

Federated Clustering: An Unsupervised Cluster-Wise Training for Decentralized Data Distributions

Mirko Nardi^{a,b}, Lorenzo Valerio^b, Andrea Passarella^b

^a*Scuola Normale Superiore, Pisa, Italy*

^b*IIT-CNR, Pisa, Italy*

Abstract

In the domain of decentralized machine learning, Federated Learning (FL) offers a promising paradigm, particularly for scenarios where data privacy is paramount and direct data sharing is not feasible. Although traditionally associated with supervised learning, the potential for its application in unsupervised contexts remains largely unexplored. This paper presents a novel methodology for unsupervised federated learning, aimed at identifying the entire set of categories (global K) across multiple clients within label-free, non-uniform data distribution environments—commonly referred to as Federated Clustering. Central to our approach is the integration of federated cluster-wise training, wherein clients collaboratively train models on clusters deemed as similar, suggesting a shared data distribution.

Our methodology, Federated Cluster-Wise Refinement (*FedCRef*), is underpinned by an iterative process: clients have multiple data distribution in their local dataset (local K) and train models on their initial cluster to create preliminary compressed data representations. These local models are shared across the network, enabling clients to compare models and identify similarities through reconstruction error analysis, thus facilitating the formation of federated groups.

Within these groups, clients engage in collaborative training sessions to build a shared model that accurately represents each data distribution. This federated process is coupled with continuous local clustering refinement, allowing clients to progressively enhance their local cluster splits for more

Email addresses: mirko.nardi@sns.it (Mirko Nardi),
lorenzo.valerio@iit.cnr.it (Lorenzo Valerio), andrea.passarella@iit.cnr.it
(Andrea Passarella)

precise data associations.

As a general outcome, our system comprehensively identifies all potential data distributions across the clients' network (global K) and develops robust representation models for each identified distribution. Initially, we compare our approach with traditional centralized methods to establish a baseline for performance, highlighting the benefits of our distributed solutions. Subsequently, we conduct validation experiments on the EMNIST and KMNIST datasets. These experiments demonstrate our system's capacity to incrementally refine and align cluster models with actual data distributions, significantly enhancing the precision of data representation in unsupervised federated settings.

Keywords: Federated, Learning, Unsupervised, Clustering, Decentralized

1. Introduction

Distributed machine learning at the edge is recognized as an effective strategy to address the limitations of centralized ML systems, including privacy risks, data sovereignty issues, and the high data transmission costs of collecting vast amounts of data centrally [1]. Among various distributed approaches, Federated Learning (FL) stands out [2, 3], due to its effectiveness in training ML models (typically NN-based) preserving data locality. Specifically, FL protocols involve a set of nodes (or *clients*) that individually train a model locally on their own data, and only the model updates are shared. Hence, a global model is iteratively aggregated and distributed to all nodes until convergence. Typically, this process is coordinated by a central server, but serverless protocols have also been proposed [4].

Despite its initial focus on communication efficiency, considerable progress have been made in enhancing several FL's well-known challenges [5], primarily including privacy and security [6, 7], adaptability to the heterogeneity in the data distribution (e.g., non-IID) [8] and variability in participating devices capabilities [9].

However, most research in FL still focuses on supervised tasks [10]. Notably, in many real-world contexts, labeled data is often scarce or absent, and the process of labeling is prohibitively expensive. This discrepancy highlights the need for a greater exploration of unsupervised learning techniques within the Federated Learning framework [5], raising significant challenges in the learning process due to the lack of explicit guidance from labels.

Recent advancements in Unsupervised Federated Learning (UFL) have started to address significant challenges in handling unlabeled data [11, 12]. Two primary approaches are evident. First, the integration of Federated Learning (FL) with Unsupervised Representation Learning (URL) is aimed at learning useful data representations through self-supervised methods within a federated setting. Self-supervised methods extended to decentralized contexts improve privacy and network utilization but require high computational resources that edge devices in FL systems often lack. Second, the clustering of unlabeled data within a federated framework, known as Federated Clustering (FC), whose goal is to identify patterns and group samples with similar features without data centralization, thereby maintaining data privacy and locality.

This work falls into Federated Clustering. This topic is relatively underexplored compared to Unsupervised Representation Learning, especially NN-based solutions, where FL is more prominent. Existing approaches in FC often carry significant limitations, such as reliance on traditional algorithms like K -means [13], and simplistic assumptions about data distribution or datasets and a limited number of clusters [14, 13]. The most relevant works and further discussions on these methodologies are presented in Section 2.

This work introduces a more generalized approach that mirrors real-world scenarios in a more accurate way with respect to the state of the art. We assume a system where each client contains data from multiple data distributions (or categories)¹, referred to as local K . Additionally, clients may share data distributions but do not share overlapping data samples. Consider, for example, a group of hospitals, each possessing unlabeled data related to various diseases. They aim to collaborate in a learning process without directly exchanging patient data in order to determine whether there are common diseases present among their patients. Therefore, we leverage the Federated Learning protocol to identify all the common categories within the system, which we refer to as the global K .

The core objective of our methodology consists of generating cluster-wise federated models, i.e., a model for each data distribution shared by at least two clients, hence training them using the corresponding samples involved. The main idea is to start from local clustering, identify nodes that share

¹In this context, terms *category* and *data distribution* are used interchangeably throughout this document

similar data distributions, and run *parallel* FL protocols among those with the same data distribution. Specifically, our heuristic is formulated as follows: (i) The system initiates with clients having an initial local clustering of their data, where local K is known. (ii) clients train models (e.g., autoencoders) within these clusters to derive compressed data representations for each cluster. (iii) An all-to-all sharing of these locally trained models is then executed to establish cluster associations based on similarities in reconstruction errors. This process facilitates the formation of federated groups. (iv) For each group utilizing FL, a shared model is trained using the samples from the participant clusters. Finally, (v) enhanced FL models are disseminated throughout the network. Clients utilize these federated models to refine their local clusters, enabling them to initiate a new iteration with improved cluster separation. We heuristically define a convergence-stopping criterion by measuring cluster stability at the client level and monitor some system parameters, including the evolution of clusters and the formation of the groups that run the FL processes in parallel.

The underlying intuition is that only coherent clusters —those comprising samples from the same data distribution— form associations. As these clusters are progressively purified through iterative refinement, they form more effective associations. Likewise, our goal is to adjust the system so that models trained on *contaminated* clusters (i.e., clusters containing significant fractions of spurious data, e.g., coming from multiple real data distributions) do not associate.

To summarize, our contributions are:

- A generalized framework for *Unsupervised* Federated Clustering that identifies all the data distributions across the system defined as global K (unknown to the system).
- A novel, iterative federated learning strategy ‘cluster-wise’, where clients contribute to multiple federated models using splits of their data, in contrast to other works that utilize the entire local dataset.
- Evaluation on realistic non-i.i.d. data distributions, i.e., cases where clients possess datasets with a variable number of data distributions (local K). This assumption is not addressed in existing literature, to the best of our knowledge.

This work can be seen as a generalization of our previous efforts [15], where a similar mechanism was employed. In our prior studies, clients locally trained

models and used federated learning to perform anomaly detection. Here, we extend the methodology and broaden the scope to include the refinement of local clusters and the detection of the system data distributions, thereby addressing a wider range of applications in decentralized environments. Quite notably, while our prior work was focused only on anomaly detection (i.e., identifying two groups of data, “normal” and “anomalous” data), here we aim at group data in homogeneous data distributions, which can be an arbitrary number, none of which anomalous (even though the proposed algorithm can be used for anomaly detection as a special case).

The remainder of this paper is organized as follows. Section 2 provides a thorough and comprehensive review of related work in the field of federated learning, unsupervised learning, and their intersection in UFL, highlighting the key challenges and existing approaches. Section 3 introduces the problem statement and the assumptions we make in order to frame the task. Section 4 details our proposed methodology, including the algorithmic framework and the rationale behind our two-fold approach. Section 5 presents the experimental setup, datasets used, and performance evaluation metrics and discusses the results and insights gained from our experiments. Finally, Section 6 concludes the paper with a summary of our findings and directions for future research.

2. Related works

Despite its advantages, Federated Learning introduces a layer of technical complexity to any learning paradigm it is applied to. Although significant research has been conducted on supervised approaches, an important and under-explored area in FL is how to effectively utilize this framework in contexts involving unlabeled data.

To date, few studies have developed Federated Clustering solutions. For instance, *K*-Fed [13] applies the federated learning framework to traditional clustering algorithms like K-means, adapting it to address challenges specific to federated environments such as communication efficiency and device heterogeneity. While *K*-Fed demonstrates effectiveness on various datasets including image data, it remains based on the k-means framework and thus may face similar limitations when dealing with very complex data structures or when the number of clusters is unknown. Clustering more complex data types, such as high-dimensional images, often presents greater challenges in capturing intricate patterns. In this context, IFCA [14] stands out by positing a scenario in which clients are partitioned into distinct clusters, each

representing a unique data distribution. The goal of IFCA is to develop specialized models for each cluster by leveraging the similarity of local models. It proposes a model broadcasting system that enables clients to identify their cluster affiliations, update their models accordingly, and collectively refine the cluster-specific models, based on the premise that clients within a cluster share similar data characteristics.

Expanding on the IFCA, its authors introduced an advanced iteration named SR-FCA [16] as an enhancement that overcomes the limitations of requiring *suitable* cluster initialization and explicit knowledge of the number of clusters. Here, a refinement process is adopted that leverages user similarities to enhance cluster accuracy, while a Federated Learning-based routine within each cluster performs simultaneous training and error correction in clustering.

