Efficient Resource-Constrained Federated Learning Clustering with Local Data Compression on the Edge-to-Cloud Continuum

Cédric Prigent, Melvin Chelli, Alexandru Costan, Alexandru Costan, Loïc Cudennec, René Schubotz, Gabriel Antoniu

Efficient Resource-Constrained Federated Learning Clustering with Local Data Compression on the Edge-to-Cloud Continuum

Cédric Prigent*, Melvin Chelli†, Alexandru Costan*, Loïc Cudennec‡, René Schubotz†, Gabriel Antoniu*
* University of Rennes, Inria, CNRS, IRISA - Rennes, France
† Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Saarland Informatics Campus, Germany
‡ DGA Maîtrise de l'Information, Rennes, France
{cedric.prigent, alexandru.costan, gabriel.antoniu}@irisa.fr
{melvin.chelli, rene.schubotz}@dfki.de, {loic.cudennec}@intradef.gouv.fr

***Abstract*—Federated Learning (FL) has been proposed as a privacy-preserving approach for distributed learning over decentralized resources. While it can be a highly efficient tool for large-scale collaborative training of Machine Learning (ML) models, its efficiency may be strongly impacted by a high variability in data distributions among clients. Clustered FL tackles this problem by grouping clients with similar data distributions and training personalized models. Despite increasing model accuracy for federated peers, existing clustering approaches overlook system and infrastructure constraints leading to sustainability problems for resource-constrained devices.**

This paper introduces a new method for resource-constrained FL clustering. We leverage pre-trained autoencoders to compress client data into low dimensional space and build lightweight embedding vectors used to cluster federated clients. A randomized quantization approach specifically secures the client embedding vectors against data reconstruction. Extensive experiments using a multi-GPU testbed with multiple scenarios introducing concept drift between clients demonstrate the generalitity of our approach to personalized FL. By minimizing the overall system overhead and improving the model convergence, our approach reduces model training cost by up to $1.44\times$ - $4.32\times$ communication, $1.03\times$ - $2.40\times$ training time compared to IFCA and $1.0\times$ - $8.60\times$ communication, $0.87\times$ - $8.40\times$ training time compared to LADD to achieve similar accuracy in the different evaluation scenarios. While each of the baselines encounters performance degradation in at least one of the scenarios, our strategy demonstrates top efficiency in all of them.

***Index Terms*—Federated learning, Clustering, Personalization, Autoencoders**

I. Introduction

Federated Learning [1] (FL) is a decentralized paradigm that enables training of Machine Learning (ML) models in a collaborative fashion across several clients or devices. It trains models locally on individual clients, rather than consolidating large amounts of sensitive data on a single server. This approach addresses concerns about data privacy and security by only sharing model updates instead of raw data, unlike traditional centralized machine learning models. While this strategy proves effective in preserving data privacy, it also faces significant challenges in terms of practicality when deploying such FL systems at large scale, due to the inherent statistical heterogeneity of data among individual clients (*i.e.,* client drift). Specifically, when the data across clients are diverse, *non-independent and non-identically distributed* (non-IID), training a global model that performs well on the data of *all* clients is a challenging task.

Personalized FL (PFL) aims to address these issues by fine-tuning the global model to fit local data distributions of federated clients. To this purpose, previous works have explored the use of meta-learning [2], global and personalized layers [3] and a mixture of local and global models [4]. A particularly relevant approach to enable PFL in distributed environments is to build clusters of clients which have similar data distributions: clients belonging to the same cluster share the same personalized model, adapted from the global one [5]. With this approach, each client benefits from the insights of other clients within the same cluster, by applying collaborative training to a more specialized task. The comparison between standard and clustered FL is illustrated in Figure 1.

FL and PFL systems are typically deployed on large, distributed infrastructures, where the training of the global model takes place on some powerful facility (Cloud or HPC system), while the local, personalized training is typically done at the Edge, *i.e.,* on less powerful computational resources close to the data production sites. This seamless combination of resources from the center to the edge, also referred to as the Computing Continuum, or the Edge-to-Cloud Continuum [6], adds a new challenge to FL through network and device heterogeneity (*i.e.,* differences in computation capacity, network latency, node volatility etc.). Thus, we are faced with the challenge of efficiently dealing with a large number of heterogeneous, potentially malicious and highly resource-constrained devices joining the FL process. On the one hand, we aim to achieve an accuracy close to the one achieved by centralized models; on the other hand, we aim to achieve high performance, scalability, security and low resource utilization when deploying such FL models on the Computing Continuum.

Addressing this trade-off is challenging for several reasons: (1) traditional centralized clustering approaches may not be

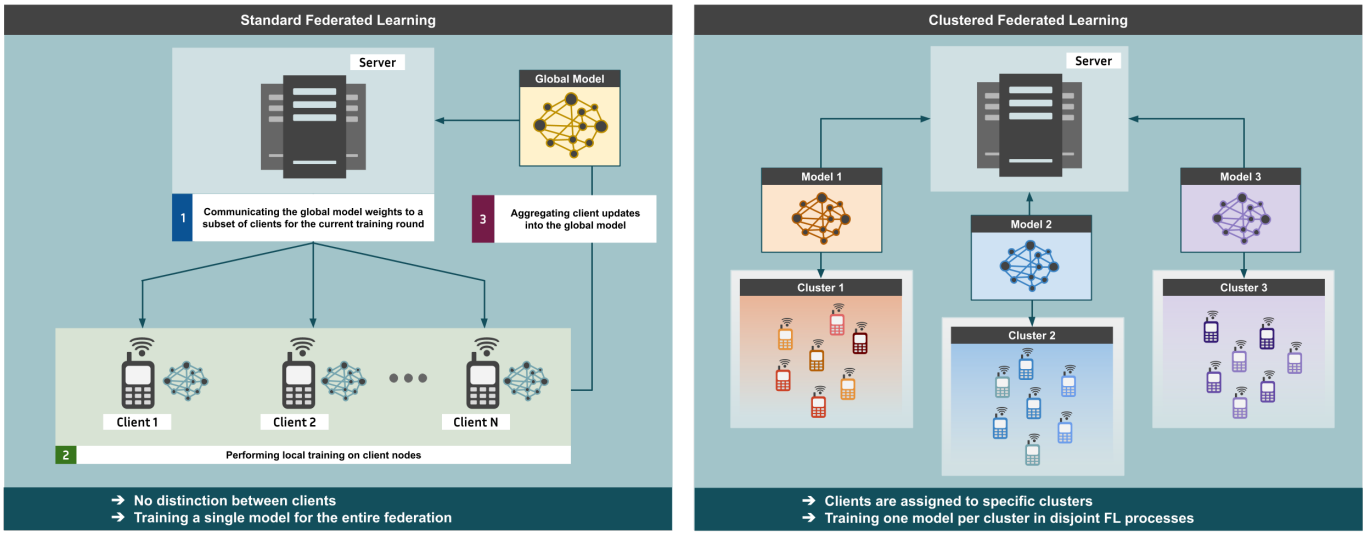


Fig. 1: **FL and clustered FL**: *while standard FL makes no distinction between clients and trains a single model, clustered FL aims to intelligently group clients to improve the training phase and provide clients with personalized models better suited to their local data distributions.*

applicable in decentralized environments [7], [8]; (2) the distributed clustering techniques typically require a large number of communication rounds to learn the distribution similarities and to form clusters; (3) the protocols need to be robust against intruder or server-side data reconstruction attacks. Previous PFL strategies have mainly focused on enhancing the ML process (by improving the training and the clustering accuracy [5], [9]), while overlooking their computing and communication overheads. Moreover, a growing body of existing approaches require all devices in the federation to take part in the training rounds, further exacerbating the overheads in large-scale settings. We focus here precisely on **reducing these overheads when deploying PFL models across the Computing Continuum**.

