Deep Embedded Clustering of Urban Communities Using Federated Learning

1st Afra Mashhadi  
Computing and Software Systems  
University of Washington  
Bothell, United States  
mashhadi@uw.edu

2nd Joshua Sterner  
Computing and Software Systems  
University of Washington  
Bothell, United States  
sternerj@uw.edu

3rd Jeffrey Murray  
Computing and Software Systems  
University of Washington  
Bothell, United States  
jeffmur@uw.edu

Abstract—Deep clustering utilizes representation learning to learn features in an unsupervised setting. Although successful, the current models rely on the assumption of the centralized dataset, which due to the privacy concerns is becoming less realistic. To address this challenge, we propose a federated deep convolutional embedded clustering framework. Our framework relies on a federated server to orchestrate the training between workers where each participant individually trains the model with the objective of decreasing clustering loss using Kullback–Leibler divergence. To avoid feature space being distorted by the clustering loss, each worker maintains their own local decoder which for privacy reasons is not shared with the federated server. Empirical results with both IID and non-IID client data on benchmark datasets demonstrates the feasibility of our federated training when compared to the centralized counterpart. We also evaluate our model on a real world application of community detection using GPS traces and measure the computational complexity and energy consumption on a smartphone.

I. INTRODUCTION

Unsupervised representation learning (URL) has been used for a range of applications including identifying urban communities [1], detecting cardiovascular biomarkers [2], etc. Deep embedded clustering models, such as those proposed by [3], [4], leverage unsupervised representation learning to train a deep learning network that uses the learned representation to optimize for clustering based on Kullback-Leibler divergence. However, a current assumption of the proposed models in the literature is that there is a centralized oracle repository of data that can be leveraged to train the URL models. This methodological assumption poses substantial challenges in real-world settings regarding: i) privacy concerns, where the participants who own the data are not willing to share/expose their data, making centralized training impractical and ii) heterogeneity and diversity of the participants’ data resulting in a degrading performance of models that are trained under the common assumption of independent and identically distributed (IID) data [5].

To this end, Federated Learning (FL) [6]–[9] has been proposed to provide a privacy-preserving mechanism to leverage de-centralized user data and computation resources to train machine learning models. The main idea behind federated learning is to have each node train on its own local data and not share the data or sensitive model parameters. In this vein, researchers have mostly focused on the applications of classifications where the participants hold labelled data and can locally train a supervised model for tasks such as image recognition [10]. Research in applications of URL in a federated setting is still in its infancy as we review in the related work section.

In this work, we propose a federated deep embedded clustering (F-DEC) model that relies on unsupervised representation learning (namely, autoencoder). We assume scenarios where the data is not labelled, and is Non-IID in twofold, that is the FL server (source) has a biased distribution of the data and additionally participants’ (target) data differs in size and distribution. In our proposed F-DEC model, first, the feature representation of the task at hand is learned through pre-training a convolutional auto-encoder neural network. Subsequent to this initialization step, the federated learning server embeds the model with an additional clustering layer and orchestrates rounds of training amongst the participants. At each round of training the model is locally trained with two objectives of reducing reconstruction loss and reducing the clustering loss.

The work presented here addresses how deep embedded clustering can be utilized in a federated setting and how to tune the hyper-parameters of the DEC algorithms so to increase the clustering performance on the Non-IID data in a federated setting. In particular the contributions of our paper are as follows:

- A deep embedded clustering model that is able to be co-trained in a federated setting across various devices with Non-IID data without compromising privacy of the workers.
- Empirical validation through extensive evaluation on two ML datasets, highlighting the impact of different optimization functions under various settings. We empirically show the importance of local structure preservation in the Non-IID setting.
- A use case demonstration of our model on real-world spatial-temporal data for the purpose of urban community detection.
- Empirical evaluation of our model’s performance in terms of energy and memory consumption on an ordinary smartphone.
The rest of this paper is structured as follows: In Section II, we offer a detailed background on Deep Embedded Clustering and introduce the theoretical notation of our paper. Section III examines the related literature. In Section IV, we posit our federated deep embedding clustering framework. Section V describes the pre-processing of our benchmark dataset, metrics and the results of our experimental evaluation. Section VI presents a real-world use case of urban community detection on real-world mobility traces. Finally, we conclude the paper and discuss possible future directions in Section VII.