Despite the innovative clustering approaches of IFCA and SR-FCA and their convergence guarantees, these methodologies exhibit significant limitations in handling non-IID data in complex scenarios. For instance, these approaches assume that each client data belongs to a single cluster, potentially oversimplifying the heterogeneity typical in real-world applications. Moreover, while theoretically robust, their practical efficacy might be constrained when applied to scenarios with dynamically varying data distributions, requiring more adaptive clustering methods.

UIFCA [17] integrates generative models within the IFCA framework to tackle a broader setting where data from the same client may originate from different clusters. Yet, this approach too is tested in relatively straightforward settings, employing synthetic datasets and the MNIST dataset. In the latter case, clustering tasks involve four rotations of a selected digit, with a simplistic non-IID data distribution mechanism. This mechanism primarily relies on a parameter to sample data points from a single cluster, while the remaining points are randomly drawn from any cluster, resulting in a dominant pattern within the clients.

FedCE [18] is another approach that extends federated learning by allowing each client to be associated with multiple clusters. However, this method focuses on enhancing classification performance across diverse data distributions, which are simulated by manipulating samples, such as applying rotations to MNIST digits. Specifically, it clusters clients based on similarities in these engineered distributions, for instance, grouping clients with similar rotation angles. Unlike unsupervised methods, FedCE leverages known class labels throughout the training process, using them to guide both the clustering and the learning of cluster-specific models.

Our research advances beyond existing methodologies by addressing more complex clustering scenarios in federated environments. We contribute by: (i) assuming that clients possess data samples from multiple data distributions, thereby enhancing our model’s ability to manage and learn from the diverse and non-IID data typically found in real-world applications; (ii) proposing that no information about the number of clusters is given to the clients, leaving the discovery of global clusters (global K) to our methodology.

Moreover, during the evaluation phase, our federated approach is rigorously tested across various datasets that are treated as *unlabeled*. Unlike previous methodologies that often rely on arbitrary subsets of data (for instance, selecting as target clusters only specific classes from a dataset like MNIST), our method is tested encompassing the entire dataset, i.e., using *naturally* occurring patterns, based on inherent class distributions. At the conclusion of the clustering process, we compare the formed clusters with the actual data distribution labels, validating the effectiveness and comprehensiveness of our approach. Such an approach not only validates the accuracy of our method against a concrete benchmark but also enhances the clusters’ interpretability in real-world, non-IID data environments.

3. Problem Statement and System Assumptions

To address the challenges mentioned above, this work focuses on scenarios characterized by a non-uniform distribution of categories across clients. Specifically, the system comprises N clients, each with a dataset that may include a variable number of categories (Fig. 1). These categories are subsets of a larger, comprehensive set that spans the entire federated system. For instance, one client’s dataset might encompass samples from two categories, while another might contain samples from four different categories, with no requirement for these category sets to overlap.

Formally we define our federated system as follows: Let there be N clients, denoted as C_1, C_2, \dots, C_N , each with a local dataset D_1, D_2, \dots, D_N . The global dataset D is the union of all local datasets, $D = \bigcup_{i=1}^N D_i$, where each D_i comprises samples from a subset of the total unique data distributions represented in D , denoted by U . The number of clusters in each local dataset D_i is denoted by K_i , where $2 \leq K_i \leq |U|$. A scenario in which each client holds a subset of the global category set U mirrors real-world situations where data is naturally partitioned and incomplete. In supervised contexts, this is referred to as *label distribution skew*, a type of non-IID data distribution [5].

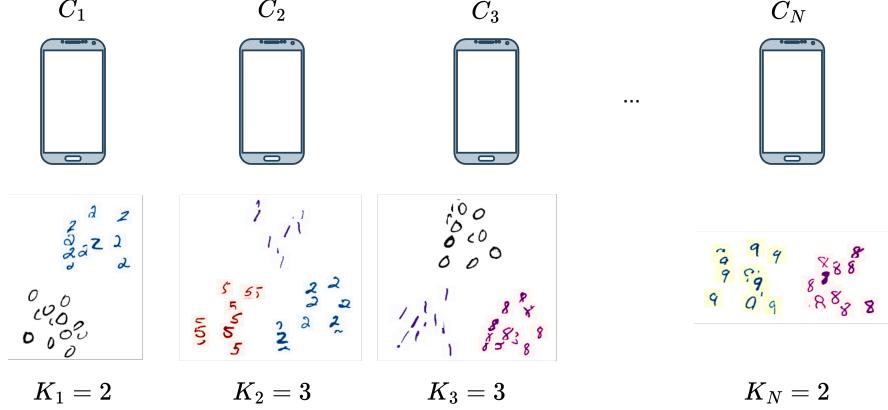


Figure 1: Schematic representation of the considered FL system involving N parties. Each party C_i has a local dataset D_i with K_i representing the number of unique data distributions within that dataset. The goal is to identify the set U , which encompasses all unique data distributions across the system ($|U| = \text{global } K$)

In our federated system, no single client initially knows all categories, assigning the collective task of identifying and learning the complete range of categories present in the dataset to the proposed methodology. Additionally, we operate under the following system assumptions:

- Decentralized (*serverless*) FL protocol with secure communication: clients communicate securely exchanging only model parameters to maintain data privacy.
- Edge computational capabilities: Each client is capable of local model training, although resources are limited with respect to centralized processing capabilities.
- Our framework operates under strict unsupervised conditions, utilizing no labels throughout the process. For the sake of simplicity in our experiments, we assume that only the number of local clusters K_i is known for each client.

Theoretically, the number of local clusters K_i can be initially estimated and fixed on each client at the start of the process, using classic clustering evaluation techniques such as the silhouette score [19].

Given this setting, the goal of this research goal is to enable the system to detect all categories U present in the global dataset and to cluster the data locally at each node precisely into those categories.

This objective is achieved by creating a group of clusters across clients, each containing samples from the same category (possibly with some spurious data due to imprecision of the local clustering algorithms), and by training corresponding federated models for each such group. This approach aims to represent the complete spectrum of categories comprehensively. More specifically, our method leverages these federated models to iteratively refine local clustering. This process enhances the alignment of local clusters with the actual underlying data distributions and thus progressively improves the accuracy of clustering over time. The methodology section (Section 4) that follows will detail the mechanisms by which this collaborative learning and refinement process is implemented.

4. Methodology

In this section, we delve into the methodology designed to address the identified challenges, as comprehensively presented in Algorithm 1. This methodology unfolds through distinct yet interlinked steps, starting with local clustering and model training, progressing through model exchange and cluster association, and advancing to forming and training federated groups. The process ends up in a phase of FL model sharing and cluster refinement, running until stopping conditions based on a convergence criterion are met.

4.1. Step 1: Local Clusters Model Training

This step covers lines 2-8 of Algorithm 1. Considering that each client C_i holds a local dataset D_i with K_i categories, we assume a starting condition characterized by the presence of an initial clustering estimation –this can be either pre-determined or derived (line 3).

Given the initial local clusters, we enable clients to train models that accurately characterize their local clusters. This approach serves two primary purposes: (i) it allows clients to share compressed representations about their clusters without exposing raw data; (ii) it provides the system with a consistent framework for comparing data, facilitating the establishment of associations between clients based on similarities in their local clusters.

For this task, we can select models, such as autoencoders, that are capable of effectively capturing data representations in an unsupervised manner. For

Algorithm 1 Federated Cluster Refinement

```

1: procedure FEDCREF(Clients set  $C$ , percentile  $\alpha$ , association threshold  $\theta$ , stability
   threshold  $\tau$ )
2:   Let  $D_i$  be the local dataset of client  $C_i$ 
3:   Let  $Q_i = \{Q_{i1}, \dots, Q_{iK_i}\}$  be the clusters of  $C_i$ 
4:   Initialize  $A = C$  as the set of active Clients
5:   while (varying  $I$  and  $G$ ) or  $|A| > 0$  do
6:     for each  $C_i \in A$  and each  $Q_{iq} \in Q_i$  do
7:       Train model  $AE_{iq}$  on cluster  $Q_{iq}$ 
8:     end for
9:     All-to-all exchange  $AE_{iq}$  models among  $C_i \in C$ 
10:    for each pair of clusters  $(Q_{iq}, Q_{jp})$  where  $Q_{iq} \in C_i$  and  $Q_{jp} \in C_j$  do
11:       $D1_{\text{norm}} = \text{Normalize}_{[0,1]} |E_{iq \rightarrow iq} - E_{jp \rightarrow iq}|$ 
12:       $D2_{\text{norm}} = \text{Normalize}_{[0,1]} |E_{jp \rightarrow jp} - E_{iq \rightarrow jp}|$ 
13:      if  $Pr\{D1_{\text{norm}} \leq \theta\} \geq \alpha$  and  $Pr\{D2_{\text{norm}} \leq \theta\} \geq \alpha$  then
14:        Associate  $Q_{iq}$  and  $Q_{jp}$ 
15:      end if
16:    end for
17:    Share associations links and initialize  $\Gamma$  as the undirected graph of clusters
18:    Let  $\mathcal{G}$  be the connected component (c.c.)
19:    Let  $G = \{G_i \in \mathcal{G} \mid |V_{G_i}| \geq 2, i = 1, 2, \dots, |\mathcal{G}|\}$  set of c.c. with at least two
   cluster
20:    Let  $I$  be the set of Isolated clusters (isolated nodes)
21:    for each connected component  $G_p$  in  $G$  do
22:      Train model  $AE_{G_p}$  using FL on  $G_p$  clusters
23:    end for
24:    Let  $AE_G$  the set of all trained  $AE_{G_p}$ 
25:    Distribute  $AE_G$  to all  $C_i$ 
26:    for each  $C_i \in A$  do
27:      ClusterRefine( $C_i, AE_G$ )
28:      if  $\text{ACC}(y'_{C_i}, y_{C_i}) \geq \tau$  then
29:         $A = A \setminus C_i$ 
30:      end if
31:    end for
32:  end while
33: end procedure

```

each client C_i , and each identified cluster q of their dataset D_i , a unique model, denoted as AE_{iq} , is trained with the primary objective of accurately reflecting the inherent structure of the data within that cluster (line 7).