Despite increasing convergence between AI and High-Performance Computing [10], which has led to the adoption of various parallelization techniques [11] and hierarchical FL [12], the training of FL models remains a time-consuming and resource-intensive task. Indeed, the amount of computation used in the largest AI training runs on the Edge-to-Cloud Continuum has doubled every 3.4 months since 2012 [13]. Under such circumstances, it becomes important **to enable scalable, secure** and **resource-efficient PFL** considering **both the device heterogeneity** and **the training accuracy**.

To mitigate the above challenges, we propose a novel, resource-constrained clustering approach for PFL. We leverage pre-trained autoencoders (AEs) [14] to project client data in a low dimensional latent space. While providing compressed representations of client data, AEs also preserve their meaningful characteristics, which is essential to detect similarities between different data points. Before sending these compressed representations (*i.e.*, embedding vectors) to the server, federated clients perform randomized quantization to prevent data reconstruction risks, thereby preserving privacy. Embedding vectors are finally aggregated by the server in order to perform agglomerative clustering of the federated clients. As client embedding vectors are compressed representations of clients data, their transfer to the server induces a very light communication overhead. In addition, data compression through AEs incurs light computational overhead compared to the actual FL training. This enables our system to cluster federated clients in an efficient manner.

Our key contributions are summarized as follows:

- a set of **design principles** to address system constraints in clustered FL and a **resource-constrained clustering algorithm**; they leverage AEs to project client data in low dimensional space to enable efficient clustering;
- an approach to **automatically set the number of clusters in FL** based on bayesian optimization and agglomerative clustering;
- a **randomized quantization** approach **to mitigate reconstruction and inference attacks** and an experimental study to demonstrate its robustness in these situations;
- an extensive experimental evaluation on a **real distributed testbed (Grid'5000 [15]) using 8 GPUs** showing the ability of the proposed algorithm to adapt in diverse scenarios introducing client heterogeneity through label shifts and concept drifts. Our proposal enables **accurate clustering, improving model convergence** with minimal overhead, resulting in top efficiency compared to state-of-the-art clustering approaches.

The remainder of this paper is organized as follows. Section II introduces relevant background and discusses the limitations of existing work that have motivated our research. Section III introduces the design principles guiding our proposal. Our clustering approach is presented in Section IV and

its implementation is detailed in Section V. Results from the experimental evaluation are reported in Section VI. Section VII discusses the impact of our proposal and Section VIII concludes this paper.

II. Background and Related Work

In this section, we revisit several key FL concepts to set the context of our work. We explain why clustering clients based on their data similarity helps tackle data distribution heterogeneity, and recall dimensionality reduction as a means for efficient, lightweight FL clustering. We explore the corpus of literature that has laid the groundwork for these concepts and discuss their limitations in the context of the Edge-to-Cloud Continuum.

A. Clustered Federated Learning

Clustering approaches for FL group clients into multiple clusters based on their data distributions similarity and train a separate model for each cluster to mitigate the adverse effect of non-IID data. Most of the existing strategies indirectly measure the data distribution similarity among clients by using *model weights, gradient updates, or local loss*.

Gradient similarity has been used to form clusters through recursive bipartitioning [5] as well as greedy agglomerative processes [16]. **IFCA** [9] (used as an evaluation baseline) leverages Empirical Risk Minimization to iteratively assign clients to dynamic clusters. Authors of [17] consider that client data may follow a mixture of data distributions. They propose FedSoft that follows a similar approach to IFCA for cluster assignment, but rather than assigning a single cluster per client, they let clients join several *soft clusters* with different importance weights. In [18], hierarchical clusters are built by iteratively merging clusters based on gradient similarity.

A different approach rely on *integrating generative adversarial networks (GANs) and clustering methods*. ClusterGAN [19] addresses the challenges of maintaining cluster structure in the latent spaces of GANs. Building on this, the authors of [20] use GANs for dynamic client clustering in FL, extending ClusterGAN with cluster calibration mechanisms.

Feature extraction was used in [21]. Authors introduce **LADD** (used as an evaluation baseline) for FL in semantic segmentation, which forms clusters based on image styles. In [22], a drift detection algorithm forms new clusters for drifted clients based on model performance degradation. They use hierarchical clustering to determine the number of models needed to address the drift issue.

B. Dimensionality Reduction

A prominent pre-processing approach used in the past for reducing the overhead of clustering has been dimensionality reduction. Essentially, these methods reduce the complexity of high-dimensional data while retaining their essential characteristics. Various strategies have been explored in this context. Principal Component Analysis (PCA) [7] reduces dimensions by identifying principal components along which data variance is maximized, thereby preserving variability. Similarly, t-SNE [8] is a non-linear technique reducing high dimensional data to lower dimensions mainly to help visualisation of high-dimensional data and cluster analysis. Histogram of oriented Gradients (HOG) and Uniform Manifold Approximation Projection, (UMAP) are other popular feature extraction tools which maintain local and global data structures when reducing dimensions, unlike t-SNE and PCA.

C. Shortcomings

Despite these promising results for both clustering and dimensionality reduction, it remains an open challenge to combine these techniques in order to cluster clients optimally in the FL context.

Popular dimensionality reduction techniques (such as PCA and t-SNE) **are not suited for federated analytics**: PCA, which is very sensitive to inputs, might lead to very different compressions for clients with slightly diverging data distributions; t-SNE cannot run over decentralized data. ML-driven approaches could bring a suitable solution to this problem. Autoencoders (AEs) are neural networks that learn how to project data in low dimensional latent space (encoding) and how to reconstruct them from their compressed representation (decoding). AEs can be used for dimensionality reduction [23] by extracting meaningful features for clustering [24], [25].

Clustering approaches like CFL [5], ClusterGAN [19] and FLACC [16] rely on iterative methods whose success **requires the participation of many (*i.e.*, most of the) clients** in the FL process, demanding a significant amount of time for convergence, which becomes a problem in resource constrained environments. Adding clients dynamically could potentially worsen this behaviour. Additionally, some approaches like LADD [21] (which uses image style to perform clustering) **lack generality**, as the nature of the client drift might negatively impact the accuracy of the clustering.

