Federated Unsupervised Clustering with Generative Models

Jichan Chung [1], **Kangwook Lee** [2], **Kannan Ramchandran** [1]

[1] Department of EECS, University of California, Berkeley, USA
[2] Department of ECE, University of Wisconsin-Madison, Madison, USA
jichan3751@eecs.berkeley.edu, kangwook.lee@wisc.edu, kannanr@eecs.berkeley.edu

Abstract

We consider the problem of clustering unlabeled datasets in the federated environment, where statistical heterogeneity can exist across clients. Compared to the centralized setting, the model-based solution for this problem remains relatively unexplored, possibly due to increased difficulty of training models with FedAvg algorithm under highly heterogeneous setting. A recently proposed Iterative Federated Clustering Algorithm (IFCA) addresses this issue by training multiple models that captures each cluster, and showed its effectiveness on supervised datasets, for the setting when the data are i.i.d. within the same client but separated across the clients. In this work, we develop UIFCA using generative models with IFCA framework, that solves for a more general setting where the data in the same client can also come from different clusters. For synthetic data, we observe that our method can correctly recover the cluster information of individual datapoints. We also provide analysis of our method on MNIST dataset.

Introduction

Federated learning systems (McMahan et al. 2017) have become increasingly popular as they provide a way of utilizing vast computing resources and data, while preserving the individual user's privacy. FedAvg (McMahan et al. 2017) was proposed as a replacement for the batch gradient descent algorithm that works in decentralized environment, and enabled general deep neural network to be trained over large heterogeneous networks of devices such as smartphones and wearables. The algorithm was successfully deployed in wide range of tasks such as image recognition and language modeling (McMahan et al. 2017).

We focus on the problem of finding inherent (cluster) structure in user's data, in the federated learning environment. With cluster information in hand, the service provider can provide relevant information based on the data grouped by similar topics, or improve the quality of learning in downstream tasks by giving them as explicit features. For example, when a user reads news from wide range of topics, topic-specific advertisements can be placed at the time when the user reads the relavant text, and on-device training of next word prediction task can be improved by prioritizing words that are related.

Many deep generative model-based clustering methods have been proposed for the centralized environment, and have been successful in recovering useful cluster information (Mukherjee et al. 2019; Caron et al. 2018; Liu et al. 2020). To obtain high quality cluster information in a federated setting, it is natural to consider applying these methods combined with the FedAvg algorithm. However, this combination often fails due to data heterogeneity in a federated environment adversely affecting the FedAvg algorithm (Zhao et al. 2018). Due to this issue, the model-based clustering approach for the federated setting remains underdeveloped (Kairouz et al. 2021), and only the frameworks based on classic algorithms such as K-means were proposed for the data with simple cluster structure (Dennis, Li, and Smith 2021).

Recently, Iterative Federated Clustering Algorithm (IFCA) (Ghosh et al. 2020) was proposed to address a similar, but simpler problem which assumes each client holds the data from the same cluster. By alternating between training multiple models aimed at capturing each cluster and estimating user's cluster identity in a federated manner, the algorithm is proven to be able to correctly recover the cluster structure of clients, and shows promising empirical results for clustered learning of classifiers on supervised datasets.

In this work, we develop UIFCA , a new generative model-based clustering method for *unsupervised* dataset, based on the IFCA's approach. We provide a comparison of UIFCA and k-FED algorithm (Dennis, Li, and Smith 2021)(an off-the-shelf federated clustering approach), on several different types of synthetic cluster-structured data. We also evaluate our algorithm on MNIST dataset, and provide comparison against ClusterGAN (Mukherjee et al. 2019), a popular recent approach to unsupervised data in the centralized setting combined with FedAvg algorithm. For all cases, we evaluate for a particular type of client heterogeneity where a constant portion of the client's data belongs to a single distribution, and the remaining portion are drawn from the mixture of all distributions. For the synthetic datasets, we observe that our method can correctly recover cluster information for both i.i.d and non-i.i.d. cases, while the baseline fails in non-i.i.d. We also provide analysis of our method on MNIST dataset. To the best of our knowledge, this work is the first to attempt model-based clustering an unsupervised data in a heterogeneous federated environment.

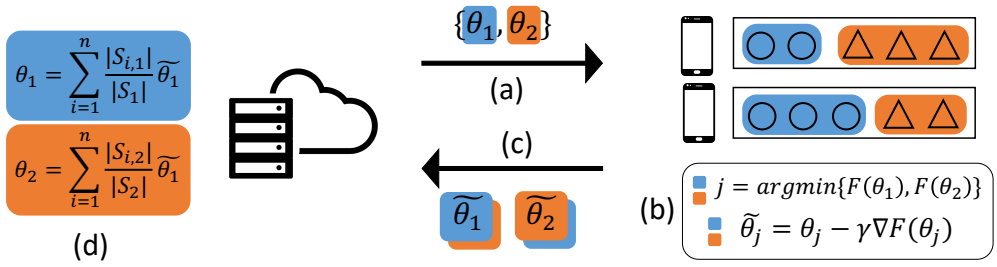


Figure 1: An overview of UIFCA . (a) Central server broadcast models to n clients. (b) Clients identify their local datapoints' cluster memberships and run local updates with received models. (c) The clients send back the local models to central server. (d) Central server aggregates the models within the same estimated cluster S_j.