The objective is to minimize the reconstruction error of the data points within that cluster. We define the reconstruction error for each sample x_k as $e_k = l(x_k - \hat{x}_k)$, where l represents a function that measures discrepancies between the original data points x_k and their reconstructions \hat{x}_k .

Accordingly, the loss function \mathcal{L}_{iq} to be minimized for training the model AE_{iq} for the q -th cluster within client C_i 's dataset is given by:

$$\mathcal{L}_{iq} = \frac{1}{n_q} \sum_{k=1}^{n_q} l(x_k - \hat{x}_k) \quad (1)$$

After this process, each client possesses K_i models, each fine-tuned to represent a specific local cluster. These models can now be compared with models from other clients to determine if there are similar clusters with common data characteristics, as detailed in the subsequent steps of the methodology.

4.2. Step 2: Model Exchange and Local Clusters Association

The next phase of our methodology involves an all-to-all model exchange among clients (lines 9-16). The sharing of trained models allows clients to evaluate the effectiveness of external models by comparing their reconstruction errors against those of their models. Hence, this comparison enables clients to determine whether their clusters share the same data distribution.

For instance, client C_i evaluates the reconstruction error of its cluster q using the model received and trained on cluster p of client C_j . Conversely, C_i 's model, trained on its cluster q , is tested on C_j 's cluster p . If successful reconstruction is achieved in both cases, clusters p and q likely belong to the same data distribution.

In the methodology, to perform an association between two clusters q and p of two clients C_i and C_j , the core idea is that if the two models AE_{iq} and AE_{jp} can reconstruct the data points similarly, it indicates that they have learned comparable features or patterns from their respective training data (Figure 2).

When C_i receives the model AE_{jp} , it computes the following vector of errors on its cluster q :

$$\mathbf{E}_{iq \rightarrow iq} = [e_1, e_2, \dots, e_{nq}]$$

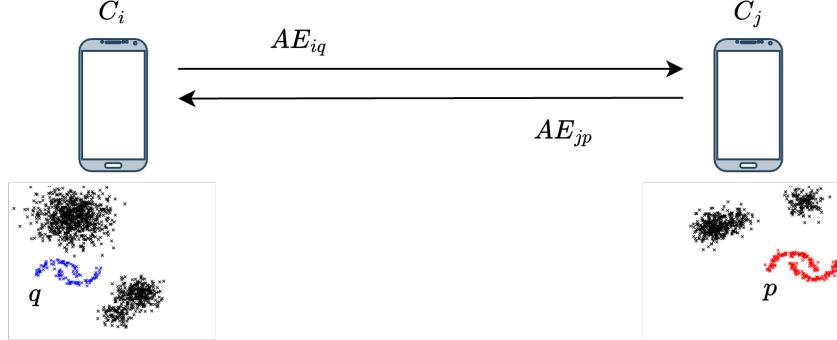


Figure 2: Diagram illustrating the model exchange mechanism: Client C_i assesses the reconstruction effectiveness of a model AE_{jp} received from Client C_j , and vice versa, to establish if clusters q and p indicate similar data distributions.

$$\mathbf{E}_{jp \rightarrow iq} = [e'_1, e'_2, \dots, e'_{nq}]$$

where e_k and e'_k represent the errors for each data point x_k in cluster q when reconstructed by AE_{iq} and AE_{jp} , respectively.

To assess model similarity, we compute the absolute differences between each element of two error vectors and then normalize these differences to the unit range $[0, 1]$. An association between clusters is established if the probability that these normalized differences fall below a predefined threshold θ meets a certain percentile criterion. Specifically, we define that two clusters can be associated if the empirical difference between their reconstruction errors is lower than the threshold θ with a probability equal to α . This is formally expressed as follows:

$$Pr\{\text{Normalize}_{[0,1]}|E_{iq \rightarrow iq} - E_{jp \rightarrow iq}| \leq \theta\} \geq \alpha \quad (2)$$

In this formulation, α represents the α^{th} percentile of the probability distribution of the normalized error differences. A value of α below the threshold θ suggests that the model AE_{jp} can reconstruct the data in cluster q of client C_i nearly as effectively as C_i 's own model AE_{iq} , indicating similar patterns learned by the clusters.

The choice of the threshold and percentile is typically empirical, often based on experimentation or domain-specific requirements. A lower threshold implies stricter criteria for association, ensuring that only models with highly similar reconstruction capabilities are associated.

Note that the bi-directionality of the association is crucial. In the procedure, a link between the two clusters is established only if the association is reciprocal, i.e., in this example, client C_j also receives AE_{iq} from C_i and must use it to effectively reconstruct its cluster p (line 13).

This mutual capability to reconstruct each other's data indicates a strong similarity between the clusters, suggesting they may share the same underlying data distribution. Conversely, it's unlikely that two clusters of clients associate because they have learned similar noisy patterns.

4.3. Step 3: Federated Groups Identification and Training

In this phase, (lines 17-23) clients share associations and construct an undirected graph $\Gamma(V, E)$ with *all* the clusters received by *all* clients as vertices. An edge in the graph is added for any two clusters that are bidirectionally associated in the previous step. For operational simplicity and efficiency, the task of graph construction can be centralized to a single node. It is easy to see that, ideally, this graph would consist of a set of connected components², where each component groups local clusters representing the same category of data. By extracting the connected components from this graph, clients identify groups of clusters to be federated. However, due to imprecisions in local clustering, some connected components may contain spurious clusters, some clusters may remain isolated, or clusters that ideally should belong to the same component may be partitioned into multiple smaller components. This underscores the need for an iterative process to refine the mapping of local clusters to the groups to be federated.

More formally, the extracted set of connected components \mathcal{G} from the graph Γ . Within this set, it identify the *communities* (or groups) of clusters:

$$G = \{G_i \in \mathcal{G} \mid |V_{G_i}| \geq 2, i = 1, 2, \dots, |\mathcal{G}|\} \quad (3)$$

i.e., the connected components consisting of at least two clusters. Clusters that do not belong to any community are labeled as *isolated*, forming the set $I = \mathcal{G} \setminus G$

Upon receiving the connected components, each group of clients can start an unsupervised federated learning instance with a model AE_{G_p} , designed to capture the representations of the clusters involved (Figure 3). This approach

²A connected component is a group of nodes for which it is possible to traverse from any node to any other node in a finite number of steps.

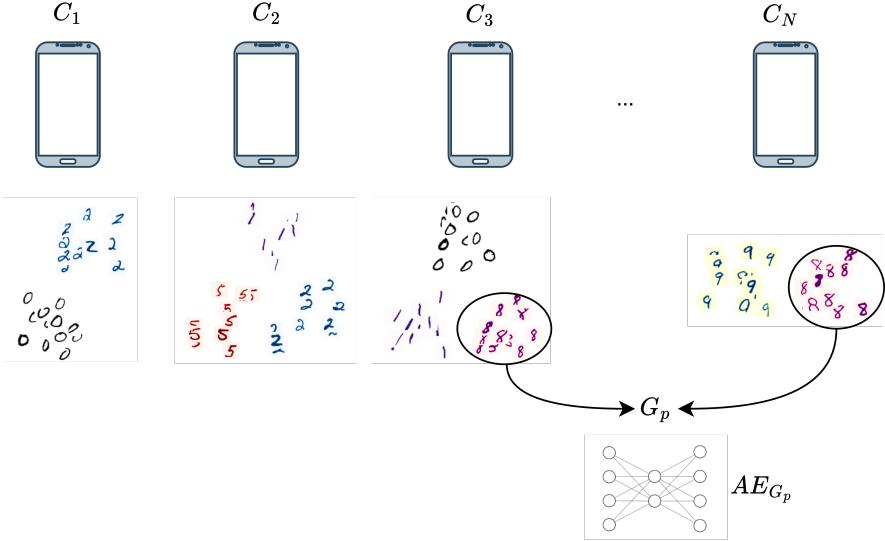


Figure 3: Diagram illustrating the federated groups’ identification and training. In the example, G_p is a group formed by two associated clusters of two clients. A corresponding federated model (AE_{G_p}) is trained using the samples involved.

leverages the diverse data contributions from various clients, allowing the models to develop a deeper understanding of category-specific features.

The federated learning protocol applied here does not specify a particular aggregator, allowing either the server or any participating client –typically the one with the lowest ID– to serve as the aggregator.

An essential aspect of this phase is managing isolated clusters, those that do not join any group. If characterized by significant impurity, indicating incorrect category correspondence, these clusters are excluded from the learning process. This exclusion ensures the integrity of the federated learning by preventing the incorporation of misleading data, thus avoiding incorrect mergers of groups that represent different data distributions.