Compared to these works, we bring a new contribution in the clustering domain for FL through the use of Autoencoders (AEs) to generate lightweight embedding vectors that capture important characteristics of FL clients. Our approach enables clustering *before* the training phase (*i.e.*, avoiding iterative and expansive processes for federated clients) in an efficient manner through the use of lightweight AEs.

D. Problem Statement

Let us recall that straightforward deployment of existing FL clustering approaches on real-life, large-scale infrastructures across the Edge-to-Cloud continuum is not feasible. Prior works have mainly focused on improving the model accuracy, overlooking the impact on performance [26]. Typically, clustering approaches rely on iterative processes that either require the clients to download available models of all clusters at each round and select one that provides the highest test accuracy, or require the entire federation to perform local training steps harvesting gradient information. Such approaches incur large communication costs for two reasons: (1) the continuous communication between the server and every client to form

clusters, and (2) the large numbers of communication rounds needed to form stable clusters.

Our goal is to devise scalable, lightweight FL clustering techniques with practicality as a core focus, making them applicable to real-world scenarios on the Edge-to-Cloud continuum. The main research question we aim to answer is **how to cluster clients efficiently** (*i.e.,* with minimum computation and communication overhead) and **accurately** (*i.e.,* the obtained clusters maximize the overall model accuracy). To the best of our knowledge we are the first to address these issues *simultaneously*.

III. Design Principles

To mitigate the limitations identified in Section II-C, we define a set of design principles for our clustering strategy.

One-shot clustering. A major limitation of iterative clustering approaches is that they may require many rounds to converge, resulting in inefficient training and waste of resources. In contrast, one-shot clustering brings the advantage of not adding any system overhead after the initial clustering phase. Thus, we leverage a single clustering phase in which we assign all participating clients to a cluster before starting the actual training.

Leverage feature extraction for clustering. Some clustering approaches rely on gradient similarity between clients which requires each client to perform a training round before the clustering phase. This raises a performance problem with massive crowds of constrained devices, all performing local updates of the ML model. Instead, we leverage privacy-preserving feature extraction from the local client datasets.

Lightweight communication and computation. Devices encountered at the Edge typically have constrained computing resources and limited network access. For some of them, handling the FL training may already lead to a substantial overhead. Our clustering strategy should minimize the communication and computation overheads for federated clients.

Provide adequate support for new clients dynamically joining the FL training. FL is typically subject to high volatility, as a massive number of devices may join or leave the federation at any time during training. This phenomenon is further exacerbated at the Edge of the continuum. Although the training strategies may yield effective models for clients actively participating in the training process, they cannot ensure comparable performance for new clients who have just joined the federation. Current solutions primarily focus on creating accurate personalized models for existing clients, often neglecting the challenges related to new clients [26]. Therefore, it is important to provide a mechanism for new clients to easily identify the most suitable cluster with respect to their local data distributions.

IV. Contribution: Resource-Constrained Clustering for Federated Learning

The purpose of clustering in FL is to group clients with similar data distributions together in order to provide a better training of ML models. One problem arising from the decentralized nature of FL is that the data distributions of individual clients are not known by the central server. This limitation hinders the ability to achieve optimal clustering.

To bridge this gap, we propose to leverage feature extraction techniques to retrieve important properties of client local datasets. We propose a new FL clustering method using autoencoders (AEs) to generate embedding vectors of client local data. First, we describe our feature extraction strategy to build client embedding vectors in a privacy-preserving manner, next, we present our FL clustering strategy which groups clients based on their embedding vectors and automatically sets the number of clusters to form.

A. Embedding Vectors with Local Feature Extraction

We describe our feature extraction approach in Algorithm 1.

(1) **Client embedding vector generation.** The first step for the federated clients is to compress their local data by using a pre-trained encoder network to build *embedding vectors*. Embedding vectors retain essential characteristics of local client data in a restricted space. To build their embedding vectors, clients generate an encoded representation for each of their data samples using the embedding model (*i.e.*, AE encoder) and compute the average representation for each class (Alg 1, line 5). We make a distinction between classes to efficiently detect possible conceptual drift and label shifts between data distributions (Alg 1, line 4). The client embedding vectors are finally built by concatenating the representation of each class (Alg 1, line 6).

Algorithm 1: Client embedding vector generation

1 **Input:** N is the total number of classes.
2 Q is the randomized quantization probability.
3 **Function** `LocalCompression(`$\mathcal{P}_k$`)`**:**
4 $\quad \mathcal{B}_{1\leq i\leq N} \leftarrow$ (split local dataset $\mathcal{P}_k$ by class)
5 $\quad \mathcal{V}_{1\leq i\leq N} \leftarrow \texttt{avg}(\texttt{encode}(\mathcal{B}_{1\leq i\leq N})$
6 $\quad \mathcal{Z} \leftarrow \texttt{concat}(\mathcal{V}_{1\leq i\leq N})$
7 $\quad \mathcal{Z}_q \leftarrow \texttt{randomizedQuantization}(\mathcal{Z}, Q)$ (Eq. (1))
8 $\quad$ **return** $\mathcal{Z}_q$

(2) **Mitigating reconstruction risks with randomized quantization.** Sharing low dimensional embeddings with a central server could lead to privacy breaches. In particular, client embedding vectors could be used to reconstruct or infer properties of the original data. To avoid such privacy problems, one solution is to reduce the sensitivity of the query that generates the embedding vector. Previous works have been exploring the use of additive noise [27] as well as randomized quantization [28], [29] techniques to achieve local differential privacy guarantees and improve privacy in distributed networks.

We decide to use randomized quantization with a strong quantization (*i.e.*, binary quantization) on the client embedding vectors for two reasons: (1) it strongly reduces the sensitivity of the client embedding vector, thus improving privacy; (2) it also reduces the communication overhead by discretizing the

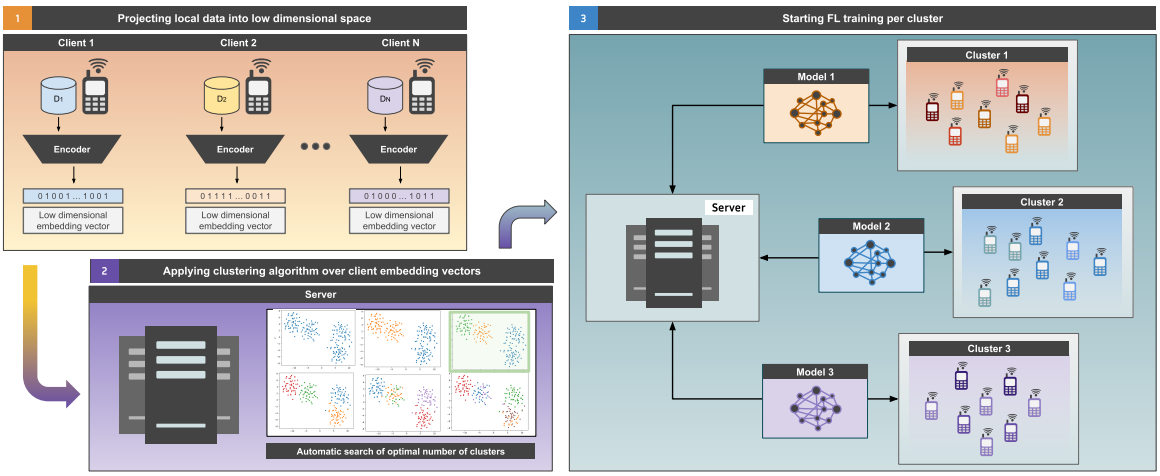


Fig. 2: **FL clustering using client embedding vectors**: *(i) each client generates a low dimensional embedding vector retaining essential characteristics of its local data. (ii) Client embedding vectors are aggregated at the server to perform clustering of federated clients, and (iii) enable separate training of personalized models.*