Related Works

Unsupervised clustering using deep neural network for the centralized environment

Many attempts have been made in using deep neural network to unsupervised clustering problem. CusterGAN (Mukherjee et al. 2019) designs a new GAN architecture that learns representations that forms clusters in latent space. A variant of Variational Autoencoder model was proposed to capture the cluster informations (Dilokthanakul et al. 2017). Self-conditioned GAN (Liu et al. 2020) trains GAN model conditioned by pseudo-labels, where pseudo-labels are iteratively assigned from K-means clustering on the learned representation. DeepCluster (Caron et al. 2018) trains deep classifiers with similar idea. These works assumes full access to training data that captures the distribution they come from. We consider the same problem for the federated, where above assumption is not available.

Clustered federated learning for supervised task.

IFCA (Ghosh et al. 2020) and HypCluster (Mansour et al. 2020) present alternating minimization type algorithm that jointly identifies clusters in data and trains classifiers in in federated environment, as a way to tackle the issue of non-i.i.d. data distribution. The authors show good clustering performance in relatively simple settings as detecting rotations in image datasets provided with labels, but the practical efficiency in clustering without image label is not yet known. Our work extends the application of this algorithm to unlabelled image clustering task, by training generative models. (Kim et al. 2020) applied IFCA algorithm to cluster a time-series dataset in the federated environment.

Unsupervised clustering for the federated setting

For the federated environment, clustering methods for unsupervised datasets are underdeveloped. (Dennis, Li, and Smith 2021) proposes a variant of K-means algorithm (Lloyd 1982) with focus on reducing the number communication rounds. We find that methods involving DNN to cluster a unsupervised dataset are not well studied in the federated settings, therefore we analyze DNN-based methods for centralized setting in the federated settings, as well as applying IFCA algorithm to solve the same problem.

Notations

We use $[r]$ to denote the set of integers $\{1, 2, ..., r\}$.

Problem Setup

We consider a standard data clustering task in a distributed setting, where one central server communicates with n client machines. We assume that total m datapoints are inherently partitioned into K disjoint clusters, $S_1^*, \ldots, S_K^*$, and our goal is to find them. We denote j-th datapoint in client i by $x^{i,j}$. Each set S_k^* consists of $\frac{m}{K}$ datapoints coming from the distribution D_k, for $k \in [K]$. We consider unsupervised learning task where the cluster information $S_1^*, \cdots, S_K^*$ is not visible from the learning algorithm. The central server is able to communicate with client machines using predefined secure protocol, such as secure aggregation.

We aim to find an algorithm that finds clusters regardless of whether the client's data is given i.i.d. or not. To quantify the level of heterogeneity in clients, we define heterogeneity level p that measures how much the data's cluster is skewed across the machines. For example, a dataset with $K = 10$ clusters with $p = 0$ refers to the case where data are distributed to clients in purely i.i.d. manner, and in $p = 1$ case, each client holds data from a single distribution. For a client holding s datapoints, the first sp data points are sampled from a single cluster, and remaining $s(1-p)$ data points are drawn from any clusters at random. Illustrations of different p cases are given in Figure 2.

In order to estimate the cluster information of datapoints in clients, we train K different models that capture each cluster's datapoints, following the approach of IFCA (Ghosh et al. 2020). Each model is a generative model that is trained to capture the distribution of a given cluster data. Cluster membership of a datapoint can be evaluated by picking the model that gives the highest likelihood, i.e., selecting the model's distribution that it is the most close to. We define $\{\theta_1, \theta_2, ..., \theta_K\}$ as the model parameters learned for each cluster, and $f_\theta(\cdot)$ as the loss function of a sample evaluated by the model θ. Each client i assigns its each of its datapoint $x^{i,j}$ to one of the cluster sets $\{S_{i,1}, S_{i,1}, ..., S_{i,K}\}$, and runs model update on each parameter θ_k with set $S_{i,k}$ to optimize

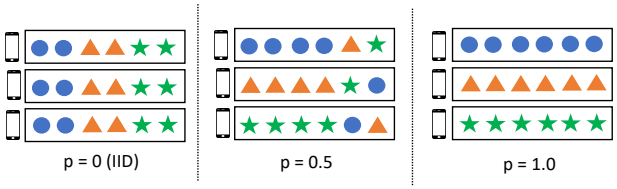


Figure 2: An illustration of a client heterogeneity type we consider.

the following local objective:

$$\min_{\theta_k} \frac{1}{|S_{i,k}|} \sum_{x^{i,j} \in S_{i,k}} f_{\theta_k}(x^{i,j}).$$

for $k \in [K]$. Then the model updates are aggregated at the central server to optimize the following global objective:

$$\min_{\theta_k} \frac{1}{|S_{*,k}|} \sum_{i=1}^{n} \sum_{x^{i,j} \in S_{i,k}} f_{\theta_k}(x^{i,j}).$$

where $S_{*,k}$ is union of $\{S_{i,k}\}_{i=1}^{n}$. This objective minimizes the loss for all the data assigned to cluster k in all clients, for each cluster $k \in [K]$. We also define $F_\theta(B) = \sum_{x \in B} f_\theta(x)$ to be the loss used in batch gradient update of a data batch B.