However, these isolated clusters are not permanently discarded; they are considered for recovery and integration in later cluster refinement steps.

4.4. Step 4: Federated model sharing and Clusters Refinements

This step covers lines 24-31 of Algorithm 1. Here, the trained federated models are sent to all the involved clients. The primary objective at this stage is for each client C_i to refine their local clusters, leveraging both the received federated models $\{AE_{G_g}, g = 1, 2, \dots, |G|\}$ and their own locally trained

models $\{AE_{iq}, q = 1, 2, \dots, K_i\}$. Practically, the output of this process is a new assignment of each data point in the local dataset into K_i distinct clusters.

The cluster refinement process is executed as an individual subroutine by each client (Algorithm 2). At the start of this algorithm, all local samples in the dataset D_i of client C_i are considered unassigned, forming the local set R . As the algorithm progresses, points are assigned to clusters and subsequently removed from R . The process continues until R is empty, indicating that all points have been (re)clustered. The steps involved are as follows:

1. **Models Evaluation:** For each data point x_k in R of client C_i , calculate the reconstruction error using every available model for the client. This is given by the set M including both local model AE_{iq} for each local cluster q and the received federated models AE_{G_q} . The reconstruction error for x_k under model M_j is denoted as e_{k,M_j} (line 6 of Alg. 2)
2. **Best Model Selection:** For each data point x_k in R , determine the model that yields the minimum reconstruction error, i.e., $M_{best}(x_k) = \arg \min_{M_j} e_{k,M_j}$ (line 7 of Alg. 2)
3. **Cluster Assignment:** Identify the model M_{max} that is M_{best} for the highest number of points in R (i.e., the highest number of unassigned data points). Assign these data points to a new cluster H_p (line 9 of Alg. 2). Remove these data points from R and the corresponding model from future iterations.
4. **Iterative Refinement for K_i Clusters:** Repeat the steps of model evaluation, error comparison, and cluster assignment with the remaining unassigned samples. Continue this process until K_i clusters and models are formed for client C_i . (i.e., the set H and M_{select})
5. **Final Assignment of Remaining Samples:** For any remaining unassigned data points, assign them based on the best-performing model (the one that outputs the lowest reconstruction error) out of the M_{select} selected models (line 15 of Alg. 2)

Note that the K_i local models AE_{iq} act as a safeguard, ensuring that if the federated models are not fit to the local data, then the new clustering will not significantly alter the existing partitions. Essentially, if a client's

data lacks the patterns on which the federated models were trained, the local models are more likely to produce the lowest reconstruction error, thereby maintaining stable partitioning.

Algorithm 2 Client C_i Local Clusters Refinement Subroutine

```

1: procedure CLUSTERREFINE( $C_i$ , Received federated Models  $AE_G$ )
2:   Let  $H = \{H_1 \dots H_{K_i}\}$  be the set of  $K_i$  empty clusters for client  $C_i$ 
3:   Let  $R = D_i$  be the set of local unassigned samples
4:   Let  $M = \{AE_{iq}, q = 1, 2, \dots, K_i\} \cup \{AE_{G_g}, g = 1, 2, \dots, |G|\}$ 
5:   for  $p = 1$  to  $K_i$  do
6:     Compute  $e_{k,M_j}$  for all models  $M_j \in M$  and all samples  $x_k \in R$ 
7:     Determine  $M_{best}(x_k) = \arg \min_{M_j} e_{k,M_j}$  for each  $x_k$ 
8:     Select  $M_{max} = \max_{M_j} \text{count}(M_{best}(x_k) = M_j)$ 
9:      $H_p = \{x_k \mid M_{best}(x_k) = M_{max}\}$  ▷ p-th cluster is formed
10:     $M_{select} = M_{select} \cup M_{max}$ 
11:     $M = M \setminus M_{max}$ 
12:     $R = R \setminus H_p$ 
13:   end for
14:   for each remaining unassigned  $x_k \in R$  do
15:     Assign  $x_k$  to corresponding cluster of the best performing model
        of  $M_{select}$ 
16:   end for
17:   Set  $H$  as the new local cluster partitioning of the local dataset  $D_i$ 
18: end procedure

```

4.5. Step 5: System Convergence and Stopping Conditions

After the clients refine their local clusters using the federated models, before starting a new iteration we check the stopping conditions. These are implemented based on the state observed at the end of each iteration before initiating a new one.

Client-level Local Clustering Stability Check: This process involves comparing the cluster labels from previous and current iterations within each client. We define the set of active clients in any iteration as A . An active client is one whose local clustering has not yet reached a state of stability based on a chosen metric. We utilize the unsupervised clustering Accuracy (ACC), also referred to as clustering accuracy:

$$\text{ACC}(y', y) = \frac{1}{n} \max_m \sum_{i=1}^n 1\{y'_i = m(y_i)\} \quad (4)$$

This metric is used to measure the agreement between two sets of labels for the samples, say y and y' , regardless of how the labels are named. It is commonly used to find the best match between predicted clusters and ground-truth labels, when available, but it can also be used to measure the agreement between two different clustering results. For instance, if the vectors of labels y and y' have different symbols but perfectly aligned, the ACC will be 1, indicating perfect agreement.

In the formula, n represents the total number of samples, y'_i and y_i are the labels for the i -th sample in each assignment respectively. The function m is a mapping that rearranges the labels of y to best match those of y' . The indicator function $1\{\cdot\}$ checks if the labels match under this mapping, returning 1 for a match and 0 otherwise. The ACC is then calculated as the maximum sum of these matches, normalized by the number of samples, effectively measuring the proportion of samples that are correctly matched under the best possible label alignment. To avoid checking all possible permutations, the Hungarian algorithm is used to find the optimal mapping in polynomial time [20].

In this step, the ACC is evaluated between the previous (y') and current (y) cluster labels for each specific client C_i :

$$A = \{C_i : \text{ACC}(y'_{C_i}, y_{C_i}) < \tau\} \forall C_i \in C$$

A client is removed from the active set A if its ACC exceeds a predefined threshold τ , which we empirically set within the range $[0.8, 1]$. When the ACC is higher than τ , the difference between two iterations' clustering is minimal, suggesting that the client's clusters have stabilized.

Clients that have achieved this level of stability, also referred to as *non-active* clients, are still required to participate in the federated learning process. While these clients halt further training of their local models and refinement of their clusters (lines 7 and 26 of Algorithm 1, respectively), they continue to engage in the exchange of models and potential new associations with other participants. This is crucial since it anchors the federated network's stability by providing a consistent set of samples (i.e., clusters) and models. Note that stable clients with incorrectly split clusters remain involved in the

model exchange but are less likely to form associations with other clients. Consequently, they do not contribute to the process with their noisy data.

Global-level Stop Condition: In addition to the other condition, based on the active clients in the process, we monitor the trends in the number of isolated clusters (I) and the formation of communities (G) within the system. The process is halted if these metrics remain stable over a predetermined number of iterations, indicating no significant changes. In our experiment, we specifically use the criteria that when both the number of communities and the number of isolated clusters do not change by more than 10% over three consecutive iterations, the process is stopped. This ensures that the process halts when further iterations are unlikely to yield significant changes.

In summary, the iterative federated learning process takes into account both the stability of individual clients and the overall system metrics. The process concludes either when there are no more active clients remaining in set A , or when the global stopping condition is satisfied.

5. Experimental setting

In this section, we present a series of experiments designed to evaluate the effectiveness of our methodology. We start by describing the datasets selected for the analysis and the setup creation process, which measures the system’s performance in different contexts and the range of effectiveness.

5.1. Dataset

Our study utilizes the EMNIST [21], KMNIST, and KMNIST49 datasets [22]. We use the actual classes of these datasets as target data distributions (global K or $|U|$).

EMNIST Digits is an extended version of the classic MNIST dataset. It includes 280,000 grayscale images of handwritten digits (10 classes), each of size 28x28 pixels. This dataset offers a more extensive and diverse set of samples than MNIST, making it ideal for testing our system in scenarios with numerous clients while maintaining the familiar MNIST structure.

The KMNIST (Kuzushiji-MNIST) dataset comprises 70,000 28x28 grayscale images of handwritten Japanese characters (10 classes, similar to MNIST’s 10 digits). It introduces a different pattern recognition challenge due to the distinct characteristics of Japanese script compared to Arabic numerals. KMNIST tests the algorithm’s adaptability and effectiveness in dealing with

a different yet structurally comparable type of data, offering insights into the model’s versatility.

Finally, expanding on the challenges presented by KMNIST, we use KMNIST49 (Kuzushiji-49), which includes a more extensive set of 270,912 images across 49 classes, each representing different Japanese characters. The dataset maintains the 28x28 pixel grayscale format. Being designed for classification tasks, KMNIST49 is unbalanced, with some classes having very few samples. To ensure a disjoint partition and maintain the focus on robust class representation, we have removed the classes with fewer samples, resulting in a dataset encompassing $|U| = 31$ classes with around 6000 samples each. The increased class count and data volume of KMNIST49 allow us to assess the scalability and accuracy of our algorithm in more demanding classification scenarios, particularly in handling a higher number of categories and more complex character structures.

Being an unsupervised task, we use the actual labels in two phases: (i) at the initialization phase, where the actual labels are used to generate the system of clients and to distribute the samples among them, and (ii) at the evaluation phase, i.e., when the algorithm terminates, the actual labels are used to assess the accuracy of the clusters formed by each local client (section 5.3).