II. BACKGROUND

A Deep Embedded Clustering network such as DEC [4] and IDEC [3] is composed of two main components, an AutoEncoder (AE) which is used in order to learn the hidden representation of the data and a clustering layer that is used to group the embedded points together.

Similar to other variations of autoencoders, a CAE is composed of an encoder part $f_\omega(\cdot)$ and a decoder $g_\omega(\cdot)$ respectively. In [11], a deep embedded clustering based on the convolutional autoencoder is proposed. A CAE aims to find a code for each input sample by minimizing the mean squared errors (MSE) between its input and output over all samples such that $x^t = g_\omega(f_\omega(x))$. To do so first some convolutional layers are stacked on the input images to extract hierarchical features. Then the output of the last convolutional layer is flattened to form a vector, followed by a fully connected layer which is called the embedded layer. The embedded layer corresponds to the latent features (also known as the code). The decoding parts of the network are the mirror construction of the described part in which some convolutional transpose layers (known as deconvolution blocks) transform embedded features back to the original image. The objective of the CAE part is to minimize the reconstruction loss denoted as $L_r$ and is measured as mean squared error

$$L_r = \frac{1}{n} \sum_{i=1}^{n} ||G_\omega(F_\omega(x_i)) - x_i||$$

where $n$ is the number of images in the dataset, and $x_i$ is the $i$th image.

In addition to reducing the reconstruction loss, the Deep Embedded Clustering networks aim to minimize a clustering loss function. This is done by creating a clustering layer connected to the embedded layer of CAE. The clustering layer maps each learned representation ($z_i$) of input image $x_i$ into a soft label. Then the clustering loss $L_c$ is defined as Kullback-Leibler divergence (KL divergence) between the distribution of soft labels and a predefined target distribution.

$$L_c = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

where $q_{ij}$ is the similarity between embedded point $z_i$ and cluster center $\mu_j$ measured by Student's t-distribution:

$$q_{ij} = \frac{\sum 1 + ||z_i - \mu_j||^2}{1 + ||z_i - \mu_j||^2}$$

And the target distribution $p_{ij}$ is formulated as:

$$p_{ij} = \frac{q_{ij}^\gamma}{\sum q_{ij}^\gamma}$$

The end to end network objective is to reduce a loss function that is a weighted sum of the reconstruction loss ($L_r$) and the clustering loss ($L_c$).

$$L = \alpha L_r + \gamma L_c$$

In DEC the authors presented the result of clustering on various datasets but did not account for the reconstruction loss in their analysis ($\alpha = 0$ and $\gamma = 1$). In contrast, IDEC [3] (Improved-DEC) and [11] demonstrated that it is important to account for both reconstruction loss and clustering loss and presented optimum results for with $\alpha$ set to 1 and $\gamma$ set to 10 (for the learning rate of 0.001). Both previous works have also reported insufficient performance for just applying kmeans clustering to the autoencode (i.e., AE+kmeans with $\alpha = 1$ and $\gamma = 0$) emphasizing that the feature representation by itself is not sufficient and it is important to optimize for the clustering loss during the propagation.

III. RELATED WORK

In recent years, few works have started to concurrently examine federated unsupervised learning. In [12], authors introduce Privacy-Preserving Unsupervised Domain Adaptation (PPUDA), which uses additive homomorphic encryption to perform unsupervised domain adaptation in a federated setting with two data holders, one in the source domain and one in the target domain. In [13] Federated Adversarial Domain Adaptation (FADA) was introduced, a technique for performing domain adaptation in a federated setting with non-IID source data. However, FADA assumes the existence of labeled source data, which is unavailable in many applications of federated learning. Zhang et al. [14] proposed Federated Contrastive Averaging (FedCA) as an approach to federated unsupervised representation learning. FedCA uses a global dictionary module which aggregates representations of data samples from each data holder. However, FedCA does not address the privacy concerns raised by sharing the global dictionary with each data holder. In [15], Federated Unsupervised Representation Learning (FURL) was introduced as a technique for performing unsupervised representation learning in a federated setting and was shown to perform effectively for human activity detection tasks. FURL performs pretraining in the federated setting, and was demonstrated using both a self-supervised learning algorithm and an autoencoder based algorithm. However, the authors do not account for clustering under the federated setting and in particular for non-IID
cases. Multi-Center Federated Learning, proposed in [16], trains multiple global models from non-IID data. During each training step, each updated local model is aggregated with the closest global model (as measured by the distance between model weights). The number of global models in Multi-Center Federated Learning is a hyperparameter. In [17], authors proposed FedFa, an algorithm for increased fairness and accuracy in FL. FedFA uses a double momentum gradient, where both the clients and the server update the model with momentum. The server uses the difference between the aggregated model and the previous global model as part of its momentum computation. FedFA also introduces a new approach to weighted aggregation that takes into account the local training accuracy and participation.