We note that our procedure of inferring cluster information is inherently secure, and also beneficial. A client can receive models $\{\theta_1, \theta_2, ..., \theta_K\}$ from the server and *locally* evaluate its datapoints' cluster membership. The central server can access each cluster's content by investigating the model(for example, sampling datapoints with high likelihood), and provide additional beneficial information (such as advertisements) that may be relevant for that cluster. The users have options to choose which additional information they will use, based on the evaluated cluster information, which is not revealed to the central server.

Algorithms and Models

In this section, we first present a straightforward extension of existing algorithms to a federated learning setting which we will consider as baselines, and discuss potential drawbacks of these methods. Then, we provide details of UIFCA algorithm and the models used.

Baselines

A natural approach to clustering in a federated environment is to implement a distributed version of k-means algorithm proposed by (Dennis, Li, and Smith 2021). Each worker can compute the local estimate of the centroids, and global centroids can be updated by gathering local centroids and running k-means clustering algorithm over these centroids. The client can infer the cluster membership of each datapoint by referring to local centriods' cluster assignment among all local centroids in the server. This approach is summarized in Algorithm 1. We consider this approach as a baseline to compare with UIFCA .

Algorithm 1: k-FED algorithm

1: Client i clusters local data $x^{i,j}$ into $S_{i,1}, \ldots, S_{i,K}$ based on distance to $\mu_1, \ldots, \mu_K$(for all $i \in [n]$).

$$S_{i,k} \leftarrow \{j | k = \arg\min_{k \in [K]} \|\mu_k - x^{i,j}\|\}$$

2: Client i computes local centroids $\mu_{i,1}, \ldots, \mu_{i,K}$ and send to central server.
3: Central server runs k-means algorithm over all local centroids $\mu_{i,j}$(for all $i \in [n]$, $j \in [K]$).
4: Client i assigns local data $x^{i,j}$ to a cluster according to cluster assignment of its local centroids $\mu_{i,1}, \ldots, \mu_{i,K}$ obtained from server.

For the data with more complex cluster structures such as real-world images, the k-means based approach may not work well. One may consider deep learning model-based clustering methods, which showed great success in capturing complex features of images and clustering them. The method commonly involves learning a clustering model that infers a cluster label of a sample or a representation that is well separated by the sample's inherent cluster.

In order to adapt these methods into a federated environment, a natural approach is to run distributed model training using secure training methods such as FedAvg algorithm. As a baseline for our experiments, we consider clusterGAN (Mukherjee et al. 2019) for clustering real image data, combined with FedAvg. For each communication round, each client will update the local replica of clusterGAN model with local data, and updated models from clients will be aggregated at the central server.

However, this approach is problematic when it applies to heterogeneous clients. When client's heterogenity increases, FedAvg's local objective can become different completely different from one another, resulting in increasing difficulty in learning a consensus model that performs well for all distributions (Li et al. 2020). Often, gradient averaging (or mini-batch SGD) is proposed as alternative for FedAvg for better convergence behavior in non-i.i.d setting (Yun, Rajput, and Sra 2021; Woodworth, Patel, and Srebro 2020; Woodworth et al. 2020), but we do not consider it since its practical use is not common, due to communication being the bottleneck for large models, and FedAvg enables multiple local updates within one communication round while gradient averaging only updates once.

We claim that our method has structural advantage compared to baselines in this sense. Our method captures the datapoints that are likely to be from same distribution, and runs FedAvg with them. This enables learning models in heterogeneous client data, leveraging fast local updates of FedAvg.

UIFCA

IFCA (Ghosh et al. 2020) algorithm is a clustering algorithm that clusters clients by its data using deep neural networks in a federated setting. The algorithm recovers optimal clusters by iteratively alternating between estimating cluster iden-

Algorithm 2: UIFCA

Input: Client samples $\{x^{i,*}\}_{i=1}^{n}$, number of cluster rounds T, initial cluster assignment $\{\{S_{i,k}^{0}\}_{k=1}^{K}\}_{i=1}^{n}$
For $t = 1, \cdots, T$ **do**
  (Learning) Fit generative model for each cluster
  For cluster $k \in [K]$ **in parallel do**
    $\theta_k^{(t+1)} = \texttt{LearnClusterModel}(S_*^{(t)}, S_{*,k}^{(t)})$
  (Assigning) Assign each sample to a cluster:
  For client $i \in [m]$ **in parallel do**

$$S_{i,k}^{(t)} \leftarrow \{j | k = \arg\min_{k \in [K]} f_{\theta_k^{(t)}}(x^{i,j})\}$$

Algorithm 3: $\texttt{LearnClusterModel}(S)$

Input: Data assigned to cluster k $S = (S_{1,k}, \ldots, S_{n,k})$
Choose: Number of communication rounds τ
  Initialize θ^0 randomly
  For each round $l = 1, \cdots, \tau$ **do**
    For client $i \in [n]$ **in parallel do**
      $\theta_i^{(l)} = \texttt{ModelUpdate}(\theta^{(l-1)}, S_{i,k})$
    $\theta^{(l)} = \sum_{i=1}^{m} \frac{|S_{i,k}|}{|S_{*,k}|} \theta_i^{(l)}$
Return: θ^{τ}

Algorithm 4: $\texttt{ModelUpdate}(\theta, S)$

Input: Initial parameter θ, set S
Choose: Step size η, number M of gradient steps, batch size N
For $j = 1, \cdots, M$ **do**
  $B_j \leftarrow \texttt{random\_subset}(S, N)$
  $\theta^j \leftarrow \theta^{j-1} - \eta \left(\frac{1}{N} \nabla_\theta F_{\theta^{j-1}}(B_j)\right)$
Return: θ^M

tities and optimizing the cluster models. Starting from K randomly initialized models, cluster identities of clients are found by assigning the model that gives best score (usually referring to smallest loss), and models are updated by averaging the model's SGD updates from clients within the same cluster. IFCA (Ghosh et al. 2020) was proven to be able to recover correct cluster identities under mild conditions, and was shown to be successful in simple clustering tasks such as grouping the images by rotations by training classifier models as cluster models with supervised data.