5.2. Experimental setup and hyperparameters

We have designed a system setup that allows us to initialize experiments based on a specified client configuration. If not explicitly specified in the experiments, default values are assumed. The key parameters of this setup include:

Number of Clients (N): A variable number of clients is configured for each experiment to examine the algorithm’s performance under different network sizes. Default value is $N = 25$.

Global Dataset (D): The dataset used to initialize the system is selected from those listed in Section 5.1. It sets the target number of data distributions (global K or $|U|$) that the algorithm aims to identify.

Number of Clusters per Client (K_i): Each client is randomly assigned a number of clusters, ranging from 2 to $|U|/2$, where $|U|$ is the total number of distinct data distributions (or categories) in the global dataset, such as $|U| = 10$ for EMNIST. For example, considering EMNIST, a client could

be assigned anywhere between 2 to 5 data distributions (e.g., samples from classes 4, 5, and 8). For each assigned data distribution, the client is initialized with S unlabeled samples. This setup simulates a scenario where clients have uneven exposure to the overall data patterns, reflecting a distribution skew among the clients. Clients would compute their own K_i using methods like the silhouette coefficient to find the optimal value of K_i . However, for simplicity, in our experiment, we assume K_i is provided to the clients.

Number of Samples per Cluster (S): Each client is allocated 500 samples per cluster. This allocation ensures an adequate data volume for effective local model training while still presenting a challenge due to limited data diversity.

Percentile, Association, and Stability Thresholds (α, θ, τ): The α th percentile of the probability distribution, the association threshold, and the internal ACC stability threshold are set after a series of validation runs to ensure that associations are meaningful. Default values are 75, 0.2, and 0.8 respectively.

Overlap Fraction (O): This parameter measures the degree of shared data among client clusters introduced at system generation time. The default setting is 0 overlap.

Initial Clustering Methods: Two distinct clustering approaches are employed at the outset of the federated learning process: *DEC (Deep Embedded Clustering)* [23]: this method represents a realistic clustering approach, where each client independently performs clustering on their local data. *Dirty Uniform Clustering*: this simulated clustering method involves mapping the samples to wrong cluster labels with a fixed probability (parameterized by **dirtiness**), allowing us to test the algorithm’s resilience and error-correction capabilities. Furthermore, we use the latter for faster experiment initialization, avoiding the high computational load and extensive work of fine-tuning required by DEC. Note that, for the purpose of testing our methodology, the choice of the *initial* clustering method used by clients is orthogonal, as its only purpose is to provide an initial split of local data into local clusters. Therefore, using a simulated clustering algorithm does not limit the significance of the presented results while allowing us finer control of the imprecision of the initial clustering. Still, we also perform tests with a real state-of-the-art

clustering algorithm, such as DEC, to test the methodology in a complete pipeline.

Models architecture and setting: We employ fully connected autoencoders with identical configurations for both local training and federated models, featuring an encoder layer setup of $d - 100 - 64 - 32$ and a mirrored decoder, where d is set to 784 to match the 28x28 format of MNIST-like images. Optimization is achieved using the Adam optimizer with MSE as the loss function. Our findings indicate no performance gains from additional layers. The federated learning cycle is completed over 15 rounds, using FedAvg [2] as aggregation protocol. For Deep Embedded Clustering (DEC), we strictly follow the original paper’s specifications, employing an encoder of $d - 500 - 500 - 2000 - 10$.

5.3. Performance Metrics

We monitor several key metrics to evaluate the effectiveness of the clustering process. The main target is the number of communities ($|G|$) that should align to the number of global categories ($|U|$) to ensure that the system accurately reflects the underlying data distribution.

To asses that the communities found are correct, we track two additional metric: (i) the number of isolated clusters ($|I|$), which are clusters not part of any groups in G (ideally, this count should be zero, assuming that at least two clients share the same class); and (ii) the number of wrong associations ($|W|$), measured as the number of local clusters associated with the wrong federation group, with the objective of achieving zero wrong associations. We point out once more that the proposed algorithm is not aware of these metrics and the knowledge behind. These metrics are determined by comparing the cluster association graph against the ideal graph, where connections are based on the clusters correct labels, which are known during the initialization phase.

Moreover, recalling that the local clusters refinement is key for continuously improving data associations and developing more precise federated models –as it ensures that the samples involved more accurately reflect the correct data distribution– we utilize the unsupervised clustering Accuracy (ACC) defined in Eq. (4) to monitor the local accuracy of the clients. Specifically, we compute $\text{ACC}(y_t, y)$ for each client, where y_{t_i} are the ground truth labels and y_i are the clustering labels produced by the algorithm.

Unless otherwise stated, the results presented hereafter are averages over five simulation results, while confidence intervals are computed at a confidence level of 95%.

5.4. Results

5.4.1. Experiment set 1: Overall performance with DEC clustering

The primary objectives of this set of experiments are: (i) to provide a practical understanding of the methodology (FedCRef); (ii) to assess the overall performance of our methodology with a complete and realistic algorithmic pipeline comprised of a state of the art clustering algorithm, i.e., DEC; (iii) to compare our method with an equivalent centralized approach, where all data from the generated system is collected and analyzed using DEC.

Experiment setup. For this purpose, we present detailed results on the EMNIST dataset obtained by using Deep Embedded Clustering (DEC) as the initial clustering method. We configured DEC hyperparameters, as in the original paper, to maintain a standard performance baseline. We use the default values described in Sec 5.2 to generate the distributed system. The centralized setup is built by merging the distributed datasets from the decentralized setup.

Table 1: Clustering Outcomes of our FedCRef with *DEC* as the local clustering method for clients (Mean \pm 95% CI) over five runs

Dataset	ACC init (Local DEC)	ACC End	Comm. Found (K) / Real Global K	Wrong Ass. (%)	Isolated Cl. Start → End
EMNIST	0.781(\pm 0.045)	0.894(\pm 0.031)	10.8(\pm 1.1) / 10	1.1(\pm 1.6)	87.5(\pm 3.2) → 19.1(\pm 7.2)

Results Analysis. In Figure 4, we provide a detailed analysis of the algorithm’s performance on the EMNIST Digits dataset over iterations. These key metrics are also summarized in Table 1. Specifically, Figure 4a shows the formation of Communities of clusters (G), which initially begin at zero and quickly stabilize close to the actual data distribution (global K). Conversely, Figure 4d depicts the reduction of the number of isolated clusters, starting from an average of 87.5 across 25 clients and eventually reducing to about 19. These remaining clusters, while isolated, continue to receive (if active) the federated models to benefit from the collective learning process and potentially refine their local clusters.

The effectiveness of our algorithm in accurately identifying data distributions is evidenced by the minimal number of incorrect associations between clusters. As shown in Figure 4c, the algorithm consistently maintains nearly zero wrong associations, demonstrating its capacity to tolerate minor errors without compromising the integrity of the community structure.

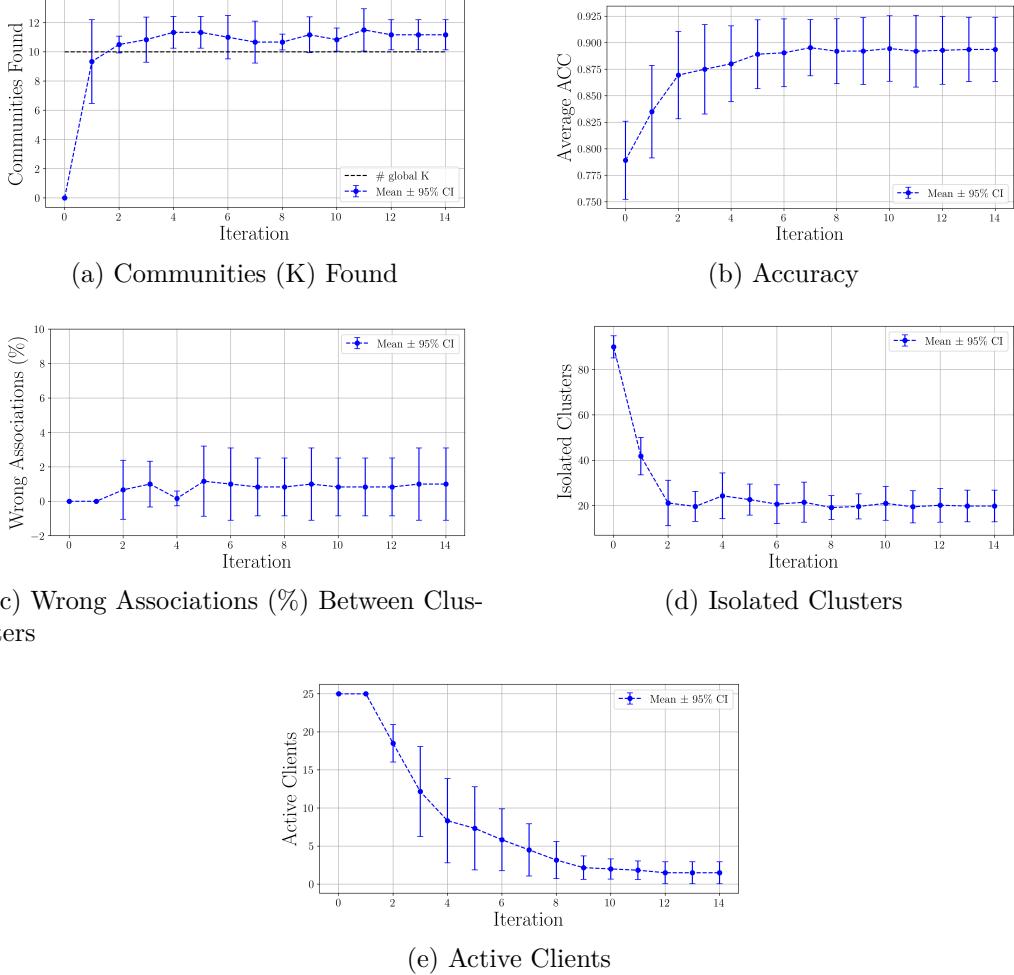


Figure 4: Performance Metrics of FedCRef metrics over iterations on the EMNIST Digits Dataset (global $K = 10$ data distributions). Mean \pm 95% confidence interval across five runs. Client initialization ranges randomly from 2 to $|U|/2$ clusters, resulting in an average of 3.5 clusters. The initial local clustering is established using the DEC algorithm.

The unsupervised accuracy (ACC), illustrated in Figure 4b, shows significant and consistent improvement throughout the iterations. Starting with an initial average accuracy of 0.78 (computed by DEC), the algorithm fine-tunes the local clusters by integrating global system knowledge, leading to a final average accuracy of 0.894 with a narrower confidence interval.

Lastly, the number of active clients, shown in Figure 4e, decreases over time. This indicates that fewer clients need to actively refine their clusters as the system approaches a stable condition, i.e., where the global data distributions are established.

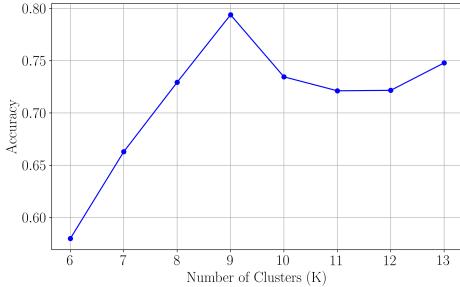
We now compare the performance of our method in this scenario with an equivalent centralized solution, in which all data are sent to a central server where the DEC method is performed. In this case, the optimal number of global distribution (global K) is determined using the silhouette score [19], which assesses performance across various values of K. The comparison results are presented in Table 2: the decentralized method (FedCRef) outperforms the centralized approach (Global DEC) in terms of accuracy, with an ACC of 0.894 compared to 0.793. Such an improvement of our decentralized solution over the centralized one shows the intrinsic difficulty in finding the correct number of clusters a single model faces when the input space is multi-dimensional. Conversely, in decentralized settings, due to non-iid data partitioning, each local model is exposed only to a subset of categories, making the multidimensional clustering problem less challenging. This suggests that decentralized and federated clustering might be, in general, a successful approach to solving the problem, with the advantage of keeping the size of the model to be trained contained.

The silhouette score for the centralized scenario is evaluated after dimensionality reduction (2D) using t-SNE³, which is necessary due to the reliance of the silhouette score in a lower-dimensional space. Centralized performance evaluation is shown in Fig. 5b.

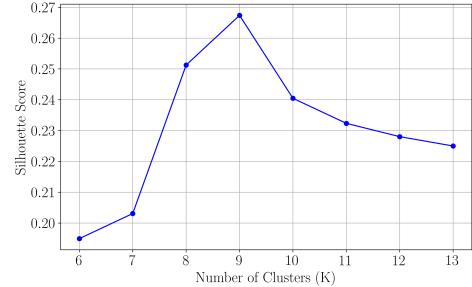
5.4.2. Experiment set 2: impact of initial clustering precision

Experiment setup. In this set of experiments, we employ a *simulated clustering* approach across all selected datasets. As detailed in Section 5.2, this method enables precise control over the initial clustering precision performed

³EMNIST is very similar to MNIST, which is well-known for being effectively separated into distinct clusters by t-SNE due to the characteristics of its data distribution.



(a) Centralized DEC: Accuracy by Varying K



(b) Centralized DEC: Silhouette Score by Varying K

Figure 5: Centralized approach (clients EMNIST splits collected) with DEC clustering: performance evaluation by varying the number of clusters K . Silhouette score analysis is performed using t-SNE dimensionality reduction to ease clustering evaluation. Results are presented with a mean across five runs.

Table 2: Comparison of final Accuracy (ACC) and clustering performance using the EMNIST dataset: our decentralized method (FedCRef) versus a centralized approach with DEC. The optimal cluster number in the centralized model is assessed using the silhouette method

Dataset	Method	Paradigm	ACC	K Found / Real Global K
EMNIST	FedCRef (our)	Decentralized	0.894	10.8/10
	Global DEC	Centralized	0.793	9/10

by each client, allowing us to study the sensitivity of our method to this critical parameter. The **dirtiness** parameter, which represents the probability that a sample is initialized to the wrong cluster, is used to achieve this control. The other parameters not specified are set to their default values.

It is important to note that a dirtiness value of 0 can be considered a best-case baseline, as all clients start with the correct data distribution split. This condition serves as a reference point for comparing the algorithm’s performance under different levels of initial clustering precision. The other values of dirtiness are 0.3 and 0.5, respectively. In this way, we can test the performance of our solution with a level of dirtiness slightly higher than the one we would get from applying a real clustering method, i.e., DEC produces a level of dirtiness around 0.2. Dirtiness 0.5 represents a particularly unfavorable starting condition, simulating a technique like DEC, making more than twice the errors in clustering formation.

Results Analysis. In Table 3, the clustering outcomes are shown. Despite varying starting dirtiness, the algorithm consistently identifies the correct number of communities corresponding to actual data distributions in both the EMNIST and KMNIST datasets. Lower dirtiness yields minimal wrong associations, indicating high precision in community formation. A significant decrease in isolated clusters indicates the algorithm’s ability to consolidate clusters, especially when starting from cleaner conditions. In contrast, the KMNIST-49 dataset’s complexity (global $K = 31$) limits comprehensive data distribution detection. Nonetheless, the algorithm’s conservative approach prioritizes accurate community detection with minimal incorrect associations at the expense of more isolated clusters.

Table 4 summarizes the clients’ local accuracy. As dirtiness increases, we observe a reduction in final accuracy (ACC End) across all datasets. However, this reduction still represents a significantly *higher* improvement relative to the initial accuracy (ACC Init). While the confidence intervals widen with greater dirtiness, indicating more variability, the consistent improvement across all datasets suggests that the algorithm effectively adapts to noise.

We also include a baseline (ACC *Perfect Association*) where the association graph between clusters is provided, meaning the clients do not need to compare and associate their clusters to form associations for federated group training. This baseline achieves very high accuracy, and our experimental results closely approximate this benchmark, demonstrating the robustness of our method.

The least variability in performance improvement with changing dirtiness levels is observed in KMNIST-49, suggesting that high number of data

Table 3: Clustering Outcomes across different datasets (average values over five runs), including 95% confidence intervals (CI). *Simulated clustering* is used as the initial clustering method, with varying **dirtiness** levels.

Dataset	Dirtiness	Comm. Found (K) / Real Global K	Wrong Ass. (%)	Isolated Cl. Start → End
EMNIST	0.0	10.1(± 0.7) / 10	0.0(± 0.0)	89.4(± 2.5) → 17.3(± 3.9)
	0.3	9.8(± 1.1) / 10	5.1(± 1.7)	87.6(± 4.2) → 16.1(± 1.3)
	0.5	10.2(± 0.9) / 10	6.9(± 1.2)	89.4(± 5.6) → 48.0(± 3.2)
KMNIST	0.0	9.8(± 1.6) / 10	0.0(± 0.0)	87.0(± 6.7) → 18.6(± 3.4)
	0.3	9.8(± 0.8) / 10	2.0(± 1.1)	88.1(± 4.3) → 19.2(± 1.4)
	0.5	10.8(± 0.9) / 10	3.1(± 0.9)	86.4(± 3.8) → 31.0(± 2.5)
KMNIST-49	0.0	27.2(± 1.6) / 31	2.4(± 2.3)	201.4(± 11.5) → 40.0(± 7.8)
	0.3	27.4(± 2.6) / 31	7.4(± 2.2)	197.8(± 14.5) → 53.4(± 4.3)
	0.5	24.8(± 4.4) / 31	8.2(± 3.0)	218.6(± 17.2) → 89.8(± 5.1)

distributions in the dataset may play a role in algorithmic stability.

Table 4: Accuracy across different initial dirtiness levels with *simulated clustering* as local clustering method for clients. Average values over five runs per configuration (a single row), including 95% confidence intervals (CI).