embedding vectors before sending them to the server. To apply our binary randomized quantization scheme, we normalize the values of the embedding vectors between 0 and 1 and discretize values of the embedding vectors in binary space using randomized rounding as proposed in [28]:

$$z = \begin{cases} round(z) & \text{with probability } 1-P \\ flip(round(z)) & \text{otherwise} \end{cases} \quad (1)$$

where P is the probability of randomly flipping bits of the embedding vector. This process is applied by the clients before communicating their embedding vectors to the server (Alg 1, line 7).

B. Lightweight Clustering with Dimensionality Reduction

We build upon an agglomerative clustering approach to automatically find the number of clusters to form. Agglomerative clustering starts by assigning each client to a singleton cluster and iteratively merges pairs of clusters until a minimum distance threshold separates each cluster.

(3) Distance threshold optimization. The distance threshold impacts the final number of clusters that will be formed by the algorithm and therefore have an impact on clustering quality. Algorithm 2 describes our distance threshold optimization approach using Bayesian optimization. Line 3, a gaussian process regressor model is initialized to predict the performance of the agglomerative clustering based on distance threshold values. A Bayesian optimization process runs for K iterations using $UpperConfidenceBound$ to explore different distance threshold values using the Calinski-Harabasz index [30] (*i.e.*, the ratio of between-cluster separation to the within-cluster dispersion) as the clustering quality metric. The surrogate model is updated in each step based on the clustering score (line 9). Finally, the algorithm returns the optimized distance.

(4) Clustered FL training. Algorithm 3 summarizes the entire FL clustering procedure. Lines 4-7, clients compress their local data and send their embedding vectors $\mathcal{V}_{1\leq i\leq N}$ to

Algorithm 2: Distance threshold optimization

```
1  Input: K is the number of iterations to run BayesOpt.
2  Function DistanceOpt(𝒳):
3      f ← GaussianProcessRegressor()
4      # Optimize f with Bayesian Optimization to find d*
5      foreach iteration k ∈ range(1, K) do
6          d ← arg max(UpperConfidenceBound(f))
7          ℒ ← AgglomerativeClustering(𝒳, d)
8          score ← −CalinskiHarabaszIndex(𝒳, ℒ)
9          f ← UpdateSurrogateModel(d, score)
10     end foreach
11     d* ← arg max(UpperConfidenceBound(f))
12     return d*
```

Algorithm 3: FL clustering from client embeddings

```
1  Input: R is the total number of rounds to run FL, N is the
       total number of clients, m is the number of clients to
       sample per round. 𝒫_i represents the partition of client i.
2  Function Server():
3      # Request embedding vectors from all clients
4      𝒮 ← sample(N)
5      foreach client i ∈ 𝒮 do
6          𝒱_i ← LocalCompression(𝒫_i)                    (Alg 1)
7      end foreach
8      # Run clustering algorithm
9      d* ← DistanceOpt(𝒱_{1≤i≤N})                        (Alg 2)
10     ℒ_{1≤i≤N} ← AgglomerativeClustering(𝒳, d*)
11     K ← len(distinct(ℒ_{1≤i≤N}))
12     # Run FL per cluster
13     w*_{1≤i≤K} ← init()
14     foreach round r ∈ range(1, R) do
15         foreach cluster k ∈ range(1, K) do
16             S_k ← sample(m)
17             w_{1≤i≤|S_k|} ← ClientUpdate(𝒫_{i∈S_k}, w*_k)
18             w*_k ← FedAvg(w_{1≤i≤|S_k|})
19         end foreach
20     end foreach
21     return w*_{1≤i≤K}
```

the server. Line 9-10, the server runs the distance threshold optimization process and runs agglomerative clustering which automatically finds the number of clusters to form. A model for each cluster is initialized (line 13) and a FL process runs for each cluster. In each round and for each cluster, the server samples a subset of clients (line 16), sends them the model for local training (line 17) and aggregates the model using `FedAvg` (line 18). Figure 2 illustrates the entire process.

V. Implementation Details

In this section we discuss some of the choices made to implement our approach. The code is **open-source** and available online[1].

Autoencoder pre-training. The AE used to generate client embedding vectors requires training before the actual clustering phase. The AE could either be pre-trained on a *publicly available dataset* or pre-trained in a *federated manner*. Pre-training with a publicly available dataset brings several benefits: it does not add overhead to the FL training (it can be done using a centralized server) and it does not require extra configuration specific to FL. On the other hand, pre-training in a federated manner might lead to more accurate encoding and clustering, as the AE is directly trained on the actual client data. In this paper, we will consider both cases.

Compensating for missing classes. Missing classes among local client datasets is common in FL. Consequently, the resulting embedding vectors of different clients could have different lengths. This could be a problem with regard to privacy, as the central aggregator would gain knowledge of existing and missing classes among federated clients. To address this problem, for each missing class among client local datasets, a random representation drawn from a uniform distribution is generated instead. This ensures that each client shares the same amount of data. Let us illustrate this process for a use-case with 28x28 input images and an AE with latent space of size 10. To compute the embedding vector, a federated client would generate an embedded representation of size 10 for each of its samples and then compute the average representation for each class. Assuming 10 data classes exist and one is absent from the client local dataset, a random representation is generated for the missing class. The client embedding vector is then constructed by concatenating representations of all classes, resulting in an embedding vector of size 100.

New clients joining during training. Any client joining the FL process should be able to easily identify the most appropriate cluster. With our approach, any client may join the FL process by asynchronously computing its embedding vector and sending it to the server. The server can easily assign the new client to a cluster by computing the distance between its embedding vector and each cluster centroid.

VI. Experimental Evaluation

We evaluate the clustering and system performance of our approach in multiple scenarios introducing concept drift and label shift. Thus, we aim to answer the following questions:

- How robust is our clustering approach based on client embedding vectors against data reconstruction attacks?
- How efficient are the embedding vectors to cluster clients accurately?
- What is the system overhead incurred by this approach?
- What is the impact of the different AE pre-training strategies?