IV. FEDERATED DEEP EMBEDDED CLUSTERING

In this paper, we posit a federated deep embedded clustering model using convolutional autoencoder embedded with a clustering layer. Motivated by privacy concerns among data owners, the concept of federated learning was introduced in 2016 [6]. Federated learning allows users to collaboratively train a shared model while keeping personal data on their devices. In our system the federated server is responsible for initializing the Convolutional Auto Encoder (CAE) in a pre-training stage and orchestrating the co-training and aggregation between federated workers. That is:

1) The server pre-trains the CAE with the objective of reducing reconstruction loss over an incomplete centralized data. It then initializes the global weights, specifies the global model hyper-parameters and the training process, and sends the task to selected participants.

2) Participants locally compute training gradients by performing an end to end training with the objective of reducing $L = L_{c} + L_{r}$, and send non-sensitive updated weights to the server.

3) The server performs aggregation and shares the new weights with participants. Steps 2–3 are repeated until the global loss function converges or a desirable training accuracy is achieved.

We describe these steps in detail next.

A. Global model initialization

The global model consists of a pretrained CAE which has been trained on the partial and imbalanced centralized dataset. During the pretraining stage, the objective of the network is to reduce the reconstruction loss (Equation 1) where the weights and biases of the CAE network for each batch $m$ is updated as follows:

$$W_{cae} = W_{cae} - \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\partial L_{r}}{\partial W_{cae}}$$  

(6)

In addition to the pre-training of the CAE the federated server is responsible for two other tasks, i) initializing the clusters by performing k-means algorithm, and ii) calculating the target distribution (Equation 4).

B. On-Device Local Training

Previous works have examined possible privacy leaks that could happen based on each participant’s submitted weights [18]. For example, in [19] authors show that when training a binary gender classifier, they can infer if a certain participant’s inputs are included in the dataset just from inspecting the shared model, with an accuracy of 90%. To preserve the privacy of the devices, we propose that the federated workers do not share the weights of the decoder components which consists of three de-convolutional layers. That is during the local training, for each mini batch $m$, the weights are updated as follows:

$$W'_{d} = W_{d} - \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\partial L_{r}}{\partial W'_{d}}$$  

(7)

The encoding layers and clustering layer weights are both updated as follows:

$$W'_{e} = W_{e} - \frac{\lambda}{m} \sum_{i=1}^{m} \left( \frac{\partial L_{r}}{\partial W'_{e}} + \gamma \frac{\partial L_{c}}{\partial W_{e}} \right)$$  

(8)

At the end of the local training the local device sends the encoders and clustering weights to the federated server. By keeping a local copy of the decoder weights and parameters, the workers are able to reduce their reconstruction loss in the next iteration of training.

C. Global Model Aggregation

At the end of each round of training, the locally updated weights are aggregated. Traditionally, in the global aggregation step, the server aggregate the local models’ weights ($w_i$) from each participant $(i)$ aiming to minimize the loss function across all the participants updated weights:

$$w_{G} = \frac{1}{\sum_{i \in N} D_{i}} \sum_{i=1}^{N} D_{i} w_{i}$$  

(9)

where $D_{i}$ corresponds to the size of the samples from participant $i$, and $N$ is the total number of federated workers in the current round of training. Equation 9 is referred to as FedAvg [6] in federated learning.

