw. Learn. Syst.*, 2022.

[63] M. Papenberg and G. W. Klau, ''Using anticlustering to partition data sets into equivalent parts,'' *Psychol. Methods*, vol. 26, no. 2, pp. 161–174, Apr. 2021.

[64] A. M. Fayaz, S. M. Neethimani, Y. S. L. Reddy, S. Subramanian, and S. Ravichandran, ''Comparative analysis of anti-clusters formed using various distance metrics and k-medoids algorithm,'' *Int. J. Adv. Sci. Technol.*, vol. 29, no. 6, pp. 7705–7717, 2020.

[65] V. Valev, ''Set partition principles revisited,'' in *Advances in Pattern Recognition*, A. Amin, D. Dori, P. Pudil, and H. Freeman, Eds. Berlin, Germany: Springer, 1998, pp. 875–881.

[66] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, ''BrainTorrent: A peer-to-peer environment for decentralized federated learning,'' 2019, *arXiv:1905.06731*.

[67] C. Hu, J. Jiang, and Z. Wang, ''Decentralized federated learning: A segmented gossip approach,'' 2019, *arXiv:1908.07782*.

[68] S. Jain and K. R. Jerripothula, ''Federated learning for commercial image sources,'' in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2023, pp. 6523–6532.

[69] C. Xie, S. Koyejo, and I. Gupta, ''Asynchronous federated optimization,'' 2019, *arXiv:1903.03934*.

[70] C. Xu, Y. Qu, Y. Xiang, and L. Gao, ''Asynchronous federated learning on heterogeneous devices: A survey,'' *Comput. Sci. Rev.*, vol. 50, Nov. 2023, Art. no. 100595.

[71] Y. Chen, X. Sun, and Y. Jin, ''Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[72] X. Lu, Y. Liao, P. Lio, and P. Hui, ''Privacy-preserving asynchronous federated learning mechanism for edge network computing,'' *IEEE Access*, vol. 8, pp. 48970–48981, 2020.

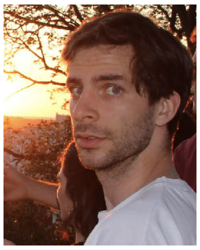
[73] J. Hao, Y. Zhao, and J. Zhang, ''Time efficient federated learning with semi-asynchronous communication,'' in *Proc. IEEE 26th Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2020, pp. 156–163.

[74] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, ''When the curious abandon honesty: Federated learning is not private,'' in *Proc. IEEE 8th Eur. Symp. Secur. Privacy (EuroS&P)*, Jul. 2023, pp. 175–199.

[75] F. Boenisch, A. Dziedzic, R. Schuster, A. S. Shamsabadi, I. Shumailov, and N. Papernot, ''Reconstructing individual data points in federated learning hardened with differential privacy and secure aggregation,'' 2023, *arXiv:2301.04017*.

[76] B. Hitaj, G. Ateniese, and F. Perez-Cruz, ''Deep models under the GAN: Information leakage from collaborative deep learning,'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 603–618.