Note that, while the common practice in the FL domain is evaluation through simulation, we run experiments on a 8-GPU distributed testbed instead, in order to assess the benefits of this proposal in real settings. All artifacts are accessible to enable the **reproducibility** of our experiments

A. Methodology

Evaluation scenarios and training datasets. We consider several scenarios introducing *concept drift* and *label shift* between federated peers using different dataset sizes and partition numbers:

- **MNIST** [31]: The dataset is partitioned in a non-IID manner between 100 clients that are uniformly distributed in 4 clusters by applying different rotations over their partition data (*i.e.*, 0, 90, 180 and 270 degrees).
- **CIFAR-10** [32]: The dataset is partitioned in a non-IID manner between 100 clients that are uniformly distributed in 4 clusters by applying different label-flipping to their data. Clients within the same cluster apply the same label-flipping to their data.
- **FEMNIST** [33]: This dataset is a federated version of the EMNIST dataset with handwritten characters assigned to their corresponding authors. Overall, the dataset is partitioned into 3550 authors (*i.e.*, clients) with their own writing styles. Clustering approaches can be used to group authors with similar writing styles in order to improve the learning phase. In this scenario, the optimal clustering is unknown.
- **PACS** [34]: We investigate the clustering performance with data coming from diverse domains. PACS is a dataset that contains data from 4 domains: *photos, arts, cartoons, sketches*. We make 10 partitions from each domain dataset by splitting them in a non-IID manner. Each client is assigned a single partition.

Models. We use a combination of small and larger models in our experiments to assess the impact of model size on different approaches. The trained models are LeNet-5 [35] (60K parameters) in the MNIST and FEMNIST scenarios, and ResNet-18 [36] (11M parameters) in the CIFAR-10 and PACS scenarios.

Hyperparameter details for each scenario are provided in Table I. For MNIST and FEMNIST scenarios, we use an autoencoder composed of 4 fully connected layers with latent space of size 20 (81K parameters). As for the CIFAR-10 and PACS scenarios, we use an architecture with 6 convolutional layers and batch normalization (47K parameters).

AE pre-training For each scenario, we evaluate two AE pre-training strategies: centralized pre-training on a publicly

[1]https://github.com/cedricprigent/efficient-fl-clustering

TABLE I: Evaluation scenarios hyperparameters

Dataset	Total training samples	Input size	Number of clients	Client sampled per round	Average number of samples per client	Local epochs	Model
MNIST	60,000	1x28x28	100	50	600	5	LeNet-5
CIFAR-10	50,000	3x32x32	100	50	500	1	ResNet-18
FEMNIST	805,263	1x28x28	3550	75	226	3	LeNet-5
PACS	7,992	3x64x64	40	20	200	1	ResNet-18

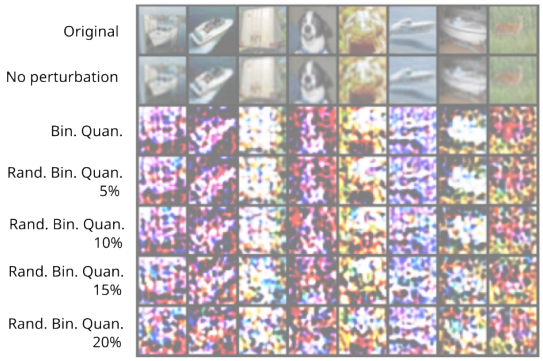
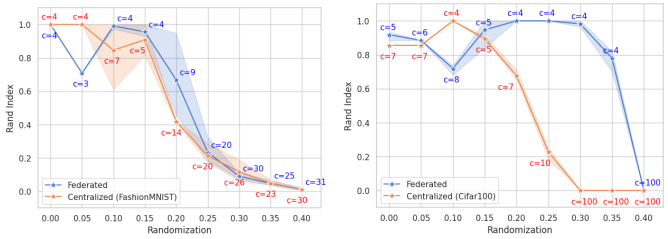


Fig. 3: Data reconstruction under different perturbation levels. Row 1 is the original input image, row 2 is the data reconstruction without perturbation, rows 3-7 are data reconstruction with different levels of randomized binary quantization.



(a) MNIST (b) CIFAR-10

Fig. 4: Clustering quality comparison between federated and centralized pre-training strategies under different levels of randomization. We also precise the number of clusters formed by the algorithm (*e.g.*, c=4).

TABLE II: Reconstruction loss for various perturbation levels.

	No perturb	Randomized Binary Quantization				
		0%	5%	10%	15%	20%
MSE	0.0175	0.7039	0.8040	0.9075	0.9811	1.0765

available dataset and federated pre-training over clients data. Regarding centralized pre-training, we pre-train the AE with Fashion-MNIST for MNIST and FEMNIST scenarios, and CIFAR-100 for CIFAR-10 and PACS scenarios.

Baselines. We compare our clustering approach with several state-of-the-art FL solutions:

- **FedAvg** [1] is the standard strategy for FL with no additional personalization mechanisms. We compare with FedAvg to determine the accuracy improvement provided by clustering. Regarding system performance, we assess the FedAvg communication and computation costs.
- **IFCA** [9] is an iterative and dynamic clustering approach. In each federated round, selected clients download the model from each cluster and estimate their cluster identities based on empirical risk minimization and optimize model parameters of the selected model. In our experiments, to reduce the overhead of IFCA we limit the empirical risk minimization to a single batch of client datasets.
- **LADD** [21] is a one-shot clustering approach using Fast Fourier Transform to compute analytics over client local datasets before performing K-Means clustering.

Performance metrics. In terms of clustering performance, we report the *testing accuracy*, defined as the accuracy achieved by the model on the local datasets of sampled clients at a specific round, as well as the *number of rounds* to achieve a target accuracy.

To measure the overhead, we track *the total system communication* in GB including all communications *from* and *to* the server, and *the total time to run an experiment* (*i.e.*, to run the AE pre-training, the clustering phase, and the FL training).

Computing environment. We run experiments on up to 4 nodes of the *chifflot* cluster of the Grid'5000 [15] distributed testbed. Each node is equipped with 2xNvidia Tesla P100 GPUs (16GB). We configure and manage the evaluation workflow with the E2Clab [37] deployment framework for reproducible experiments. FL adopts a star topology which consists in routing all clients to a central node (*i.e.*, the server). We can capture the total communications of the system by monitoring the network IO at the server node. Therefore, we deploy the server on a single node and deploy all the clients (*i.e.*, up to 3550) on the remaining nodes.

B. Resilience against Data Reconstruction

We evaluate the robustness of our approach against data reconstruction attacks. To measure the risk of data reconstruction from the shared client embedding vectors, we consider the worst case scenario where a single sample is used to produce the embedded representation of a single class from a federated peer. We pre-train an autoencoder in a centralized manner such that it is capable of reconstructing an input image. The pre-trained autoencoder is then used to: (1) encode the input image, and (2) reconstruct the input image from its encoded representation. We evaluate the quality of the reconstruction under different perturbation levels induced by randomized quantization.