V. EXPERIMENTAL EVALUATION

Our entire system is implemented in PySyft [20] and Pytorch [21] and experiments are evaluated on GPU GeForce RTX 2080 Ti with CUDA 9. Our encoder network structure is $conv_{32}^{5} \rightarrow conv_{64}^{5} \rightarrow conv_{128}^{3} \rightarrow FC_{10}$ where $conv_{k}^{n}$ denotes a convolutional layer with $n$ filters, kernel size of $k \times k$ and stride length 2 as default, and the arrow denotes the activation function. The decoding layers mirror the encoding layers. We used leaky ReLU as our activation function (negative slope of 0.01).
A. Metrics

We use two common unsupervised evaluation metrics [22] for evaluating our model that enable us to draw direct comparison with DEC and IDEC models using same verification datasets. In particular, we measured Unsupervised Clustering Accuracy (ACC) as:

\[
ACC = \max_m \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\{y_i = m(c_i)\}
\]

where \(y_i\) is the ground-truth label, \(c_i\) is the cluster assignment generated by our algorithm, and \(m\) is a mapping function which ranges over all possible one-to-one mappings between assignments and labels. The second one is Normalized Mutual Information (NMI):

\[
NMI(Y, C) = \frac{I(Y, C)}{\frac{1}{2}[H(Y) + H(C)]}
\]

where \(Y\) denotes the ground-truth labels, \(C\) denotes the clusters assignment labels, \(I\) is the mutual information metric and \(H\) is entropy.

B. Datasets

Training. We evaluate our model using MNIST and USPS datasets as benchmarks in DEC and IDEC papers. The MNIST-train dataset consists of total 70000 handwritten digits of 28x28 pixels. The USPS-train dataset contains 9298 gray-scale handwritten digit images with size of 16x16 pixels. In order to study the effect of imbalanced data, we partition the training datasets into a split, where we use a segment of the dataset for pre-training the CAE network on the FL server and the remaining of the data distributed amongst the federated workers.

For the centralized pre-training phase, we assume that the training dataset is incomplete and imbalanced. That is the federated server can only learn a partial representation of the feature space \(x\) and labels \(y\). In so doing, we follow a linear partitioning as introduced by [4], where the percentage of feature space \(x\) and labels \(y\). In doing so, we follow a linear sampling rate. That is for a given minimum retention rate \((r_{\text{min}})\), data points of digit 0 will be kept with probability of \(r_{\text{min}}\) and class 9 with a probability of 1, with the other classes linearly in between. We then distribute the remaining of the datasets across the clients following two different distribution strategies:

- **Independent and Identical Distribution (IID):** where there is a shared feature space \((x)\) and label \((y)\) and an equal distribution of the number of images per each class (handwritten digits) across the devices.
- **Non-identical client distributions:** Specifically we target the case of skew in the label distribution where the marginal distributions \(P_i(y)\) may vary across clients, even if \(P(x|y)\) is the same across different clients for features \(x\) and label \(y\). This type of non-IID could in particular happen for example, when the distribution of labels varies across clients or where certain labels are used by one demographic but not others [8]. We distribute the data amongst the clients such that each device holds data from one class only (i.e., on handwritten digit).

Validation. Finally, we use MNIST-test (10k images, 12% of overall MNIST dataset) and USPS-test (3k images, 24% of overall USPS) datasets for verification in order to compute the described metrics. The validation is performed at the federated server after the local models are aggregated at the end of each round of training.

C. Results

We pre-trained the CAE component of the model on the described datasets. For the pre-training phase we train the encoder for 300 epochs and use Adam [23] with default parameters. For the following analysis we used networks that are pre-trained on 10% of the training dataset with \(r_{\text{min}} = 0.1\) for the linear partitioning. Figure 1 presents a sample image reconstruction after the pre-training for imbalanced MNIST and USPS dataset respectively, and presents the reconstruction loss \((L_r)\) during the pre-training stage.

1) **Centralized Benchmark:** In order to derive a benchmark comparison with the centralized deep clustering algorithms, we first report the performance of the pre-trained IDEC and DEC model when adjusted for convolutional encoders [11] on the centralized setting, where pre-training phase is done on the imbalanced dataset (10% of the original dataset with \(r_{\text{min}} = 0.1\) linear split) as described above. Table I presents the accuracy and NMI (in brackets) for IDEC [3] (with \(\gamma = 10\)), CAE+K-means (i.e., \(\gamma = 0\)) and DEC [4] for full versus imbalanced pre-training dataset. As it can be seen the performance of these algorithms drop dramatically when the assumption of balanced dataset is removed.

Figure 2 demonstrates the impact of \(\gamma\) for experiments with various learning rate \((\lambda)\), where DEC is denoted as dashed lines and IDEC as solid lines. In contrast with the previous reporting of the coupling between the \(\gamma\) and \(\lambda\) (where lower \(\lambda\) achieved better performance when paired with higher \(\gamma\)), we observe that the higher values of \(\gamma\) degrade the IDEC improved performance by causing a more significant distortion of the latent space achieving similar results to DEC. The best performance is achieved when the model trained with the smallest learning rate and for smaller values of \(\gamma\) indicating the necessity in optimizing for reconstruction as opposed to clustering. Figure 3 illustrates the clustering and reconstruction loss for IDEC (with \(\lambda = 0.0001\)) and different values of \(\gamma\). We can observe from this figure that as the value of \(\gamma\) grows, that is more weight is given to the clustering loss, the feature space gets distorted (i.e., higher reconstruction loss). We observe similar results for USPS-imbalance dataset which are omitted due to lack of space.

2) **F-DEC Performance on IID:** In this section, we examine the performance of our model when applied in IID setting where all the devices have equal and uniformly distributed number of images from each class. We measure the clustering metrics for a varying number of federated workers. Figure 4 illustrates the verification NMI computed after global aggregation (FedAvg) at the end of every training round for
<table>
<thead>
<tr>
<th>Methods</th>
<th>MNIST-full</th>
<th>MNIST-imbalanced</th>
<th>USPS-full</th>
<th>USPS-imbalanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>88.63 (87.59)</td>
<td>45.26 (44.25)</td>
<td>77.90 (81.08)</td>
<td>44.59 (44.92)</td>
</tr>
<tr>
<td>CAE+K-means</td>
<td>84.90 (79.27)</td>
<td><strong>60.93 (52.58)</strong></td>
<td>74.15 (73.30)</td>
<td><strong>53.21 (44.98)</strong></td>
</tr>
<tr>
<td>IDEC (γ = 10)</td>
<td><strong>88.97 (88.49)</strong></td>
<td>45.78 (44.56)</td>
<td><strong>79.00 (82.57)</strong></td>
<td>48.17 (41.74)</td>
</tr>
</tbody>
</table>

**TABLE I**

COMPARISON OF CLUSTERING PERFORMANCE IN TERMS OF ACCURACY (%) AND NMI (% IN BRACKET).

\[ \gamma = 10 \]. As it can be seen the smaller the number of the workers, the faster the global model converges. This is due to the larger size of the local dataset at each worker that is used for local training and thus after each round of local training the aggregated global model improves in accuracy drastically. We also measure the training time on the reported setup to take on average 70 seconds per epoch of local training for MNIST and 12 seconds for USPS dataset.

Figure 5 illustrates the accuracy of the global model for each worker at the beginning, middle and the end of the training for 20 and 10 workers. As it can be seen the model performs equally for everyone (due to the IID assumption) during the training. After 50 training rounds, we observe a greater shift on the x-axis (accuracy) for the experiments with 10 devices versus the one with 20 devices.

Next we examine the impact of \( \gamma \) on the performance of F-DEC for varying number of workers under IID setting. Figure 6 illustrates this results. As it can be seen for higher value of \( \gamma \) the performance improves as the devices share an identical distribution of the data and therefore prioritizing KL divergence allows the model to reduce the clustering loss quicker. Indeed, we observe a coupling effect between \( \gamma \) and the number of participants, that is as the number of participants increase the clustering loss decays slower and later in the training. An impact of this coupling is observed in experiments with 50 workers where the clustering loss only starts decreasing at 100 epochs, causing the poor performance that is reported in the Figure 6. Similar observations have been made on the USPS dataset but omitted due to space.