[77] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014.
[78] Z. Li, J. Zhang, L. Liu, and J. Liu, "Auditing privacy defenses in federated learning via generative gradient leakage," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 10122–10132.
[79] L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov, "Exploiting unintended feature leakage in collaborative learning," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2019, pp. 691–706.
[80] Y. Wen, J. Geiping, L. Fowl, M. Goldblum, and T. Goldstein, "Fishing for user data in large-batch federated learning via gradient magnification," 2022, *arXiv:2202.00580*.
[81] C. Dwork, "Differential privacy," in *Proc. 33rd Int. Colloq. Automata, Lang., Program.*. Cham, Switzerland: Springer, 2006, pp. 1–12.
[82] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3454–3469, 2020.
[83] N. M. Jebreel, J. Domingo-Ferrer, A. Blanco-Justicia, and D. Sánchez, "Enhanced security and privacy via fragmented federated learning," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–15, 2022.
[84] M. R. Garey and D. S. Johnson, "'Strong' NP-completeness results: Motivation, examples, and implications," *J. ACM*, vol. 25, no. 3, pp. 499–508, Jul. 1978.
[85] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-IID data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 5972–5984.
[86] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
[87] D. Steinley, "K-means clustering: A half-century synthesis," *Brit. J. Math. Stat. Psychol.*, vol. 59, no. 1, pp. 1–34, May 2006.
[88] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.
[89] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, CA, USA, Tech. Rep. TR-2009, Apr. 2009.
[90] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," in *Proc. Workshop Federated Learn. Data Privacy Confidentiality*, 2019.
[91] *Tensorflow Federated Stack Overflow Dataset*, TensorFlow, Mountain View, CA, USA, 2019.
[92] F. Varno, M. Saghayi, L. R. Sevyeri, S. Gupta, S. Matwin, and M. Havaei, "AdaBest: Minimizing client drift in federated learning via adaptive bias estimation," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2022, pp. 710–726.
[93] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023.
[94] R. Bellman, "Dynamic programming," *Science*, vol. 153, nos. 37–31, pp. 34–37, 1966.
[95] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *London, Edinburgh, Dublin Phil. Mag. J. Sci.*, vol. 2, no. 11, pp. 559–572, Nov. 1901.
[96] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, Feb. 2019.
[97] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.
[98] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017.
[99] X. Xia, C. Xu, and B. Nan, "Inception-v3 for flower classification," in *Proc. 2nd Int. Conf. Image, Vis. Comput. (ICIVC)*, Jun. 2017, pp. 783–787.



ANDREA SILVI received the bachelor's degree in computer engineering from Politecnico di Torino, in October 2019, and the master's degree in data science and engineering, in December 2022. He is currently pursuing the Ph.D. degree with the Chalmers University of Technology, Göteborg, Sweden, supervised by Prof. Moa Johansson. He joined the Visual and Applied Learning (VANDAL) Laboratory to work on federated learning during the master's thesis, under the supervision of Prof. Barbara Caputo, Dr. Marco Ciccone, and Debora Caldarola. His current research interests include emergent communications in multi-agent systems and neuro-symbolic AI.

ANDREA RIZZARDI received the bachelor's degree, in 2018, and the master's degree in data science and engineering from the Polytechnic of Turin, in 2022. His research interests include statistics and coding, federated learning, and tiny ML.

DEBORA CALDAROLA received the bachelor's and master's degrees in computer engineering from the Polytechnic of Turin, Italy, in 2018 and 2020, respectively, where she is currently pursuing the Ph.D. degree supervised by Barbara Caputo and co-advised by Marco Ciccone. She is also visiting Stanford University advised by Sanmi Koyejo. Her research interests include trustworthy machine learning, with a specific interests include federated learning and fairness studied through the lens of the loss landscape (e.g., sharpness-aware training). She recently organized the Women in Computer Vision Workshop (WiCV) in conjunction with ICCV and is a member of Eta Kappa Nu and the IEEE Honor Society.

BARBARA CAPUTO received the Ph.D. degree in computer science from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2005. From 2007 to 2013, she was a Senior Researcher with Idiap-EPFL. Then, she moved to Sapienza Rome University, thanks to a MUR professorship and joined Politecnico di Torino, in 2018, where she is currently a Full Professor. She leads the Artificial Intelligence (AI) Hub, Politecnico di Torino. Since 2017, she has been a double affiliation with Italian Institute of Technology (IIT). She is one of the 30 experts who contributed to write Italian strategy on AI and a Coordinator of Italian National Ph.D. on AI and industry 4.0, sponsored by MUR. She is an ERC Laureate and a fellow of ELLIS. Since 2019, she has been serving on the ELLIS Board.

MARCO CICCONE received the Ph.D. degree (cum laude) in computer science and engineering from the Polytechnic of Milan, working on iterative and conditional models for visual representation learning. He is currently an ELLIS Postdoctoral Researcher with the VANDAL Group, Polytechnic of Turin. His research interests include the intersection of meta, continual, and federated learning to scale the training of agents with heterogeneous data and mitigate the effect of catastrophic forgetting and heterogeneity across tasks, domains, and devices.

• • •

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement