Federated t-SNE and UMAP for Distributed Data Visualization

Anonymous submission

Abstract

High-dimensional data visualization is crucial in big data era and these techniques such as t-SNE and UMAP have been widely used in science and engineering. Big data, however, is often distributed across multiple data centers and subject to security and privacy concerns, which leads to difficulties for the standard algorithms of t-SNE and UMAP. To tackle the challenge, this work proposes Fed-tSNE and Fed-UMAP, which provide high-dimensional data visualization under the framework of federated learning, without exchanging data across clients or sending data to the central server. The main idea of Fed-tSNE and Fed-UMAP is implicitly learning the distribution information of data in a manner of federated learning and then estimating the global distance matrix for t-SNE and UMAP. To further enhance the protection of data privacy, we propose Fed-tSNE+ and Fed-UMAP+. We also extend our idea to federated spectral clustering, yielding algorithms of clustering distributed data. In addition to these new algorithms, we offer theoretical guarantees of distance and similarity estimation and analyze the property of differential privacy. Experiments on multiple datasets demonstrate that, compared to the original algorithms, the accuracy drops of our federated algorithms are tiny.

1 Introduction

High-dimensional data are prevalent in science and engineering and their structures are often very complicated, which makes dimensionality reduction and data visualization appealing in knowledge discovery and decision-making (Jolliffe and Cadima 2016; Hinton and Salakhutdinov 2006; Van Der Maaten et al. 2009). In the past decades, many algorithms have been proposed for dimensionality and visualization (Pearson 1901; Fisher 1936; Sammon 1969; Baker 1977; Kohonen 1982; Schölkopf, Smola, and Müller 1998; Roweis and Saul 2000; Tenenbaum, De Silva, and Langford 2000; Van der Maaten and Hinton 2008; McInnes et al. 2018). Perhaps, the most popular algorithms in recent years are the t-distributed stochastic neighbor embedding (t-SNE) developed by (Van der Maaten and Hinton 2008) and the Uniform Manifold Approximation and Projection (UMAP) proposed by (McInnes et al. 2018). T-SNE and UMAP map the data points to a two- or three-dimensional space, exhibiting the intrinsic data distribution or pattern of the original high-dimensional data. Due to their superiority over other methods such as PCA (Jolliffe and Cadima 2016), Isomap (Tenenbaum, De Silva, and Langford 2000), and autoencoder (Hinton and Salakhutdinov 2006), they have been used for visualizing images, tabular data (Hao et al. 2021), text (Grootendorst 2022), and graphs (Wu, Zhang, and Fan 2023) in diverse fields and provide huge convenience for scientific research and engineering practice (Becht et al. 2019). Besides visualization, t-SNE and UMAP are also useful in clustering (Linderman and Steinerberger 2019) and outlier detection (Fu, Zhang, and Fan 2024). There are also a few variants of t-SNE (Yang et al. 2009; Carreira-Perpinán 2010; Xie et al. 2011; Van Der Maaten 2014; Gisbrecht, Schulz, and Hammer 2015; Pezzotti et al. 2016; Linderman et al. 2019; Chatzimparmpas, Martins, and Kerren 2020; Sun, Han, and Fan 2023) and UMAP (Sainburg, McInnes, and Gentner 2021; Nolet et al. 2021). For instance, Van Der Maaten (2014) used tree-based algorithms to accelerate the implementation of t-SNE. Sainburg, McInnes, and Gentner (2021) proposed a parametric UMAP that can visualize new data without re-training the model.

In many real cases such as mobile devices, IoT networks, medical records, and social media platforms, the high-dimensional data are distributed across multiple data centers and subject to security and privacy concerns (Dwork, Roth et al. 2014; McMahan et al. 2017; Kairouz et al. 2021), which leads to difficulties for the standard algorithms of t-SNE and UMAP. Specifically, in t-SNE and UMAP, we need to compute the pair-wise distance or similarity between all data points, meaning that different data centers or clients should share their data mutually or send their data to a common central server, which will leak data privacy and lose information security. To address this challenge, we propose federated t-SNE and federated UMAP in this work. Our main idea is implicitly learning the distribution information of data in a manner of federated learning and then estimating the global distance matrix for t-SNE and UMAP. The contribution of this work is summarized as follows:

- We propose Fed-tSNE and Fed-UMAP that are able to visualize distributed data of high-dimension.
- We further provide Fed-tSNE+ and Fed-UMAP+ to enhance privacy protection.
- We extend our idea to federated spectral clustering for distributed data with privacy protection.
- We provide theoretical guarantees such as reconstruction

error bounds and differential privacy analysis.