3) F-DEC Performance on Non-IID: Under Non-IID setting we observe an opposite effect in regards with the clustering weight (Figure 7). As \( \gamma \) increases the performance of the model decays due to the disparity that exists between the representation feature space of local data at each device. Indeed the lowest performance is observed when the reconstruction loss is ignored (dotted lines) and the devices only optimize for soft label assignment. Figure 7 also reports of the interplay between the number of workers and the performance. In contrast to the IID setting we see that the larger number of participants not only is correlated with better performance but also with less distortion effect of greater \( \gamma \) values. This is due to the increased overlap of feature space amongst the workers as the number of workers grows. That is when K=10, data from each digit is carried by only 1 worker versus 5 workers when K=50. Table II reports the accuracy and NMI (in brackets) for F-DEC with \( \gamma = 0.01 \).

Finally to understand the performance of the global model for each individual worker, Figure 8 illustrates the density accuracy plot for 50 federated workers (MNIST). Although the global model with lowest clustering weight (\( \gamma = 0 \)) has the highest performance on the verification dataset, we can see that...
Fig. 3. Reconstruction loss (right) and Clustering loss (left) for various values of $\gamma$ on MNIST-imbalanced centralized IDEC.

Fig. 4. The performance of F-DEC for MNIST under IID setting.

<table>
<thead>
<tr>
<th>No. Devices</th>
<th>MNIST-imbalanced</th>
<th>USPS-imbalanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>50.02 (42.49)</td>
<td>59.19 (47.34)</td>
</tr>
<tr>
<td>20</td>
<td>52.96 (44.76)</td>
<td>64.22 (50.89)</td>
</tr>
<tr>
<td>50</td>
<td>56.80 (48.46)</td>
<td>67.66 (54.44)</td>
</tr>
</tbody>
</table>

TABLE II

The Performance of F-DEC in non-IID Setting.

it performs very unevenly across the participants. Indeed we see that as $\gamma$ increases the distribution of accuracy per worker becomes closer to the mean, illustrating fairness amongst the participants.

D. Performance on Mobile Device

In order to evaluate the viability of our approach in real-world setting, we tested the performance of our model on the LG G7 ThinQ Smartphone with an 8-core processor with four 1.8 GHz cores and four 2.8 GHz cores. We train an instance of F-DEC on a device with 100 images from MNIST. Our measurements show that each epoch of local training takes on average 35 seconds and the battery usage is 15 mAh, with a peak memory usage under 225 MB, and a size of 1.1MB for saved models.

VI. Real-world Use Case: Urban Community Detection

We investigate the performance of our model for identifying urban communities based on spatio-temporal visitation patterns using the GeoLife [24] dataset. The GeoLife trace records mobility in various scenarios in the city of Beijing, including different modes of transportation (e.g., walking, cycling, and driving). The trace contains GPS trajectories of 182 users, collected over three years and sampled every five seconds (overall 17,621 trajectories, over 50K hours).

Data Processing. In transferring the location based data to heatmap images we follow the methodology suggested by [1]. The raw traces contain GPS trajectories for each user, represented in the dataset by a sequence of latitude and longitude set of coordinates over time ($T$). The first step in our approach was to split the raw traces by month for each user, such that each user contained their entire individual temporal resolution. The second step in our implementation was to fairly represent each user and their individual mobility patterns by a static bounding box within the city limits of Beijing, China. With all of our data pre-processed by month and only includes data points within the bounded region, we can now feed the trajectory data into our Frequency Matrix (FM).

With the static bounding box region, we calculated the width, $w$, and length, $l$, of the outer rectangle in miles using the Haversine Formula. Based on the input cell size $C$, the pixels represent each cell (in miles) within the total width or length of the bounding box. Such that our Frequency Matrix is defined as:

$$ FM = \begin{bmatrix} C_{0,0} & C_{0,1} & \cdots & C_{0,w} \\ C_{1,0} & C_{1,1} & \cdots & C_{1,w} \\ \vdots & \vdots & \ddots & \vdots \\ C_{l,0} & C_{l,1} & \cdots & C_{l,w} \end{bmatrix} $$ (12)

To generate images, we normalized each value at FM(i, j) between 0 and 1 with the natural logarithm with a base of the max value present in the FM. We represent this frequency in terms of pixel intensity where the pixel intensity of $FM(i,j)$ is represented with respect to the maximum value within the FM. For the purpose of pre-training we set $T$ to be the full duration of the traces such that each user only has 1 heat map image capturing their spatial preferences. For the federated training we set $T$ to monthly granularity, resulting in 60 images per each user. This distribution of the target and source dataset reflects the real-world setting where a centralized server may have some partial aggregated knowledge of people’s movement in the city (e.g., traces from transportation datasets), and the users maintain a full record of their own trajectory and mobility.
Fig. 5. Density plot of accuracy per worker at the beginning, middle and end of training for 20 and 10 workers respectively.

Fig. 6. The performance of F-Dec on IID setting for MNIST, for varying number of workers (K) and clustering weight $\gamma$.

Fig. 7. The performance of F-DEC on Non-IID setting for MNIST.

Fig. 8. Density plot of 50 workers’ accuracy for various $\gamma$.

Fig. 9. Structural Similarity between the heatmap images of users in GeoLife dataset.

The images that were produced as an output of the described transformation are $15 \times 23 = 345$ pixels, which corresponds to an input cell size of 0.5 miles within the bounding box boundary of Beijing. The resulting dataset is 182 images for pre-training and a ranging collection between 1 to 60 images for each user (local dataset). We pre-train the model using 100 epochs with $\lambda = 0.001$ and evaluate the performance of the model for varying number of communities ($k$).

**Metric.** In order to measure the accuracy of the clusters we calculate the in-cluster structural similarity, SSIM [25] and out-cluster SSIM pairwise after each round of training. That is we expect the users that belong to the same community to have a higher in-cluster SSIM and lower out-cluster SSIM on average. Unlike Mean Square Error (MSE, the SSIM metric has been shown not to be significantly impacted by the changes in luminosity and contrast. Given that images representing mobility features extracted from mobility traces use pixel intensity to encode the frequency of the visit spent in a given area, the SSIM is a well suited metric to compute the image similarity. Figure 9 presents the SSIM matrix between every user in the GeoLife dataset ($\mu = 0.07, \sigma = 0.12$), the brighter pixels present higher structural similarity of the images.

**Communities.** Previous work [1] have reported 8 urban communities in the city of Beijing using GeoLife traces. Our results also echo that observation as we find the communities between 6 to 8 to have highest in-cluster communities and lowest out-cluster similarity. Table III presents our results for the federated training and the benchmark comparison based
<table>
<thead>
<tr>
<th>Communities</th>
<th>SSIM In-cluster (out-cluster)</th>
<th>Federated (182 workers)</th>
<th>Centralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.104 (0.044)</td>
<td>0.111 (0.044)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.107 (0.045)</td>
<td>0.112 (0.040)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.114 (0.046)</td>
<td>0.149 (0.060)</td>
<td></td>
</tr>
</tbody>
</table>

**TABLE III**

**PERFORMANCE COMPARISON USING SSIM IN AND OUT-CLUSTER FOR FEDERATED AND CENTRALIZED MODEL.**

on the centralized implementation as described in [1].

**VII. CONCLUSION**

In this paper we presented the empirical results of deep embedded clustering in the federated setting. Our framework leverages co-training and orchestrates on-device local training that optimises for clustering and reconstruction loss simultaneously. We empirically showed the impact of heterogeneous and imbalanced data both at the pre-training stage and on the federated workers (non-IID setting). Our results report the interplay between the clustering weight and the number of workers and their impact on both clustering performance and fairness. We presented the viability of our model when it is computed on the mobile devices and presented a use case of urban community detection using real mobility traces. It is worth noting some limitations of systems such as ours in offering fair and equitable approaches that can be used in urban planning. Indeed, fairness and understanding who is represented in such systems is important to ensure that a subsection of population are not excluded due to participation barriers (be it cultural or resources limitation). Our future work will focus on validating model’s fairness criteria (in addition to the accuracy) in the FL server to ensure every round of training is diversifying the model to work equally well for all the participants.

**REFERENCES**