We adapt IFCA's training method to our problem to leverage its powerful clustering ability in federated settings. IFCA considers a setting where each client has data drawn i.i.d. from a single distribution, while our problem setting assumes that the data points in a client can come from different clusters. To reflect this change, our algorithm runs client local updates for all K cluster models with data assigned to each corresponding cluster, while in IFCA, each client only updates one cluster model locally. Also, in order to accomodate unsupervised data, we use a generative model as cluster parameter model, to let each model capture each cluster's data distribution.

We now discuss details of our algorithm. The algorithm is formally presented in algorithms 2 to 4 and illustrated in Figure 1.

The algorithm starts with K randomly initialized model parameters $\theta_1^{(0)}, \ldots, \theta_K^{(0)}$, and initial random cluster assignment $\{\{S_{i,k}^{(0)}\}_{k=1}^{K}\}_{i=1}^{n}$. In the t-th cluster round, the center machine broadcasts current model parameters $\theta_1^{(t)}, \ldots, \theta_K^{(t)}$ to all the machines.

For each cluster $k \in [K]$, the clients collectively runs FedAvg algorithm with model $\theta_k^(t)$ to capture the distribution of the k-th cluster's data across the clients, using the received model parameters(shown in Algorithm 3). Each client will run batch gradient update M times for each model and its corresponding cluster set (shown in Algorithm 4). These local model updates are averaged in the central server, weighted by cluster's size. After running τ times of averaging, an optimized cluster models $\theta_1^{(t+1)}, \ldots, \theta_K^{(t+1)}$ are found. These models are then broadcast to all clients. A cluster round ends up with client re-evaluating the cluster identities of local datapoint by finding model parameter with lowest loss(highest likelihood), i.e., $\text{argmin}_{k \in [K]} f_{\theta_k^{(t)}}(x^{i,j})$). The cluster rounds iterate over T times to find optimal cluster structure in the client's data.

Using normalizing flow models with UIFCA

For selecting the which generative model to use with UIFCA , we consider normalizing flow model (Tabak and Turner 2013). Normalizing flow models are generative models that model the distribution of input data, by learning an invertible function g that maps from base distribution $p_Z(z)$ to the target distribution $p_X(x)$. With base distribution $p_Z(z)$ given (commonly standard Gaussian), the likelihood of of a sample x can be found by change of variables formula:

$$p_X(x) = p_Z(g^{-1}(x)) \left| \det \frac{\partial g}{\partial x} \right|$$

The model is trained to capture the distribution by maximizing the log-likelihood of the training data with respect to the parameters of the mapping function:

$$\max_g \sum_{i=1}^{n} \log(p_X(x_i))$$

Among many options of generative models that can provide sample likelihood (such as variational autoencoders (Kingma and Welling 2014)), normalizing flow models are best fit to our needs, since it explicitly models the data distribution, it can give the most exact estimate of sample likelihood compared to generative models.

Experiments

In this section, we present our experimental results. We evaluate UIFCA with synthetic datasets and realistic image datasets based on MNIST. Our method correctly recovers

Table 1: Cluster accuracies (%) on synthetic datasets.

	p	0.0	0.25	0.5	0.75	1.0
Gaussian	UIFCA	**100**				
	k-FED	**100**				
Subspace	UIFCA	**100**				
	k-FED	11.5	11.7	12.5	15.8	19.3

clusters for synthetic settings, but does not perform well on MNIST, so we also discuss possible reasons and ways to improve it.

Synthetic experiments

We consider following two types of synthetically generated data.

Gaussian clusters Data samples with dimension $d = 32$ are generated from K Gaussian distributions with same standard deviation $\sigma = 1$ and different centers. To ensure that the clusters have less overlapping data, we generate distribution centers $\theta_k^* \sim \mathsf{Bernoulli}(0.5)$ for all $k \in [kK]$, coordinate-wise, and scale them by R. The R represents minimum separation between each center. For our experiments, we use $R = 5$ for minimal overlap.

Subspace clusters Subspace clustered data (Parsons, Haque, and Liu 2004) is a mixture of distributions that lies in different subspaces. The cluster structure is not easily discoverable using simple clustering algorithms such as k-means. For generating the data, a $d = 32$ dimensional basis set of $d_{\text{subspace}} = 16$ orthonormal basis vectors are sampled for each of k clusters, and clustered dataset is achieved by muitlplying random gaussian coefficients to each basis sets of each cluster.

For the model, we use 1-layer planar flow (Rezende and Mohamed 2016) with following linear transformation function:

$$g(z) = w^T z + b$$

with Gaussian prior $z = \mathcal{N}(0, 1)$. We initialize the models by first generating random parameters for a single model, and adding small random normal noise to each parameter. This procedure will ensure avoiding initial cluster degeneracy cases, where a particular model initializes a much smaller loss compared to other models, gets most datapoints assigned and fits data regardless of the clusters, while other models cannot learn due to the small number of data points assigned to them. We run Algorithm 2 for $T = 20$ cluster rounds, with each round consisting of $\tau = 100$ communication rounds in Algorithm 3 and $M = 100$ local batch updates. For distributed training with FedAvg, we assume that all clients participate in each communication iteration. For each type of synthetic data, we test our algorithm with $k = 4$ clusters and $n = 4$ clients with each client having 1000 datapoints, resulting $m = 4000$ datapoints in total. We report cluster accuracy, defined as $\frac{1}{m}\sum_c \max_y |S_c \cap S_y|$ which measures purity of each cluster in terms of given true label.

Clustering performance of UIFCA is reported in Table 1. As we can see, for our provided synthetic cases, UIFCA is

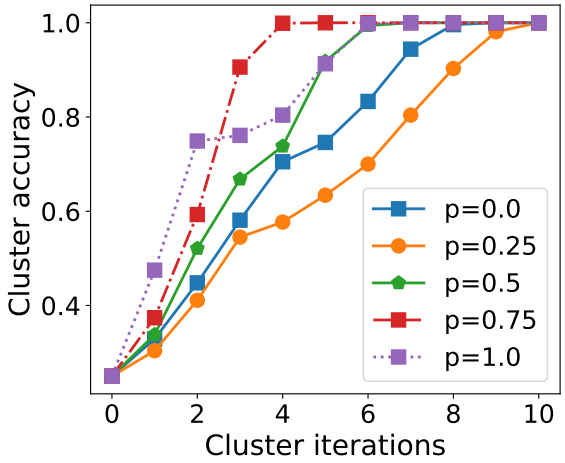


Figure 3: Cluster accuracies with respect to cluster rounds for synthetic Gaussian clustered data, for different heterogeneity levels p.

able to fully recover cluster information of individual datapoints inside clients, without compromising security assumptions of federated setting. For Gaussian clusters, each flow model learns the transformation from standard gaussian prior to each cluster distribution. For subspace clusters, the datapoints are given to follow gaussian distribution defined in different sets of basis sets, and our model learns the projection from standard basis to such subspaces. k-FED algorithm does not perform well for subspace clustered data since the data are spread out over different dimensions, distance metric of the algorithm becomes irrelevant to the cluster structure.

To provide in-depth look of UIFCA 's behavior, we plot the cluster accuracy of UIFCA with Gaussian clustered data at each cluster round in Figure 3. We can observe cluster accuracy iteratively improving over cluster rounds. Starting from random cluster assignment, each flow model captures the distribution of data assigned to its cluster. The model learns to assign high likelihood to the majority type of the data (We denote 'type' by the ground truth cluster of a data.) leading to grabbing more data of the major type and less of the other type, thus improving the cluster. Note that the cluster accuracy converges faster as client's data heterogeneity(p) increases, due to a FedAvg's weighted averaging behavior. FedAvg's local update would converge faster when the large portion of (same cluster) data points are provided in the same device, performing close to centralized SGD. On the other hand, convergence would be relatively slow when p decreases, due to instability caused by model averaging procedure.

MNIST experiments

We also test the performance of UIFCA with two types of clustered dataset based on MNIST (LeCun, Cortes, and Burges 2010) dataset. Then, we discuss limitations of our approach and a possible approach to improve.

Cluster by digits The MNIST dataset consists of $m = 60000$ 28x28 images of 10 digits with each digit having approximately 6000 images. Setting the digit information

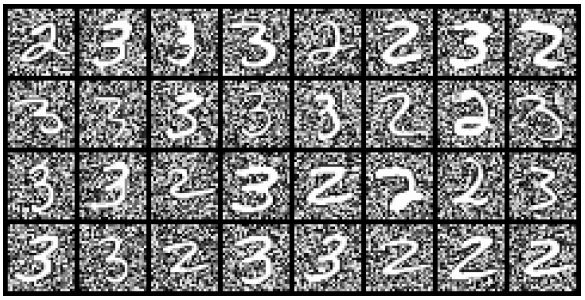


Figure 4: Randomizing the background of MNIST digit images.

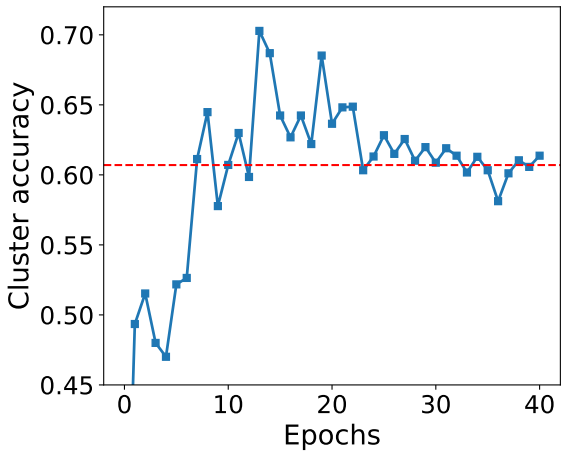


Figure 5: A sample of cluster accuracy measured at each epoch in single cluster iteration. The red horizontal line refers to the cluster accuracy of the given cluster set.

as a ground truth label for the clustering task, we evaluate unsupervised clustering performance of our algorithm with $K = 10$ clusters. We simulate the setting where the data are distributed clients according to different heterogeneity level p, same as the synthetic experiments.

Cluster by rotation To simulate data coming from different distributions, we also create a clustered dataset by applying rotations. We select one of the 10 digits and generate dataset of $k = 4$ clusters by applying $0, 90, 180, 270$ degree of rotation, resulting in unsupervised dataset of appriximately $24,000$ images. Mixing two or more digits is not considered, in order to ensure the cluster is formed by rotation, not digits. We consider digits $2, 3, 4, 5$, which are some of the digits that does not confuse rotation recognition(such as 8). We run clustering algorithm for each of the digits and report average cluster accuracy.

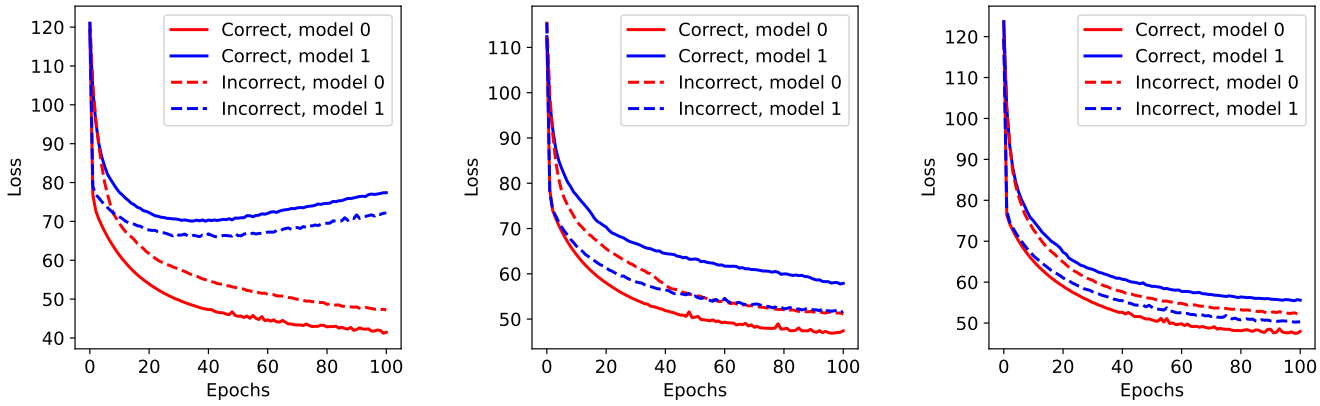
Cluster representations Since many pre-trained models are publicly available, it is often more practical to embeddings(representations) of user data for clustering rather than clustering the raw images. We test our method for clustering image representations from a pre-trained network. For the pre-trained model, we train a RotNet (Gidaris, Singh, and Komodakis 2018) that predicts image rotations, trained using rotated MNIST images in centralized settings. Each client will have access to this pre-trained model, and is able to obtain the representations of its local data from this model. For the network, we use Alexnet (Krizhevsky, Sutskever, and Hinton 2012) and extract activations from last convolutional layer.

Using initialization obtained from baseline We often find our method performing bad due to initializations. To improve this situation, we also test an additional method that starts UIFCA method with initial cluster assignment obtained from k-FED (named k-FED + UIFCA). We provide experimental result of this method for clustering representations.

Issues with RealNVP We consider RealNVP (Dinh, Sohl-Dickstein, and Bengio 2017) as the flow model to use with our algorithm, which is one of the common types of normalizing flow model targeted for images. However, our preliminary experiments showed that using standard RealNVP architecture with UIFCA cannot cluster at all. We observed that the likelihoods of the samples from cluster set used for training the model, were indistinguishible from samples outside the cluster set, which makes likelihood-based cluster assignment completely fail. We reason this failure by referring to an observation from (Kirichenko, Izmailov, and Wilson 2020), stating that a typical RealNVP model captures graphical styles rather than features. A RealNVP model typically consists of multiple stacks of affine coupling transformation function. In each function, input is split into two parts, and the function is optimized to model the transformation between the two. The typical way to split the image input is to apply a pixel-level checkerboard pattern (often called checkerboard masking) . This architecture can be good at producing realistic images, but may assign high likelihood to images outside the training set, if the graphical style matches the training set. For our task of clustering MNIST, digit images consist of similar graphical images with a large portion of black background and a pattern of strokes, thus clustering based on likelihoods would have failed.

We consider two different solutions to tackle this issue. First, we consider randomizing the background of the digit images, as shown in Figure 4. For black pixels with value less than 0.01, we randomly set pixel value from $[0, 1]$. Because the background does not have a pattern, we expect models can focus more to the strokes of the digits. Second, we apply one of the methods proposed to address this issue in (Kirichenko, Izmailov, and Wilson 2020), which changes the masking pattern of the RealNVP models from the checkerboard type to the cyclic one in each coupling function. For details, we refer to (Kirichenko, Izmailov, and Wilson 2020). We adapt their proposed architectural change to UIFCA from the author's code repository, and observe improved clustering performance compared to standard RealNVP models.

In order to select best configuration, we conduct an ablation study of these two solutions. We evaluate sample cluster assignment of two RealNVP flow models based on its likelihood, each trained with digit 2 and 3 respectively. The experiment measures the ability of distinguishing the samples in training set and the sample outside the training set,



(a) $m = 800$ (b) $m = 2400$ (c) $m = 4000$

Figure 6: Mean loss of correctly-clustered samples and misclustered samples, measured with $k = 2$ models, in single cluster iteration.