We report in Figure 3 the data reconstruction results under different perturbation levels (*i.e.*, randomized quantization) for

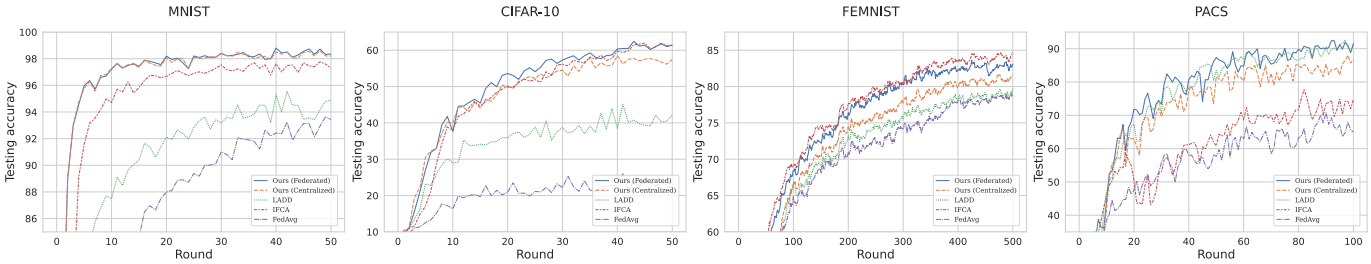


Fig. 5: Testing accuracy achieved by each strategy in different scenarios. Ours (Federated) stands for our approach using the federated AE pre-training policy. Ours (Centralized) stands for our approach using the centralized AE pre-training policy.

TABLE III: Number of rounds to achieve a target accuracy [*Rnd. to X%*] and final testing accuracy [*F. Accuracy*]. To reduce the training noise, the final accuracy is smoothed over the N last training rounds (5 rounds for MNIST, CIFAR-10 and PACS, and 20 rounds for FEMNIST).

	MNIST		**CIFAR-10**		**FEMNIST**		**PACS**	
Strategy	*Rnd. to 95%*	*F. Accuracy*	*Rnd. to 50%*	*F. Accuracy*	*Rnd. to 80%*	*F. Accuracy*	*Rnd. to 80%*	*F. Accuracy*
Ours (federated)	**5**	**98.49**	**17**	**61.06**	**212**	82.94	**31**	**90.65**
Ours (centralized)	**5**	**98.35**	19	59.53	305	81.18	50	86.93
IFCA	11	97.50	20	**61.19**	**212**	**84.16**	Unreached	73.12
LADD	40	94.16	Unreached	40.83	392	79.12	**31**	**90.94**
FedAvg	Unreached	93.05	Unreached	22.93	418	78.62	Unreached	65.16

CIFAR-10 samples. Table. II presents MSE reconstruction loss for each perturbation. We notice that quantizing the embedding vector in a discrete space (third row of Figure 3) impairs the data reconstruction with a loss of details (*e.g.*, colors, shapes), however major shapes may remain present in the image. Adding randomization (rows 4 to 7) helps hindering the reconstruction of main shapes as seen when increasing the randomization level. With 10% randomization and higher, extracting any meaningful knowledge from data reconstructions becomes very difficult. Results from Table II also highlight this trend, with a big spike in reconstruction loss when quantizing the embedding vector (Randomized Binary Quantization 0%) and then linearly increasing with randomization levels.

C. Impact of Randomized Quantization

We perform a hyperparameter study to understand the impact of randomized quantization over the clustering performance of our strategy. We use the Adjusted Rand Index metric to measure the quality of the clustering. Knowing the ground truth clustering labels, Adjusted Rand Index returns a similarity score between -0.5 for discordant clustering and 1 for perfect clustering whereas random labeling should return a score close to 0.0.

We compare the performance of our strategy with two AE pre-training policies:

1) pre-training the AE using FL on the client personal data (*i.e.*, using the same data that is used for training the classification model);
2) pre-training the AE in a centralized manner on a publicly available dataset.

We report in Figure 4 the Adjusted Rand Index obtained for MNIST and CIFAR-10 datasets under different randomization levels. As seen in Figure 4a, both the pre-trained AE using FL and the one pre-trained in a centralized manner on Fashion-MNIST exhibit excellent clustering performance up to a 15% randomization rate. However, beyond this threshold, there is a noticeable decline in performance. In contrast, Figure 4b shows that using FL to pre-train the AE provides better results than the centralized approach. The centralized pre-training achieves decent clustering up to 20% randomization, while the federated approach achieves high Rand Index up until 35%. Note that a lower score do not necessarily mean poor clustering. Rand Index returns a similarity score based on a set of ground truth labels. On the other hand, a clustering algorithm could possibly identify alternative patterns that were not considered initially when synthetically building clusters.

Based on these findings, we set the randomization to 10% to perform our experiments in the remainder of the evaluation. Also, we use the Fashion-MNIST pre-trained AE for MNIST and FEMNIST scenarios, and the CIFAR-100 pre-trained AE for CIFAR-10 and PACS scenarios.

D. Clustering Accuracy

In the next series of experiments we examine the testing accuracy of the clustering approaches over client local datasets, compared to baselines on Grid'5000. Figure 5 reports the accuracy achieved by each strategy over FL training in each evaluation scenario. Table III presents further details about the convergence rate, such as the number of rounds to achieve a target accuracy, and the final testing accuracy achieved by each strategy. We emphasize that regardless of the scenario, the clustering-based approaches always improve accuracy compared to the FedAvg baseline.

In the MNIST scenario, our strategy exhibits the best model convergence, whether using the federated or the centralized version of the autoencoder. It achieves 95% accuracy in 5

TABLE IV: Total system overhead. [*Comm. (GB)*] refers to total communication overhead of a strategy over an entire scenario. [*Time (s)*] refers to total training time including clustering and (federated) pre-training phase.

	MNIST (50 rounds)		**CIFAR-10** (50 rounds)		**FEMNIST** (500 rounds)		**PACS** (100 rounds)	
Strategy	*Comm. (GB)*	*Time (s)*	*Comm. (GB).*	*Time (s)*	*Comm. (GB)*	*Time (s)*	*Comm. (GB)*	*Time (s)*
Ours (federated)	3.2	118	349.3	489	44.2	436	280.5	415
Ours (centralized)	2.7	109	349.0	472	42.7	438	280.1	402
IFCA	5.0	111	700.4	838	66.9	354	561.2	714
LADD	2.7	107	349.0	482	42.7	345	280.1	394
FedAvg	2.7	108	349.0	468	42.7	324	280.1	382

TABLE V: Training efficiency - Resource usage to achieve a target accuracy.

	MNIST (95% accuracy)		**CIFAR-10** (50% accuracy)		**FEMNIST** (80% accuracy)		**PACS** (80% accuracy)	
Strategy	*Comm. (GB)*	*Time (s)*	*Comm. (GB).*	*Time (s)*	*Comm. (GB)*	*Time (s)*	*Comm. (GB)*	*Time (s)*
Ours (federated)	0.75	19	**117.5**	**170**	**19.6**	**146**	**86.6**	139
Ours (centralized)	**0.25**	**10**	131.2	177	26.1	208	139.6	198
IFCA	1.08	24	278.8	335	28.4	150	Unreached	Unreached
LADD	2.15	84	Unreached	Unreached	33.5	254	**86.3**	**122**
FedAvg	Unreached	Unreached	Unreached	Unreached	35.7	271	Unreached	Unreached

rounds and a final accuracy of 98.49%. In contrast, IFCA reaches a slightly lower accuracy with 97.50%. LADD's style extraction technique is ineffective for MNIST digits (which do not differ in styles) resulting in minor improvements compared to FedAvg.

In the CIFAR-10 scenario, which introduces label shifts, both our approach and IFCA provide the top results. They exhibit similar convergence rates (17 - 20 rounds to achieve 50% accuracy) and final accuracy (approximately 61%). With its style extraction technique, LADD partially succeeds in detecting label shifts and achieves 40.83% accuracy, which is halfway between FedAvg accuracy and the top accuracy.

In the FEMNIST scenario, IFCA is the best performing approach with a 84.16% testing accuracy. Our approach with the federated pre-training achieves the 80% target accuracy after 212 rounds which is similar to IFCA. However, it has a slightly lower final accuracy (82.94% for the federated version). The centralized version achieves 81.18% which is a little worse than the federated version. Similar to the MNIST scenario, LADD style extraction is unable to detect drifts in clients data and only brings a slight improvement compared to FedAvg ($+0.5\%$ accuracy).

As for the PACS scenario, our approach and LADD exhibit the best results with similar convergence and final accuracy within [90.65% - 90.94%]. IFCA is unable to find satisfying clusters and only brings limited improvement compared to FedAvg by achieving 73.12% ($+7\%$) accuracy. In this scenario introducing domain drift, the style extraction technique used by LADD effectively distinguishes between the various disparities in data distribution.

Summary. Overall our approach achieves top performance in 3 of the 4 evaluation scenarios (MNIST, CIFAR-10 and PACS) and provides satisfying results for FEMNIST, showing that client embedding vectors can efficiently capture important characteristics of clients data. In contrast, IFCA struggles in the PACS scenario (introducing domain drift between clients), and LADD achieves limited improvements compared to FedAvg in most scenarios. Thus, we conclude that our approach makes a first step towards the generalization of FL clustering with potential applicability to a wide range of domains.

E. System Overhead Breakdown

Finally, we examine the communication and computation overheads of our clustering approach. Table IV reports the total system overhead (*i.e.*, total communication and training time) of each strategy. Regardless of the scenario, both our approach with the centralized pre-trained AE and LADD entail no additional communication overhead compared to FedAvg. Our approach with the federated pre-trained AE requires slightly more computing time and communication bandwidth due to the pre-training done directly by the federated clients.

This overhead is relatively higher in the MNIST and FEMNIST scenarios as the trained classification model (*i.e.*, LeNet-5) is lighter than the one used for CIFAR-10 and PACS scenarios (*i.e.*, ResNet-18), resulting in less computation and communication to run the FL training task while requiring similar resources to run the AE pre-training task. Nevertheless, the overhead of the federated pre-training remains moderate in all scenarios.

We also observe a higher training time overhead ($+35\%$) in the FEMNIST scenario. This is due to the clustering optimization phase (*i.e.*, Bayesian optimization) searching for the best number of clusters dealing with a high number of clients (*i.e.*, 3550). However, the clustering optimization which is performed by the server does not increase the resource usage at the clients.

In contrast, IFCA causes significant communication and computational overhead in all scenarios. It doubles communication overhead for MNIST, CIFAR-10 and PACS scenarios and doubles training time for ResNet-18 training scenarios (*i.e*, CIFAR-10 and PACS). While our strategy may lead to additional computation overhead when the number of clients is high, it only concerns the server. In comparison, IFCA's overhead concerns both the clients and server, and increases drastically with both the complexity of the trained model and the number of clusters to form.

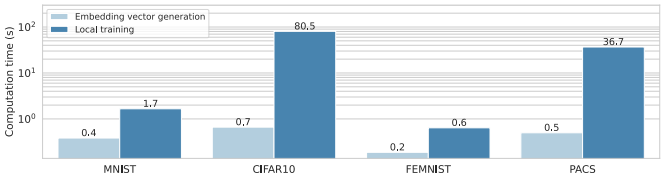


Fig. 6: Embedding vector generation and local training times on a Raspberry Pi 4.

Finally, we investigate the efficiency of each strategy (*i.e.*, accuracy relative to resource consumption). Table V reports the communication and training time of the different strategies to achieve the target accuracy in each scenario. We observe that our strategy achieves top efficiency in all scenarios (*i.e.*, requiring the minimum resources to converge to the target accuracy). Note that our strategy outperforms IFCA in the FEMNIST scenario with respect to accuracy-resources efficiency. Moreover, we observe that IFCA struggles in the CIFAR-10 and PACS scenarios, and LADD only succeeds in a single scenario (*i.e.*, PACS).

Client embedding vector generation overhead To get a better understanding of the client embedding vector generation overhead on resource constrained devices, we run additional experiments on a Raspberry Pi 4 from the Chameleon experimental testbed [38]. For each scenario, we run (1) the client embedding vector generation and (2) a local training round for one epoch and monitor the resulting computation times. We run our experiments using 10 client partitions for each scenario and present the average computation time to run both operations in Figure 6. In the MNIST and FEMNIST scenarios which train LeNet-5, the embedding vector generation task requires respectively $4\times$ and $3\times$ less computation time than one training epoch. In the CIFAR-10 and PACS scenario which train ResNet-18, it requires respectively $115\times$ and $73\times$ less computation time than a local training epoch. In addition to being cost-efficient, the embedding vectors are only computed once before clustering which makes their computational overhead negligible compared to the local training tasks that run for several iterations during the FL training rounds.

VII. Discussion

Generalizability. The experiments in section VI illustrate the effectiveness of our clustering approach in 4 scenarios introducing different conceptual drifts (*i.e.*, rotation, writing style, domain) and label shifts between clients. Each scenario uses a different setting with different datasets, input sizes, number of clients and number of samples per partition. We demonstrate the effectiveness of the autoencoder to efficiently encode clients data. Both the centralized training using public datasets and the federated training of the autoencoder resulted in accurate clustering, highlighting the applicability of our approach to various use cases.

Autoencoder pre-training policy. In our experimental evaluation, we examined two AE pre-training policies: *centralized* and *federated*. The centralized approach brings the benefit of relieving the federated system from any additional system overhead, all while delivering reliable clustering - making it a promising solution for constrained Edge devices. Conversely, the federated approach could be of interest for federations of devices that can accommodate a slight additional overhead, particularly for datasets that are very specific to the application. In such cases, pre-training from a public dataset may lack the specialization required to ensure accurate clustering. Finally, a hybrid approach combining the benefits of both worlds could be a good trade-off. The AE could be pre-trained in a centralized fashion on a public dataset, and then additionally fine-tuned in a federated manner, enabling better personalization with limited overhead.

Optimal number of clusters. In this work, we proposed a mechanism to automatically set the number of clusters through agglomerative clustering and Bayesian optimization. Note that a common practice in FL clustering is to predefine the number of clusters [9], [17], [21] which is rarely applicable in real world scenario. Other works have been using iterative approaches to automatically find the optimal number of clusters (*e.g.*, using gradient similarity), however demanding many rounds to converge which results in waste of client resources [5], [16]. In comparison, our approach based on one-shot clustering enables the clusters to be formed before the training phase and make the best use of client resources.

Drift detection and client migration. A drift detection mechanism could further be derived from our system based on client embedding vectors. To do so, clients could recompute their embedding vector periodically (*e.g.*, once every 50 rounds) and check their consistency with their assigned cluster centroid. Clients could dynamically migrate from one cluster to another if their data distribution changes over time.

Privacy vs. accuracy trade-offs. We leverage randomized quantization on embedding vectors to preserve privacy of individual clients. The level of randomization was fixed to prevent risks of data reconstruction while maintaining accurate clustering. In real-life settings, the level of randomization could be adjusted according to the sensitivity of clients data. If the data is not sensitive, the randomization rate could be decreased allowing for more accurate clustering. Additionally, an adaptive randomized quantization could be applied where each client individually selects its personal privacy budget based on the sensitivity of its personal data allowing for a privacy-accuracy trade-off at each client.

VIII. Conclusion

Statistical heterogeneity among local data distributions (also known as client drift) makes it difficult to optimize a single global model for entire federations of devices. FL clustering aims to solve this problem by grouping clients with similar data distributions to improve training of personalized models. Existing approaches usually focus on the training accuracy, disregarding system constraints, which is problematic for resource-constrained devices, frequently encountered in Edge-to-Cloud environments. In this paper, we make a first step towards tackling this important problem by proposing a new

algorithm for FL clustering based on local compression of clients data using pre-trained autoencoders.

Practicality is a central focus of this work: the implementation of the clustering approach is open-source, while the artifacts of the experimental evaluation will be made publicly available for reproducibility purposes. Extensive experiments on 8 GPUs of the Grid'5000 testbed have underlined the high clustering accuracy of our approach. The evaluation on 4 scenarios introducing conceptual drifts and label shifts demonstrates the capacity to generalize to various problems and the applicability to real-world scenarios. Furthermore, our approach exhibits negligible communication and computation overhead compared to standard FL, while remaining resilient against data reconstruction attacks.

Encouraged by these promising results, in future work we plan to focus on evolving data distributions, by studying the capacity of our approach to detect drifted clients and dynamically migrate clients from one cluster to another. We also plan to investigate centroid update strategies with respect to new clients joining clusters during training.

ACKNOWLEDGMENT

This work was funded by the ENGAGE Inria-DFKI project. In addition, this work was (partially) supported by a French government grant managed by the Agence Nationale de la Recherche under the France 2030 program, reference "ANR-23-PECL-0007" (PEPR CLOUD - STEEL project). Experiments presented in this paper were carried out using the French Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

REFERENCES

[1] B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[2] A. Fallah, A. Mokhtari, and A. E. Ozdaglar, "Personalized federated learning: A meta-learning approach," *CoRR*, vol. abs/2002.07948, 2020.

[3] M. G. Arivazhagan, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *ArXiv*, vol. abs/1912.00818, 2019.

[4] Y. Deng, M. M. Kamani, and M. Mahdavi, "Adaptive personalized federated learning," *ArXiv*, vol. abs/2003.13461, 2020.

[5] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.

[6] P. Beckman, J. Dongarra, N. Ferrier, and G. Fox, *Harnessing the Computing Continuum for Programming Our World*, 2020, pp. 215–230.

[7] H. Hotelling, "Analysis of a complex of statistical variables into principal components." *J. of Educational Psychology*, vol. 24, no. 6, p. 417, 1933.

[8] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," *Advances in neural information processing systems*, vol. 15, 2002.

[9] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *IEEE Transactions on Information Theory*, vol. 68, no. 12, pp. 8076–8091, 2022.

[10] E. A. Huerta, A. Khan, E. Davis *et al.*, "Convergence of artificial intelligence and high performance computing on NSF-supported cyberinfrastructure," *Journal of Big Data*, vol. 7, no. 1, p. 88, 2020.

[11] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed dl," *ACM Comput. Surv.*, vol. 52, no. 4, 2019.

[12] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *IEEE ICC 2020*, 2020, pp. 1–6.

[13] OpenAI, "AI and compute," 2023, https://openai.com/index/ai-and-compute/.

[14] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.

[15] F. Cappello, E. Caron, M. Dayde, F. Desprez, O. Richard, and et. al., "Grid'5000: a large scale and highly reconfigurable grid experimental testbed," in *6th IEEE/ACM Intl. W. on Grid Computing.*, 2005.

[16] M. Mehta, "A greedy agglomerative framework for clustered federated learning," *IEEE T. on Industrial Informatics*, pp. 1–12, 12 2023.

[17] Y. Ruan and C. Joe-Wong, "Fedsoft: Soft clustered federated learning with proximal local updating," in *AAAI Conference on AI*, 2021.

[18] C. Briggs, Z. Fan, and P. András, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, 2020.

[19] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan, "Clustergan : Latent space clustering in generative adversarial networks," in *AAAI Conf. on Artificial Intelligence*, 2018.

[20] Y. Kim, E. A. Hakim, J. Haraldson, H. Eriksson, J. M. B. da Silva, and C. Fischione, "Dynamic clustering in federated learning," in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[21] D. Shenaj, E. Fanì, M. Toldo *et al.*, "Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 444–454.

[22] E. Jothimurugesan, K. Hsieh, J. Wang, G. Joshi, and P. B. Gibbons, "Federated learning under distributed concept drift," 2023.

[23] W. Wang, Y. Huang, Y. Wang, and L. Wang, "Generalized autoencoder: A neural network framework for dimensionality reduction," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2014, pp. 490–497.

[24] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*. Springer, 2017, pp. 373–382.

[25] S. E. Chazan, S. Gannot, and J. Goldberger, "Deep clustering based on a mixture of autoencoders," in *IEEE MLSP)*. IEEE, 2019, pp. 1–6.

[26] A. Z. Tan, H. Yu, L. zhen Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, pp. 9587–9603, 2021.

[27] P. Kairouz, Z. Liu, and T. Steinke, "The distributed discrete gaussian mechanism for federated learning with secure aggregation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5201–5212.

[28] Y. Youn, Z. Hu, J. Ziani, and J. Abernethy, "Randomized quantization is all you need for differential privacy in federated learning," 2023.

[29] K. Chaudhuri, C. Guo, and M. Rabbat, "Privacy-aware compression for federated data analysis," in *Uncertainty in Artificial Intelligence*. PMLR, 2022, pp. 296–306.

[30] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974.

[31] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[32] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[33] 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," *arXiv preprint arXiv:1812.01097*, 2018.

[34] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5543–5551.

[35] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[37] D. Rosendo, P. Silva, M. Simonin, A. Costan, and G. Antoniu, "E2clab: Exploring the computing continuum through repeatable, replicable and reproducible experiments," in *IEEE CLUSTER*, 2020, pp. 176–186.

[38] K. Keahey, J. Anderson, Z. Zhen *et al.*, "Lessons learned from the chameleon testbed," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*. USENIX Association, Jul. 2020, pp. 219–233.