2 Related work

t-SNE t-SNE (Van der Maaten and Hinton 2008) aims to preserve the pair-wise similarities from high-dimension space $\mathcal{P}$ to low-dimension space $\mathcal{Q}$. The pair-wise similarities are measured as the probability that two data points are neighbors mutually. Specifically, given high-dimensional data points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ in $\mathbb{R}^D$, t-SNE computes the joint probability matrix $\mathbf{P} \in \mathbb{R}^{N\times N}$, in which $p_{ij} = 0$ if $i = j$, and $p_{ij} = \frac{p_{i|j}+p_{j|i}}{2N}$, if $i \neq j$, where

$$p_{j|i} = \frac{\exp\left(-\|\mathbf{x}_i-\mathbf{x}_j\|_2^2/(2\tau_i^2)\right)}{\sum_{\ell\in[N]\setminus\{i\}}\exp\left(-\|\mathbf{x}_i-\mathbf{x}_\ell\|_2^2/(2\tau_i^2)\right)}. \qquad (1)$$

In (1), τ_i is the bandwidth of the Gaussian kernel. Suppose $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ are the low-dimensional embeddings in $\mathbb{R}^d$, where $d \ll D$, t-SNE constructs a probability matrix $\mathbf{Q}$ by

$$q_{ij} = \frac{\left(1+\|\mathbf{y}_i-\mathbf{y}_j\|_2^2\right)^{-1}}{\sum_{\ell,s\in[N],\ell\neq s}\left(1+\|\mathbf{y}_\ell-\mathbf{y}_s\|_2^2\right)^{-1}} \qquad (2)$$

where $i \neq j$. Then t-SNE obtain $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ by minimizing the Kullback-Leibler (KL) divergence

$$\underset{\mathbf{y}_1,\ldots,\mathbf{y}_N}{\text{minimize}} \sum_{i\neq j} q_{ij} \log \frac{p_{ij}}{q_{ij}} \qquad (3)$$

UMAP UMAP (McInnes et al. 2018) is a little similar to t-SNE. It starts by constructing a weighted k-NN graph in the high-dimensional space. The edge weights between points $\mathbf{x}_i$ and $\mathbf{x}_j$ are defined based on a fuzzy set membership, representing the probability that $\mathbf{x}_j$ is in the neighborhood of $\mathbf{x}_i$. Specifically, the membership strength is computed using

$$\mu_{i|j} = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_i\right), \qquad (4)$$

where σ_i is a local scaling factor determined by the k-NNs of $\mathbf{x}_i$. The final membership strength is symmetrized as

$$\mu_{ij} = \mu_{i|j} + \mu_{j|i} - \mu_{i|j}\cdot\mu_{j|i} \qquad (5)$$

In the low-dimensional space, the probability of two points being neighbors is modeled using a smooth, differentiable approximation to a fuzzy set membership function. The edge weights between points $\mathbf{y}_i$ and $\mathbf{y}_j$ are given by

$$\mu'_{ij} = \frac{1}{1+a\|\mathbf{y}_i-\mathbf{y}_j\|^{2b}} \qquad (6)$$

where a and b are hyperparameters typically set based on empirical data to control the spread of points in the low-dimensional space. UMAP minimizes the cross-entropy between the high-dimensional fuzzy simplicial set and the low-dimensional fuzzy simplicial set, i.e.,

$$\underset{\mathbf{y}_1,\ldots,\mathbf{y}_N}{\text{minimize}} \sum_{i\neq j} \mu_{ij}\log\left(\frac{\mu_{ij}}{\mu'_{ij}}\right) + (1-\mu_{ij})\log\left(\frac{1-\mu_{ij}}{1-\mu'_{ij}}\right) \quad (7)$$

Discussion Studies about federated dimensionality reduction or data visualization are scarce in the literature. Grammenos et al. (2020) proposed a federated, asynchronous, and (ϵ, δ)-differentially private algorithm for PCA in the memory-limited setting. Briguglio et al. (2023) developed a federated supervised PCA for supervised learning. Novoa-Paradela, Fontenla-Romero, and Guijarro-Berdiñas (2023) proposed a privacy-preserving training algorithm for deep autoencoders. Different from PCA and autoencoders, in t-SNE and UMAP, we need to compute the pair-wise distance or similarity between data points, which leads to significantly higher difficulty in developing federated learning algorithms. Saha et al. (2022) proposed a decentralized data stochastic neighbor embedding, dSNE. However, dSNE assumes that there is a shared subset of data among different clients, which may not hold in real applications.

3 Federated Distribution Learning

3.1 Framework

Suppose data $\boldsymbol{X} = \{\boldsymbol{X}_p\}_{p=1}^P$ are distributed at P clients, where $\boldsymbol{X}_p \in \mathbb{R}^{m\times n_p}$ belongs to client p and $\sum_{p=1}^P n_p = n_x$. To implement t-SNE and UMAP, we need to compute a matrix $\boldsymbol{D}_{\boldsymbol{X},\boldsymbol{X}} \in \mathbb{R}^{n_x\times n_x}$ of distances between all data pairs in $\boldsymbol{X}$, which requires data sharing between the clients and central server, leading to data or privacy leaks. We propose to find an estimate of the distance or similarity matrix without data sharing. To do this, we let the central server construct a set of intermediate data points denoted by $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_y}] \in \mathbb{R}^{m\times n_y}$ and then compute distance matrices $\boldsymbol{D}_{\boldsymbol{Y},\boldsymbol{Y}}$ and $\{\boldsymbol{D}_{\boldsymbol{X}_p,\boldsymbol{Y}}\}_{p=1}^P$. These distance matrices can be used to construct an estimate $\widehat{\boldsymbol{D}}_{\boldsymbol{X},\boldsymbol{X}}$ of $\boldsymbol{D}_{\boldsymbol{X},\boldsymbol{X}}$ by applying the Nytröm method (Williams and Seeger 2001) (to be detailed later). However, the choice of $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_y}] \in \mathbb{R}^{m\times n_y}$ affects the accuracy of $\widehat{\boldsymbol{D}}_{\boldsymbol{X},\boldsymbol{X}}$, further influencing the performance of t-SNE and UMAP.

Since Nytröm method (Williams and Seeger 2001) aims to estimate an entire matrix using its small sub-matrices, the sub-matrices should preserve the key information of the entire matrix, which means a good $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_y}] \in \mathbb{R}^{m\times n_y}$ should capture the distribution information of $\boldsymbol{X}$. Therefore, we propose to learn such a $\boldsymbol{Y}$ adaptively from the P clients via solving the following federated distribution learning (FedDL) framework:

$$\underset{\boldsymbol{Y}}{\text{minimize}}\ F(\boldsymbol{Y}) \triangleq \sum_{p=1}^P \omega_p f_p(\boldsymbol{Y}) \qquad (8)$$

where f_p is the local objective function for each client, and $\omega_1, \ldots, \omega_P$ are nonnegative weights for the clients. WLOG, we set $\omega_1 = \cdots = \omega_P = 1/P$ for convenience in the remaining context. In this work, we set f_p to be the Maximum Mean Discrepancy (MMD) (Gretton et al. 2012) metric:

$$\begin{aligned} f_p(\boldsymbol{Y}) &= \mathrm{MMD}(\boldsymbol{X}_p, \boldsymbol{Y}) \\ &= \frac{1}{n_p(n_p-1)}\sum_{i=1}^{n_p}\sum_{j\neq i}^{n_p} k\left((\boldsymbol{X}_p)_{:,i}, (\boldsymbol{X}_p)_{:,j}\right) \\ &\quad - \frac{2}{n_p n_y}\sum_{i=1}^{n_p}\sum_{j=1}^{n_y} k\left((\boldsymbol{X}_p)_{:,i}, (\boldsymbol{Y})_{:,j}\right) \\ &\quad + \frac{1}{n_y(n_y-1)}\sum_{i=1}^{n_y}\sum_{j\neq i}^{n_y} k\left((\boldsymbol{Y})_{:,i}, (\boldsymbol{Y})_{:,j}\right) \end{aligned} \qquad (9)$$

or in the following compact form

$$\begin{aligned} f_p(\boldsymbol{Y}) &= \text{MMD}(\boldsymbol{X}_p, \boldsymbol{Y}) \\ &= \tfrac{1}{n_p(n_p-1)} \left[\mathbf{1}_{n_p}^T \boldsymbol{K}_{\boldsymbol{X}_p,\boldsymbol{X}_p} \mathbf{1}_{n_p} - n_p\right] - \tfrac{2}{n_p n_y} \mathbf{1}_{n_p}^T \boldsymbol{K}_{\boldsymbol{X}_p,\boldsymbol{Y}} \mathbf{1}_{n_y} \\ &\quad + \tfrac{1}{n_y(n_y-1)} \left[\mathbf{1}_{n_y}^T \boldsymbol{K}_{\boldsymbol{Y},\boldsymbol{Y}} \mathbf{1}_{n_y} - n_y\right] \end{aligned} \tag{10}$$

where $k(\cdot,\cdot)$ is the kernel function and $\boldsymbol{K}_{\cdot,\cdot}$ denotes the kernel matrix computed from two matrices. MMD is a distance metric between two distributions and (10) is actually an estimation of MMD with finite samples from two distributions. If we use the Gaussian kernel $k(\boldsymbol{x}_i, \boldsymbol{y}_j) = \exp(-\gamma\|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2)$, MMD compares all-order statistics between two distributions. For any $\boldsymbol{X} \in \mathbb{R}^{m\times n_x}$ and $\boldsymbol{Y} \in \mathbb{R}^{m\times n_y}$, we calculate the Gaussian kernel matrix as $\boldsymbol{K}_{\boldsymbol{X},\boldsymbol{Y}} = \exp(-\gamma \boldsymbol{D}^2)$, where $\boldsymbol{D}^2$ is the squared pairwise distance matrix between $\boldsymbol{X}$ and $\boldsymbol{Y}$, i.e., $\boldsymbol{D}^2 = \text{Diag}(\boldsymbol{X}^T\boldsymbol{X})\mathbf{1}_{n_y}^T - 2\boldsymbol{X}^T\boldsymbol{Y} + \mathbf{1}_{n_x}\text{Diag}(\boldsymbol{Y}^T\boldsymbol{Y})^T$.

Combining (8) and (10), we have the following optimization problem of federated distribution learning

$$\underset{\boldsymbol{Y}}{\text{minimize}} \sum_{p=1}^{P} \omega_p \times \text{MMD}(\boldsymbol{X}_p, \boldsymbol{Y}) \tag{11}$$

By solving this problem, the central server or $\boldsymbol{Y}$ equivalently can learn the distribution information of the data distributed on the P clients. Based on such an $\boldsymbol{Y}$, we can estimate the distance or similarity matrix between all data points in $\boldsymbol{X}$, which will be detailed later.

```
Algorithm 1: Federated Distribution Learning
Require: Distributed data $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_P\}$ at $P$ clients.
 1: Server broadcast an initial $\boldsymbol{Y}^0$ to all clients.
 2: for round $s = 1$ to $S$ do
 3:     Client side:
 4:     for client $p = 1$ to $P$ in parallel do
 5:         Set $\boldsymbol{Y}_p^{s,0} = \boldsymbol{Y}^{s-1}$
 6:         Update local variable $\boldsymbol{Y}_p^s$:
 7:         for $t = 1$ to $Q$ do
 8:             $\boldsymbol{Y}_p^{s,t} = \boldsymbol{Y}_p^{s,t-1} - \eta_s \nabla f_p(\boldsymbol{Y}_p^{s,t-1})$
 9:         end for
10:         Denote $\boldsymbol{Y}_p^s = \boldsymbol{Y}_p^{s,Q}$
11:         Upload $\boldsymbol{Y}_p^s$ (resp., $\nabla f_p(\boldsymbol{Y}_p^{s,t})$) to the server.
12:     end for
13:     Server side: compute $\boldsymbol{Y}^s = \frac{1}{P}\sum_{p=1}^P \boldsymbol{Y}_p^s$.
14:     $\Big($resp., $\boldsymbol{Y}^s \leftarrow \boldsymbol{Y}^{s-1} - \eta'_s \times \frac{1}{P}\sum_{p=1}^P \nabla f_p(\boldsymbol{Y}_p^s)\Big)$
15:     Broadcast $\boldsymbol{Y}^s$ to all clients.
16: end for
Ensure: $\boldsymbol{Y}$
```

3.2 Optimization

For a client p, we consider the corresponding local optimization problem

$$\underset{\boldsymbol{Y}}{\text{minimize}} \ f_p(\boldsymbol{Y}) \tag{12}$$

where $f_p(\boldsymbol{Y}) = \text{MMD}(\boldsymbol{X}_p, \boldsymbol{Y})$. Due to the presence of kernel function, we have to use some numerical methods like gradient descent to update the decision variable $\boldsymbol{Y}$. The gradient of f_p at $\boldsymbol{Y}$ is

$$\begin{aligned} \nabla f_p(\boldsymbol{Y}) =& \frac{-4\gamma}{n_p n_y} \Big[\boldsymbol{X}_p \boldsymbol{K}_{\boldsymbol{X}_p,\boldsymbol{Y}} - \boldsymbol{Y}\text{Diag}(\mathbf{1}_{n_p}^T \boldsymbol{K}_{\boldsymbol{X}_p,\boldsymbol{Y}})\Big] \\ &+ \frac{4\gamma}{n_y(n_y-1)} \Big[\boldsymbol{Y}\boldsymbol{K}_{\boldsymbol{Y},\boldsymbol{Y}} - \boldsymbol{Y}\text{Diag}(\mathbf{1}_{n_y}^T \boldsymbol{K}_{\boldsymbol{Y},\boldsymbol{Y}})\Big] \end{aligned} \tag{13}$$

To make it more explicit, we outline the key steps of FedDL to demonstrate how the central server coordinates local models for learning global distribution in a federated way.

- Step 1: The central server initializes a global $\boldsymbol{Y}_g$ before the learning cycle begins and broadcasts it to all participating local models.
- Step 2: The local clients copy the global $\boldsymbol{Y}_g$ as their uniform initial guess $\boldsymbol{Y}_p$ and compute the gradient $\nabla f_p(\boldsymbol{Y}_p)$.
- Step 3: Each client p sends its gradient $\nabla f_p(\boldsymbol{Y}_p)$ or the updated $\boldsymbol{Y}$, i.e.,
$$\boldsymbol{Y}_p \leftarrow \boldsymbol{Y}_p - \eta \nabla f_p(\boldsymbol{Y}_p) \tag{14}$$
to the central server, where η is the step size and can be set as the reverse of the Lipschitz constant of gradient if possible.
- Step 4: The central server updates the global $\boldsymbol{Y}$ by averaging all posted $\boldsymbol{Y}_p$, i.e.,
$$\boldsymbol{Y} = \frac{1}{P}\sum_{p=1}^P \boldsymbol{Y}_p, \tag{15}$$
or performing gradient descent with the average of all $\nabla f_p(\boldsymbol{Y}_p)$, i.e.,
$$\boldsymbol{Y} \leftarrow \boldsymbol{Y} - \eta' \times \frac{1}{P}\sum_{p=1}^P \nabla f_p(\boldsymbol{Y}_p), \tag{16}$$
where η' is a step size.
- Step 5: The central server broadcasts the newly aggregated communication variables so as to trigger the next local updates.

The optimization details are summarized in Algorithm 1. In the algorithm, for each client p, the time complexity per iteration is $\mathcal{O}(mn_p^2 + mn_p n_y)$ and the space complexity is $\mathcal{O}(mn_p + mn_y + n_p n_y)$. Since the optimization is similar to FedSGD or FedAVG (McMahan et al. 2017), for simplicity, we will not repeat the proof of convergence. However, Figure 2 in Section 6.1 will show the convergence of the optimization numerically.

3.3 Privacy protection

In our proposed FedDL Algorithm 1, it is necessary to share some variables like the global distribution information $\boldsymbol{Y}$ or the gradient $\nabla f_p(\boldsymbol{Y})$ for proceeding the process of training. This may result in the leakage of data privacy. Data or gradient perturbation by some special types of noise is a common way to enhance the security of federated algorithms. In Section 5, we present the theoretical guarantees of distance estimation and similarity estimation and analyze the properties of differential privacy in such two ways, respectively.

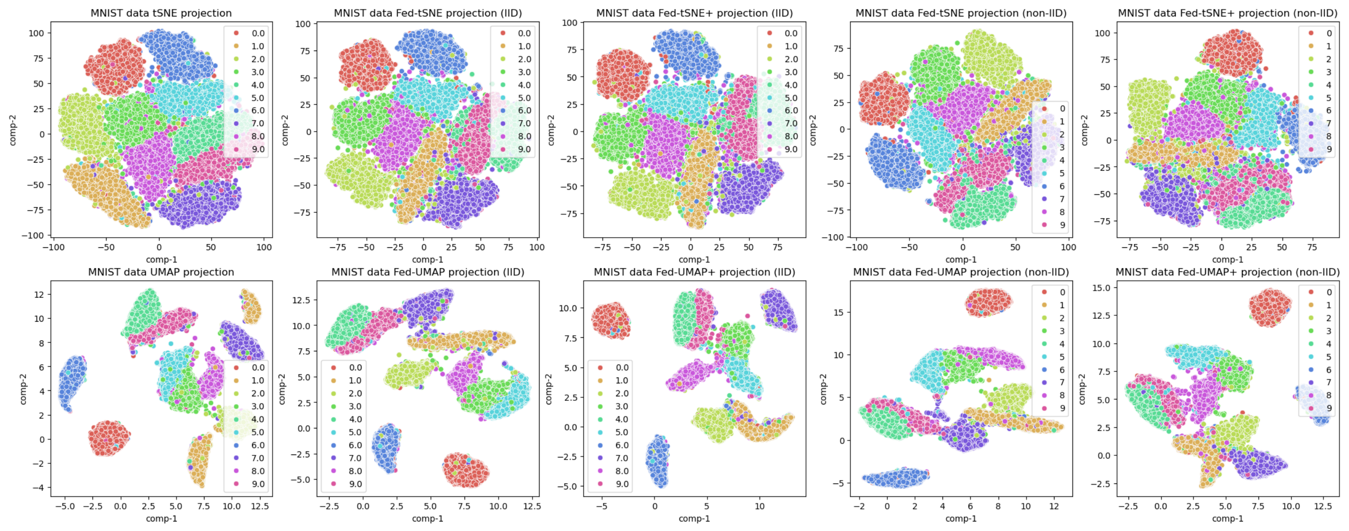

Figure 1: MNIST Data Visualization. Row 1: t-SNE, Fed-tSNE, and Fed-tSNE+. Row 2: UMAP, Fed-UMAP, and Fed-UMAP+.

4 Applications of FedDL

4.1 Federated tSNE and UMAP

Nystrom approximation is a technique that can approximate a positive semi-definite (PSD) matrix merely through a subset of its rows and columns (Williams and Seeger 2001). Consider a PSD matrix $\mathcal{S}_+^n \ni \boldsymbol{H} \succeq 0$ that has a representation of block matrix

$$\mathcal{S}_+^n \ni \boldsymbol{H} = \begin{bmatrix} \boldsymbol{W} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{Z} \end{bmatrix} \tag{17}$$

where $\boldsymbol{W} \in \mathcal{S}_+^c$, $\boldsymbol{B} \in \mathbb{R}^{(n-c)\times c}$, and $\boldsymbol{Z} \in \mathcal{S}_+^{n-c}$ for which $c \ll n$. Specifically, suppose $\boldsymbol{Z}$ is unknown, we can approximate it using $\boldsymbol{W}$, $\boldsymbol{B}$, and $\boldsymbol{B}^T$ as

$$\boldsymbol{Z} \approx \boldsymbol{B}\boldsymbol{W}_k^\dagger \boldsymbol{B}^T \triangleq \widehat{\boldsymbol{Z}} \tag{18}$$

This means we can approximate the incomplete $\boldsymbol{H}$ by $\widehat{\boldsymbol{H}} = [\boldsymbol{W}, \boldsymbol{B}^T; \boldsymbol{B}, \widehat{\boldsymbol{Z}}]$. By Nyström method, we can approximate a distance or similarity matrix on large-scale dataset in a relatively low computational complexity. Some literature gives some useful upper bounds on Nyström approximation in terms of Frobenius norm and spectral norm for different sampling techniques (Kumar, Mohri, and Talwalkar 2009b; Drineas and Mahoney 2005; Zhang, Tsang, and Kwok 2008; Kumar, Mohri, and Talwalkar 2009a; Li, Kwok, and Lu 2010). Here, we present the upper bounds of Nyström approximation in (Drineas and Mahoney 2005) for our subsequent derivation.

Theorem 1 (Error bounds of Nyström approximation)**.** *Given $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{m\times n}$, let $\widehat{\boldsymbol{H}}$ be the rank-k Nystrom approximation of $\boldsymbol{H}$ only through c columns sampled uniformly at random without replacement from $\boldsymbol{H}$, and $\boldsymbol{H}_k$ be the best rank-k approximation of $\boldsymbol{H}$. Then, the following inequalities hold for any sample of size c:*

$$\begin{aligned} &\|\boldsymbol{H} - \widehat{\boldsymbol{H}}\|_2 \le \|\boldsymbol{H} - \boldsymbol{H}_k\|_2 + \tfrac{2n\rho}{\sqrt{c}} \\ &\|\boldsymbol{H} - \widehat{\boldsymbol{H}}\|_F \le \|\boldsymbol{H} - \boldsymbol{H}_k\|_F + \rho\left(\tfrac{64k}{c}\right)^{1/4} \end{aligned} \tag{19}$$

where $\rho = \max_i \boldsymbol{H}_{ii}$.

Without the retrieval of raw data from clients, we present federated tSNE (Fed-tSNE) and federated UMAP (FedUMAP) to visualize the high-dimensional data distributed across multiple regional centers. The main idea is to perform Algorithm 1 to learn a $\boldsymbol{Y}$ and then each client p posts the distance matrix $\boldsymbol{D}_{\boldsymbol{X}_p,\boldsymbol{Y}} \in \mathbb{R}^{n_p\times n_y}$ between $\boldsymbol{X}_p$ and $\boldsymbol{Y}$ to the central server. Consequently, the central server assembles all $\boldsymbol{D}_{\boldsymbol{X}_p,\boldsymbol{Y}}$ to form

$$\boldsymbol{B} = [\boldsymbol{D}_{\boldsymbol{X}_1,\boldsymbol{Y}}^\top\ \boldsymbol{D}_{\boldsymbol{X}_2,\boldsymbol{Y}}^\top\ \cdots\ \boldsymbol{D}_{\boldsymbol{X}_P,\boldsymbol{Y}}^\top]^\top \tag{20}$$

and estimate $\boldsymbol{D}_{\boldsymbol{X},\boldsymbol{X}}$ as

$$\widehat{\boldsymbol{D}}_{\boldsymbol{X},\boldsymbol{X}} = \boldsymbol{B}\boldsymbol{W}_k^\dagger \boldsymbol{B}^\top \tag{21}$$

where $\boldsymbol{W} = \boldsymbol{D}_{\boldsymbol{Y},\boldsymbol{Y}}$, i.e., the distance matrix of $\boldsymbol{Y}$. Note that in the case that $\boldsymbol{W}$ is singular, we can add an identity matrix to it, i.e., $\boldsymbol{W} + \lambda\boldsymbol{I}$, where $\lambda > 0$ is a small constant. Finally, the central server implements either t-SNE or UMAP based on $\boldsymbol{D}_{\boldsymbol{X},\boldsymbol{X}}$. The steps are summarized into Algorithm 2.

```
Algorithm 2: Fed-tSNE and Fed-UMAP
Require: Distributed data $\{X_1, X_2, \ldots, X_P\}$ at $P$ clients.
1: Perform Algorithm 1 to compute $Y$.
2: Each client $p$ computes the distance matrix $D_{X_p,Y}$ and
   posts it to the central server.
3: The central server constructs $B$ using (20) and com-
   putes $\widehat{D}_{X,X}$ using (21).
4: The central server runs either t-SNE or UMAP on
   $\widehat{D}_{X,X}$ to obtain the low-dimensional embeddings $Z$.
Ensure: $Z$
```

Note that sampling data points from clients like in classical Nyström approximation is prohibitive in the federated settings. Thus, it motivates us to use FedDL to learn a useful set of fake points (*i.e.*, landmarks) close enough to the data across the clients in terms of MMD.

4.2 Federated Spectral Clustering

Note that after running Algorithm 1, if each client post the kernel matrix $\boldsymbol{K}_{\boldsymbol{X}_p,\boldsymbol{Y}}$ rather than the distance matrix $\boldsymbol{D}_{\boldsymbol{X}_p,\boldsymbol{Y}}$ to the central server, the central server can construct a kernel or similarity matrix $\widehat{\boldsymbol{K}}_{\boldsymbol{X},\boldsymbol{X}}$ that is useful for spectral clustering. Thus we obtain federated spectral clustering, of which the steps are summarized into Algorithm 3.

Algorithm 3: Fed-SpeClust

```
Require: Distributed data $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_P\}$ at $P$ clients.
1: Perform Algorithm 1 to compute $\boldsymbol{Y}$.
2: Each client $p$ computes the kernel matrix $\boldsymbol{K}_{\boldsymbol{X}_p,\boldsymbol{Y}}$ and posts it to the central server.
3: The central server constructs $\boldsymbol{C} = [\boldsymbol{K}_{\boldsymbol{X}_1,\boldsymbol{Y}}^\top \ \boldsymbol{K}_{\boldsymbol{X}_2,\boldsymbol{Y}}^\top \ \cdots \ \boldsymbol{K}_{\boldsymbol{X}_P,\boldsymbol{Y}}^\top]^\top$ and computes $\widehat{\boldsymbol{K}}_{\boldsymbol{X},\boldsymbol{X}} = \boldsymbol{C}\boldsymbol{W}^{-1}\boldsymbol{C}^\top$ with $\boldsymbol{W} = \boldsymbol{K}_{\boldsymbol{Y},\boldsymbol{Y}}$.
4: The central server runs spectral clustering on $\widehat{\boldsymbol{K}}_{\boldsymbol{X},\boldsymbol{X}}$ to obtain the clusters $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_c\}$.
Ensure: $\mathcal{C}$
```

5 FedDL with differential privacy

5.1 FedDL by data perturbation

We inject noise into the raw data in each client and then run FedDL to learn the global distribution information. Note that data perturbation is a one-shot operation before performing Algorithm 1. Specifically, the data $\boldsymbol{X}$ is perturbed by a noise matrix $\boldsymbol{E} \in \mathbb{R}^{m\times n_x}$ to form the noisy data matrix $\tilde{\boldsymbol{X}} = \boldsymbol{X} + \boldsymbol{E}$, where $e_{i,j} \sim \mathcal{N}(0, \sigma^2)$. Define $\tilde{\boldsymbol{X}} = \{\tilde{\boldsymbol{X}}_p\}_{p=1}^P$ and we then perform Algorithm 1 on $\tilde{\boldsymbol{X}}$ to obtain $\boldsymbol{Y}$ which gives the Nyström approximation

$$\widehat{\boldsymbol{H}}_{\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}|\boldsymbol{Y}} \simeq \boldsymbol{B}\boldsymbol{W}_k^\dagger \boldsymbol{B}^T \tag{22}$$

where $\boldsymbol{B} = \boldsymbol{K}_{\tilde{\boldsymbol{X}},\boldsymbol{Y}}$ (or $\boldsymbol{D}_{\tilde{\boldsymbol{X}},\boldsymbol{Y}}$), $\boldsymbol{W} = \boldsymbol{K}_{\boldsymbol{Y},\boldsymbol{Y}}$ (or $\boldsymbol{D}_{\boldsymbol{Y},\boldsymbol{Y}}$).

Following the logistics of existing literature, we give the upper bounds on the approximation error of Nyström approximation involved with FedDL, where we focus only on the kernel matrix because it is more complex than the distance matrix.

Theorem 2 (Error bound of Nyström approximation with FedDL having data perturbation)**.** *Given $\boldsymbol{X} = \{\boldsymbol{X}_p\}_{p=1}^P$ with $\boldsymbol{X}_p \in \mathbb{R}^{m\times n_p}$ having $\sum_{p=1}^P n_p = n_x$, $\boldsymbol{Y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n_y}] \in \mathbb{R}^{m\times n_y}$, let $\tilde{\boldsymbol{X}}_a^x = [\boldsymbol{Y}, \tilde{\boldsymbol{X}}]$ be the augmented matrix, $\boldsymbol{C} = \boldsymbol{K}_{\tilde{\boldsymbol{X}}_a^x,\boldsymbol{Y}}$, $\boldsymbol{W} = \boldsymbol{K}_{\boldsymbol{Y},\boldsymbol{Y}}$ with $\boldsymbol{W}_k^\dagger$ being the Moore-Penrose inverse of the best rank-k approximation of $\boldsymbol{W}$, and Cond$(\cdot)$ denote condition number of matrix. Denoting $\widehat{\boldsymbol{H}}_{\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}|\boldsymbol{Y}} = \boldsymbol{C}\boldsymbol{W}_k^\dagger\boldsymbol{C}^T$, it holds with probability at least $1 - n(n-1)e^{-t}$ that*

$$\begin{aligned}&\|\widehat{\boldsymbol{H}}_{\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}|\boldsymbol{Y}} - \boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X}}\|_2\\ \le&\text{Cond}(\boldsymbol{K}_{\tilde{\boldsymbol{X}}_a^x,\tilde{\boldsymbol{X}}_a^x})\left(\tfrac{|\text{MMD}(\tilde{\boldsymbol{X}},\boldsymbol{Y})|}{n_x+n_y} + 1\right) + 2n_x\\ &+ \sqrt{2}n_x\gamma\left[\sigma^2\xi_m^2 + \sqrt{2}\|\boldsymbol{D}_{\boldsymbol{X},\boldsymbol{X}}\|_\infty \sigma\xi_m\right]\end{aligned}$$