	Default	RB	CM	RB+CM
ACC(%)	50.7	95.09	**99.0**	98.6

Table 2: Ablation study of two methods: randomizing the background of digit images(RB), and changing masking pattern of in RealNVP(CM).

which is similar to Out-of-distribution detection. The results are shown in Table 2. We found changing masking pattern by itself is most effective. Randomizing background helps for the standard RealNVP model, but does not improve when the model's masking pattern is changed. Hence, we conclude that using changed masking pattern is best option for our setting.

For experiments of clustering rotations and digits of MNIST, we run Algorithm 2 for $T = 20$ cluster rounds, each having $\tau = 40$ communication rounds and $M = 100$ local updates. We use the SGD optimizer with a learning rate 1×10^{-4}, combined with the FedAvg algorithm. For clusterGAN (Mukherjee et al. 2019), we use the model imported from the author's code repository, and use SGD optimizer with learning rate 5×10^{-3} for local batch updates. We set the number of clients n to be same number as clusters k.

We report cluster accuracies for $p \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ in Table 3. For raw images(Rotated, Digits), k-FED performs overall best. As expected, clusterGAN works best at i.i.d. case ($p = 0.0$) but fails as moving towards high heterogeneity case ($p = 1.0$), due to FedAvg's bad convergence under heterogeneous environment. We observe that ours reach higher cluster accuracy for high heteogeneity level cases compared to clusterGAN, but yields overall very low cluster accuracy. For clustering representations, we find that UIFCA often works better than the k-FED approach. The k-FED + UIFCA approach gives overall best performance, showing that generative models can often find better structure in distributions than k-means based approaches, and that UIFCA can also benefit from giving good initializations, especially when the cluster accuracies are low.

One of the key reason that UIFCA perform bad is that the cluster accuracy often worsens if model training gets long. We plot a sample trace of the cluster accuracies measured at each epoch in single cluster iteration of Algorithm 2 in Figure 5. The horizontal line is the cluster accuracy of assignments from previous cluster round, which is the cluster set that the models are trained with. We can observe that cluster accuracy does not always increase and converge as training progresses. It decreases after certain point, toward original cluster accuracy. Under the hood, we observe that cluster assignments are becoming similar to assignments of previous iteration.

Main cause of this phenomenon overfitting of each cluster model to its cluster set. To see this issue in detail, we reproduce the same issue in synthetic data setting of 2 gaussian clusters. For each cluster, we define correct samples as samples that are the majority type of the cluster's set, and all other samples as incorrect. In Figure 6 we plot the mean loss of correct and incorrect samples in cluster 1, evaluated with two cluster models θ_1, θ_2, for $m = 800, 2400, 4000$ cases. For the correct sample x in cluster 1, trained the model parameters results in $f_{\theta_1}(x) < f_{\theta_2}(x)$. For the incorrect sample x', we expect $f_{\theta_1}(x') > f_{\theta_2}(x')$ so that x' can move to cluster 2 at the next cluster round. However, in Figure 6a, we observe this can happen only in early epochs. However, as the training proceeds, we observe $f_{\theta_1}(x') < f_{\theta_2}(x')$ again, due to x' becoming overfitted to model 1. The sample x' is assigned to cluster 1 again as a result, making no improvement in cluster iterations. Note that this issue gets reduced when number of data m increases, as we see Figure 6b and Figure 6c. Figure 6c shows the plot of the same configuration with $m = 4000$, and shows that two model loss $f_{\theta_1}(x), f_{\theta_2}(x)$, are not crossing, meaning that the misclustered sample will be correctly clustered at any timestep of model training.

One may consider heuristics like applying early stopping based on validation stats. We find this method often works well at synthetic settings, but fails when training with real-

Table 3: Cluster accuracies(%) of MNIST based datasets over different value of client heterogeneity level p.

	p	0.0	0.25	0.5	0.75	1.0
MNIST Rotated	UIFCA	63.5	60.8	63.4	79.5	79.6
	ClusterGAN	**95.7**	78.9	49.1	36.4	28.8
	k-FED	92.2	**87.1**	**80.5**	**78.3**	**100**
MNIST Digits	UIFCA	34.2	31.3	33.7	38.9	43.1
	ClusterGAN	**71.3**	57.4	39.4	27.5	16.8
	k-FED	57.0	**59.1**	**53.3**	**51.6**	**62.3**
MNIST Representations	k-FED	**86.0**	80.2	63.1	57.6	96.8
	UIFCA	43.6	56.4	78.2	79.5	99.5
	k-FED + UIFCA	85.1	**85.4**	**80.1**	**81.8**	**99.6**

world noisy data. The model should be trained to a level where correctly clustered data are well fit, and at the same time incorrectly clustered data are not overfit, therefore determining the optimal stopping point would be difficult without access to the ground truth cluster label. Finding the solution for this issue remains open.

Future work

Our framework has simple structure that clusters by loss given by the generative models trained with each cluster set. For the future research direction, we consider augmenting our method to involve more information. An interesting direction would be using more information than loss statistics for the cluster assignment stage, such as involving different aspects of the model (such as sample's activation on specific layer). Another way of improvement would be explicitly feeding supervision signal in training the model, such as maximizing the loss for the samples outside the cluster, which can help models in distinguishing samples inside the cluster from outside. We believe applying these methods improve our framework in model training and clustering, and paritally reduce the issue of cluster assignment converging.