Dataset	Dirtiness	ACC Init	ACC End (Our)	Improvement (%)	ACC End (Perfect Associations)
EMNIST	0.0	1.0	0.948(± 0.011)	-5.2(± 1.05)	0.982(± 0.001)
	0.3	0.701	0.879(± 0.028)	25.392(± 4.00)	0.949(± 0.0097)
	0.5	0.503	0.711(± 0.049)	41.352(± 9.74)	0.914(± 0.0059)
KMNIST	0.0	1.0	0.93(± 0.016)	-7.0(± 1.6)	0.963(± 0.0025)
	0.3	0.700	0.863(± 0.031)	23.286(± 4.43)	0.933(± 0.0047)
	0.5	0.504	0.765(± 0.058)	51.786(± 11.51)	0.905(± 0.0059)
KMNIST-49	0.0	1.0	0.889(± 0.008)	-11.1(± 0.8)	0.933(± 0.0041)
	0.3	0.699	0.769(± 0.019)	10.014(± 2.72)	0.889(± 0.0029)
	0.5	0.501	0.622(± 0.065)	24.152(± 12.97)	0.849(± 0.0011)

5.4.3. Experiment set 3: Impact of data Overlap

Experiment Setup. In this series of experiments, we explore how data overlap influences the performance of our methodology. We analyze the effects of data overlap at three distinct levels: 0 (no overlap, the default assumption), 0.3 (moderate overlap), and 0.5 (high overlap). For instance, a 0.3 overlap means that when clients clusters are initialized from the same

data distributions, the 30% of the samples of that cluster is in common. We utilize the EMNIST dataset as reference, maintaining a *dirtiness* level of 0.3. *Results Analysis* In all the overlap settings, the proposed method consistently identifies the correct number of communities aligned with the number of global data distribution K without any wrong associations. The count of isolated clusters notably decreases by the end of the process, but the most significant reduction occurred in the high overlap condition, where isolated clusters dropped from 90.167 to 1.333. Table 5 summarizes these final clustering outcomes, while the detailed trends over iterations can be seen in Figure 6.

The final ACC demonstrates relevant performance improvement as data overlap increases, as shown in Table 6. With no overlap, the final ACC is 0.903; this rises to 0.955 with moderate overlap (0.3) and further increases to 0.976 with high overlap (0.5). The improvement in ACC, calculated as the percentage increase from initial to final values, is 28.816% with no overlap, 36.234% with moderate overlap, and 39.23% with high overlap. Additionally, the 95% confidence intervals for ACC become narrower with increased overlap, indicating enhanced precision in the final accuracy measurements. These results are also very close to the accuracy obtained with the perfect association baseline.

These findings indicate that an increased degree of data overlap in federated unsupervised learning scenarios enhances both the accuracy and the precision of clustering outcomes.

The fact that data overlap helps precision of clustering is expected, as clients see more homogeneous samples, and therefore the models trained on local data are more precise when used on other clients' datasets. However, we wish to remark on the notable performance achieved by our method, even in the case of no overlap, which is the most challenging configuration.

Table 5: Clustering Outcomes at varying levels of Overlap (average values over five runs). Tests on the EMNIST dataset and fixed dirtiness at 0.3 as initial configuration.

Dataset	ACC init	Overlap	Comm. Found (K)/	Wrong Ass.	Isolated Start → End
			Real Global K		
EMNIST	0.7	0	9.8(±1.7) / 10	0.0	88.2(±3.2) → 18.8(±9.3)
		0.3	10(±0.4) / 10	0.0	89.571(±5.8) → 8.429(±3.2)
		0.5	10(±0.1) / 10	0.0	90.167(±4.4) → 1.333(±0.8)

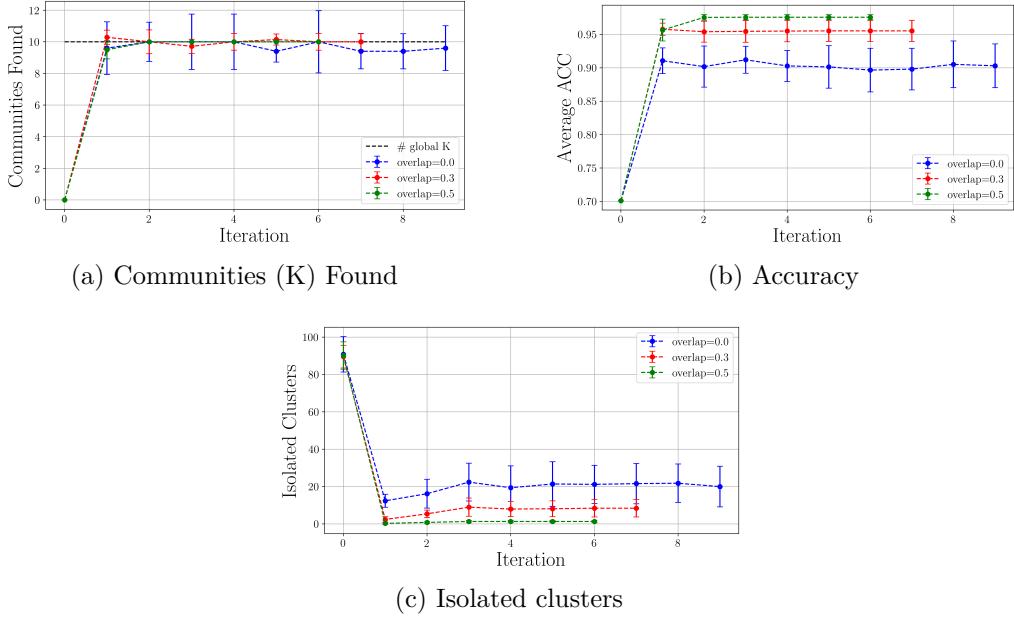


Figure 6: Algorithm’s performance metrics trends over iterations across different Overlap levels: 0 (No Overlap), 0.3 (Moderate Overlap), and 0.5 (High Overlap). Tests on the EMNIST dataset with *simulated clustering* and fixed dirtiness at 0.3 as the initial configuration. The mean \pm 95% CI are computed over five runs.

Table 6: Mean \pm 95% CI final ACC and ACC Improvement across different Overlap levels. Tests on the EMNIST dataset and fixed dirtiness at 0.3 as in the initial configuration.

Dataset	ACC Init	Overlap	ACC End	Improvement (%)	ACC End
			(Our)		(Perfect Associations)
EMNIST	0.7	0	0.903(± 0.033)	28.816(± 3.16)	0.953(± 0.015)
		0.3	0.955(± 0.016)	36.234(± 2.32)	0.962(± 0.012)
		0.5	0.976(± 0.004)	39.23(± 1.12)	0.97(± 0.003)

5.4.4. Experiment set 4: Impact of the number of clients

Experimental Setup. In this final set of experiments, we assess the system’s scalability by testing performance across various numbers of participating clients (N).

We conduct trials with systems configured for 20, 30, 40, and 50 clients to evaluate the impact on the efficacy of the algorithm.

Consistent with earlier experiments, we use the EMNIST dataset with a *dirtiness* level of 0.3, resulting in an average simulated initial clustering accuracy of 0.7 for each client. Note that, as the number of clients increases, the total number of clusters across all clients also rises; for instance, with $N = 50$ clients, the average number of clusters reaches 177. Moreover, since the default parameters are maintained unless specified otherwise, the number of samples initialized per client remains fixed, leading to an increase in the total number of global samples as the number of clients grows.

Results Analysis.

As shown in Table 7, community detection accurately reflects the 10 global data distributions for up to 40 clients, with few incorrect associations and a low count of residual isolated clusters, suggesting a high correspondence to the real category splits. However, with 50 clients, we observed some incorrect associations and the formation of more communities within the same data distribution, although there was still a significant increase in final accuracy (ACC).

The rate of incorrect associations remains stable up to 50 clients, but the number of detected communities increases, indicating the formation of separate communities within the same data distribution. Similarly, the number of isolated clusters is well-controlled, but the tendency to form more isolated clusters increases with this number of clients.

Despite these challenges, there is still a notable improvement in final ACC up to 50 clients, as shown in Table 8. However, a high number of clients results in excessive partitioning of the local data, which is an intrinsic limitation in federated learning. Detailed trends over iterations are illustrated in Figure 7.

6. Conclusion

Our study introduces a novel methodology for unsupervised federated learning that effectively addresses the inherent challenges posed by non-uniform data distributions across a network of decentralized clients where label information is completely absent (i.e., Federated clustering).

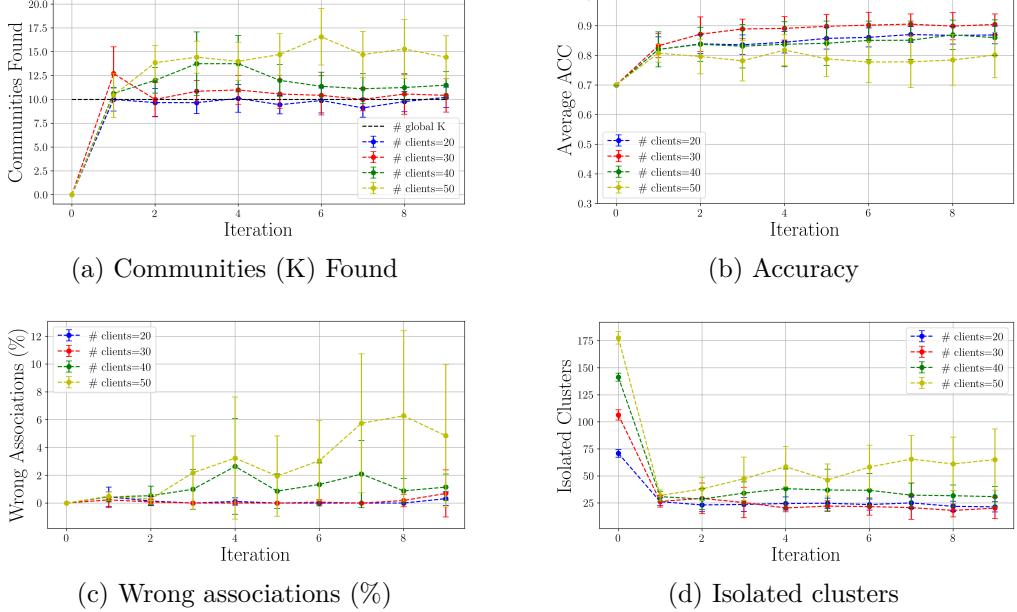


Figure 7: Algorithm’s performance metrics trends over iterations varying the number of clients of the system and default 0 overlap. Tests on the EMNIST dataset with *simulated clustering* and fixed dirtiness at 0.3 as the initial configuration. The mean \pm 95% CI is computed over five runs.

Table 7: Mean Clustering Outcomes at varying the number of clients of the system. Tests on the EMNIST dataset and fixed dirtiness at 0.3 as the initial configuration.

Dataset	ACC init	Clients (#)	K Found/ Real Global K	Wrong Ass. (%)	Isolated Start \rightarrow End
EMNIST	0.7	20	10.2(± 1.1) / 10	0.222(± 0.339)	70.9(± 3.5) \rightarrow 21.4(± 4.7)
		30	10.4(± 1.8) / 10	1.429(± 3.496)	106.4(± 4.8) \rightarrow 20.3(± 9.8)
		40	11.5(± 1.4) / 10	3.375(± 2.824)	141.4(± 3.7) \rightarrow 30.9(± 9.4)
		50	14.4(± 2.3) / 10	8.714(± 5.719)	177.7(± 5.7) \rightarrow 65.1(± 28.4)

Table 8: Mean \pm 95% CI final Accuracy (ACC) and Improvement varying the number of clients of the system. Tests on the EMNIST dataset and fixed dirtiness at 0.3 as the initial configuration.

Dataset	ACC Init	Clients (#)	ACC End (Our)	Improvement (%)	Acc End (Perfect Associations)
EMNIST	0.7	20	0.869(± 0.03)	24.143(± 4.286)	0.921(± 0.041)
		30	0.905(± 0.035)	29.101(± 5.064)	0.937(± 0.029)
		40	0.86(± 0.06)	22.857(± 8.571)	0.953(± 0.007)
		50	0.801(± 0.08)	14.265(± 10.913)	0.955(± 0.010)

In the methodology, we integrate federated *cluster-wise* training among clients, where they collaborate on training models for clusters they identify as similar (i.e., from the same data distribution). This is coupled with iterative local clustering refinement, allowing clients to enhance their local cluster splits for more accurate associations.

The aim is to identify the entire set of data distributions as precisely as possible and thus train a separate model for each of them. Unlike similar studies, we navigate a more complex landscape where multiple clusters per client are assumed—contrary to the conventional one-cluster-per-client scenario.

Our extensive empirical investigation utilizes three datasets—EMNIST-Digits, KMNIST, and KMNIST49—while employing small models, specifically flat autoencoders, suitable for edge devices. Each client begins with an initial local data distribution count (local K or K_i), which is a subset of the target global data distributions (global K).

The clients in each experiment aim to (i) identify their own data distributions, (ii) find other clients with common data distributions in their local datasets, and (iii) iteratively improve their local data splits. We tested the system using different metrics, including the data distribution found by clients versus the real global data distribution and the improvement in local splits measured as initial and final accuracy.

We successfully tested the system under realistic conditions, varying the number of clients up to 50, and found that it performed well within this range. Our experiments also showed that when there is some overlap in the data among clients, the system performs even better as the overlap increases.

For comparison, we used several baselines: the main scenario is tested against the centralized approach, where all clients send their data to a central server, and a state-of-the-art clustering algorithm (*Deep Embedding Clustering*) is applied to the entire dataset. We also compared results with

tests where all the correct associations between client splits are provided and/or clients start with a clean split, i.e., are initialized with the correct data distribution in their local samples, providing an upper bound reference.

The experiments underscore the robustness of our methodology. In a few iterations, the system successfully detects the actual global data distributions, providing federated models for each of them and correcting the initial local clusters, thereby improving the average local accuracy up to 0.95.

Additionally, simulated tests were conducted to introduce controlled errors in the initial cluster split through varying dirtiness levels. This approach demonstrated the method’s robustness, maintaining effectiveness even at high levels of dirtiness.

Acknowledgement

This work was partially supported by the H2020 HumaneAI Net (952026) and STRIVE - Sciences for Industrial, Green and Energy Transitions - MeSAS subproject - Models and Tools for Sustainable AI. A. Passarella’s work was partly funded by the PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000001 - ”RESTART”, both funded by the European Commission under the NextGeneration EU programme.

References

- [1] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, J. S. Rellermeyer, A survey on distributed machine learning, ACM Comput. Surv. 53 (2020). URL: <https://doi.org/10.1145/3377454>. doi:10.1145/3377454.
- [2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y. Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, in: A. Singh, J. Zhu (Eds.), Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, volume 54 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 1273–1282. URL: <https://proceedings.mlr.press/v54/mcmahan17a.html>.
- [3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, D. Bacon, Federated learning: Strategies for improving communication efficiency, arXiv preprint arXiv:1610.05492 (2016).

- [4] A. Lalitha, S. Shekhar, T. Javidi, F. Koushanfar, Fully decentralized federated learning, in: Third workshop on bayesian deep learning (NeurIPS), volume 2, 2018, pp. 1–9.
- [5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D’Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, S. Zhao, Advances and open problems in federated learning, Foundations and Trends® in Machine Learning 14 (2021) 1–210. URL: <http://dx.doi.org/10.1561/2200000083>. doi:10.1561/2200000083.
- [6] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, H. V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Transactions on Information Forensics and Security 15 (2020) 3454–3469.
- [7] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, G. Srivastava, A survey on security and privacy of federated learning, Future Gener. Comput. Syst. 115 (2021) 619–640. doi:10.1016/j.future.2020.10.007.
- [8] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, arXiv preprint arXiv:1806.00582 (2018).
- [9] L. U. Khan, S. R. Pandey, N. H. Tran, W. Saad, Z. Han, M. N. Nguyen, C. S. Hong, Federated learning for edge networks: Resource optimization and incentive mechanism, IEEE Communications Magazine 58 (2020) 88–93.
- [10] P. Singh, M. K. Singh, R. Singh, N. Singh, Federated Learning: Challenges, Methods, and Future Directions, in: Federated Learning for IoT Applications, Springer, Cham, Switzerland, 2022, pp. 199–214. doi:10.1007/978-3-030-85559-8_13.

- [11] S. Ji, Y. Tan, T. Saravirta, Z. Yang, Y. Liu, L. Vasankari, S. Pan, G. Long, A. Walid, Emerging trends in federated learning: From model fusion to federated X learning, arXiv (2021). doi:10.48550/arXiv.2102.12920, tex.eprint: 2102.12920.
- [12] Y. Jin, Y. Liu, K. Chen, Q. Yang, Federated learning without full labels: A survey, arXiv (2023). doi:10.48550/arXiv.2303.14453, tex.eprint: 2303.14453.
- [13] D. K. Dennis, T. Li, V. Smith, Heterogeneity for the Win: One-Shot Federated Clustering, 2021. URL: <http://arxiv.org/abs/2103.00697>. doi:10.48550/arXiv.2103.00697, arXiv:2103.00697 [cs].
- [14] A. Ghosh, J. Chung, D. Yin, K. Ramchandran, An efficient framework for clustered federated learning, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in neural information processing systems, volume 33, Curran Associates, Inc., 2020, pp. 19586–19597. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/e32cc80bf07915058ce90722ee17bb71-Paper.pdf.
- [15] M. Nardi, L. Valerio, A. Passarella, Anomaly detection through unsupervised federated learning, in: 2022 18th International Conference on Mobility, Sensing and Networking (MSN), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 495–501. URL: <https://doi.ieeecomputersociety.org/10.1109/MSN57253.2022.00085>. doi:10.1109/MSN57253.2022.00085.
- [16] Harshvardhan, A. Ghosh, A. Mazumdar, An Improved Algorithm for Clustered Federated Learning, 2022. URL: <http://arxiv.org/abs/2210.11538>. doi:10.48550/arXiv.2210.11538, arXiv:2210.11538 [cs, math, stat].
- [17] J. Chung, K. Lee, K. Ramchandran, Federated unsupervised clustering with generative models, in: AAAI 2022 international workshop on trustable, verifiable and auditable federated learning, volume 4, 2022, pp. 1–9.
- [18] L. Cai, N. Chen, Y. Cao, J. He, Y. Li, Fedce: Personalized federated learning method based on clustering ensembles, in: Proceedings of the 31st ACM International Conference on Multimedia,

MM '23, Association for Computing Machinery, New York, NY, USA, 2023, p. 1625–1633. URL: <https://doi.org/10.1145/3581783.3612217>. doi:10.1145/3581783.3612217.

- [19] K. R. Shahapure, C. Nicholas, Cluster quality analysis using silhouette score, in: 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), 2020, pp. 747–748. doi:10.1109/DSAA49011.2020.00096.
- [20] H. W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics 2 (1955) 83–97. doi:10.1002/nav.3800020109.
- [21] G. Cohen, S. Afshar, J. Tapson, A. van Schaik, Emnist: an extension of mnist to handwritten letters, 2017. [arXiv:1702.05373](https://arxiv.org/abs/1702.05373).
- [22] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, D. Ha, Deep Learning for Classical Japanese Literature, arXiv (2018). doi:10.20676/00000341. [arXiv:1812.01718](https://arxiv.org/abs/1812.01718).
- [23] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: International conference on machine learning, PMLR, 2016, pp. 478–487.