Conclusions

In this paper, we address a clustering problem in a heterogeous federated learning setting where each client can have data from more than one cluster. Based on the previous work for clustered federated learning, we propose a solution that trains multiple normalizing flow models that captures each cluster, and assigns data a cluster membership by comparing the model's likelihoods. We observe that our framework correctly recovers cluster information in synthetically generated data. For real-world unsupervised dataset, we observe that our framework performs worse than baselines in some cases, and discuss possible key reason for it. For improving our framwork, involving more information in clustering and model training would be an interesting future research direction.

References

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep Clustering for Unsupervised Learning of Visual Features. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Dennis, D. K.; Li, T.; and Smith, V. 2021. Heterogeneity for the Win: One-Shot Federated Clustering. arXiv:2103.00697.

Dilokthanakul, N.; Mediano, P. A. M.; Garnelo, M.; Lee, M. C. H.; Salimbeni, H.; Arulkumaran, K.; and Shanahan, M. 2017. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. arXiv:1611.02648.

Dinh, L.; Sohl-Dickstein, J.; and Bengio, S. 2017. Density estimation using Real NVP. arXiv:1605.08803.

Ghosh, A.; Chung, J.; Yin, D.; and Ramchandran, K. 2020. An Efficient Framework for Clustered Federated Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 19586–19597. Curran Associates, Inc.

Gidaris, S.; Singh, P.; and Komodakis, N. 2018. Unsupervised Representation Learning by Predicting Image Rotations. arXiv:1803.07728.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; D'Oliveira, R. G. L.; Eichner, H.; Rouayheb, S. E.; Evans, D.; Gardner, J.; Garrett, Z.; Gascón, A.; Ghazi, B.; Gibbons, P. B.; Gruteser, M.; Harchaoui, Z.; He, C.; He, L.; Huo, Z.; Hutchinson, B.; Hsu, J.; Jaggi, M.; Javidi, T.; Joshi, G.; Khodak, M.; Konečný, J.; Korolova, A.; Koushanfar, F.; Koyejo, S.; Lepoint, T.; Liu, Y.; Mittal, P.; Mohri, M.; Nock, R.; Özgür, A.; Pagh, R.; Raykova, M.; Qi, H.; Ramage, D.; Raskar, R.; Song, D.; Song, W.; Stich, S. U.; Sun, Z.; Suresh, A. T.; Tramèr, F.; Vepakomma, P.; Wang, J.; Xiong, L.; Xu, Z.; Yang, Q.; Yu, F. X.; Yu, H.; and Zhao, S. 2021. Advances and Open Problems in Federated Learning. arXiv:1912.04977.

Kim, Y.; Hakim, E. A.; Haraldson, J.; Eriksson, H.; da Silva Jr. au2, J. M. B.; and Fischione, C. 2020. Dynamic Clustering in Federated Learning. arXiv:2012.03788.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. arXiv:1312.6114.

Kirichenko, P.; Izmailov, P.; and Wilson, A. G. 2020. Why Normalizing Flows Fail to Detect Out-of-Distribution Data. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 20578–20589. Curran Associates, Inc.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In Pereira, F.; Burges, C. J. C.; Bottou, L.; and Wein-

berger, K. Q., eds., *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

LeCun, Y.; Cortes, C.; and Burges, C. 2010. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Li, T.; Sahu, A. K.; Talwalkar, A.; and Smith, V. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine*, 37(3): 50–60.

Liu, S.; Wang, T.; Bau, D.; Zhu, J.-Y.; and Torralba, A. 2020. Diverse Image Generation via Self-Conditioned GANs. arXiv:2006.10728.

Lloyd, S. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2): 129–137.

Mansour, Y.; Mohri, M.; Ro, J.; and Suresh, A. T. 2020. Three Approaches for Personalization with Applications to Federated Learning. arXiv:2002.10619.

McMahan, H. B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. arXiv:1602.05629.

Mukherjee, S.; Asnani, H.; Lin, E.; and Kannan, S. 2019. ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 4610–4617.

Parsons, L.; Haque, E.; and Liu, H. 2004. Subspace Clustering for High Dimensional Data: A Review. *SIGKDD Explor. Newsl.*, 6(1): 90–105.

Rezende, D. J.; and Mohamed, S. 2016. Variational Inference with Normalizing Flows. arXiv:1505.05770.

Tabak, E. G.; and Turner, C. V. 2013. A Family of Nonparametric Density Estimation Algorithms. *Communications on Pure and Applied Mathematics*, 66(2): 145–164.

Woodworth, B.; Patel, K. K.; and Srebro, N. 2020. Minibatch vs Local SGD for Heterogeneous Distributed Learning. arXiv:2006.04735.

Woodworth, B.; Patel, K. K.; Stich, S. U.; Dai, Z.; Bullins, B.; McMahan, H. B.; Shamir, O.; and Srebro, N. 2020. Is Local SGD Better than Minibatch SGD? arXiv:2002.07839.

Yun, C.; Rajput, S.; and Sra, S. 2021. Minibatch vs Local SGD with Shuffling: Tight Convergence Bounds and Beyond. arXiv:2110.10342.

Zhao, Y.; Li, M.; Lai, L.; Suda, N.; Civin, D.; and Chandra, V. 2018. Federated Learning with Non-IID Data. arXiv:1806.00582.