alternatively, it holds that

$$\begin{aligned}&\left\|\widehat{\boldsymbol{H}}_{\tilde{\boldsymbol{X}},\tilde{\boldsymbol{X}}|\boldsymbol{Y}} - \boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X}}\right\|_F\\ \le&\sqrt{n_x + n_y - k}\text{Cond}(\boldsymbol{K}_{\tilde{\boldsymbol{X}}_a^x,\tilde{\boldsymbol{X}}_a^x})\left(\tfrac{|\text{MMD}(\tilde{\boldsymbol{X}},\boldsymbol{Y})|}{n_x+n_y} + 1\right)\\ &+ 2k^{1/4}n_x\sqrt{\left(1 + \tfrac{n_y}{n_x}\right)} + \sqrt{2}n_x\gamma\left[\sigma^2\xi_m^2 + \sqrt{2}\|\boldsymbol{D}_{\boldsymbol{X},\boldsymbol{X}}\|_\infty\sigma\xi_m\right]\end{aligned}$$

Theorem 3 (Differential privacy of FedDL with data perturbation)**.** *Assume $\max_{p,j}\|(\boldsymbol{X}_p)_{:,j}\|_2 = \tau_X$, FedDL with perturbed data given by Section 5.1 is $(\varepsilon,\delta)-$differentially private if $\delta \ge 2c\tau_X/\varepsilon$, where $c^2 > 2\ln(1.25/\delta)$.*

5.2 FedDL by variable and gradient perturbation

We can also perturb the optimization variable $\boldsymbol{Y}$ or the gradient $\nabla f_p(\boldsymbol{Y}_p)$ by Gaussian noise in the training progression to improve the security of Algorithm 1. No matter which method we follow, the $\boldsymbol{Y}$ obtained by the central server is noisy, i.e., $\tilde{\boldsymbol{Y}} = \boldsymbol{Y} + \boldsymbol{E}$, where $\boldsymbol{E}$ is drawn elementwise from $\mathcal{N}(0,\sigma^2)$. Then, we do Nystrom approximation by

$$\widehat{\boldsymbol{H}}_{\boldsymbol{X},\boldsymbol{X}|\tilde{\boldsymbol{Y}}} \simeq \boldsymbol{B}\boldsymbol{W}_k^\dagger\boldsymbol{B}^T$$

where $\boldsymbol{B} = \boldsymbol{K}_{\boldsymbol{X},\tilde{\boldsymbol{Y}}}$ (or $\boldsymbol{D}_{\boldsymbol{X},\tilde{\boldsymbol{Y}}}$), $\boldsymbol{W} = \boldsymbol{K}_{\tilde{\boldsymbol{Y}},\tilde{\boldsymbol{Y}}}$ (or $\boldsymbol{D}_{\tilde{\boldsymbol{Y}},\tilde{\boldsymbol{Y}}}$).

Theorem 4 (Error bound of Nyström approximation with FedDL having gradient perturbation)**.** *With the same notations in Theorem 2, let $\tilde{\boldsymbol{X}}_a^y = [\tilde{\boldsymbol{Y}}, \boldsymbol{X}]$ be the augmented matrix. Then with probability at least $1 - n(n-1)e^{-t}$, it holds that*

$$\left\|\widehat{\boldsymbol{H}}_{\boldsymbol{X},\boldsymbol{X}|\tilde{\boldsymbol{Y}}} - \boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X}}\right\|_2 \le \text{Cond}(\boldsymbol{K}_{\tilde{\boldsymbol{X}}_a^y,\tilde{\boldsymbol{X}}_a^y})\left(\tfrac{|\text{MMD}(\boldsymbol{X},\tilde{\boldsymbol{Y}})|}{n_x+n_y} + 1\right) + 2n_x$$

alternatively, it holds that

$$\begin{aligned}&\left\|\widehat{\boldsymbol{H}}_{\boldsymbol{X},\boldsymbol{X}|\tilde{\boldsymbol{Y}}} - \boldsymbol{K}_{\boldsymbol{X},\boldsymbol{X}}\right\|_F\\ \le&\sqrt{n_x + n_y - k}\text{Cond}\left(\boldsymbol{K}_{\tilde{\boldsymbol{X}}_a^y,\tilde{\boldsymbol{X}}_a^y}\right)\left(\tfrac{|\text{MMD}(\boldsymbol{X},\tilde{\boldsymbol{Y}})|}{n_x+n_y} + 1\right)\\ &+ 2k^{1/4}n_x\sqrt{1 + \tfrac{n_y}{n_x}}\end{aligned}$$

Note that $\text{MMD}(\boldsymbol{X},\tilde{\boldsymbol{Y}}) \le \text{MMD}(\boldsymbol{X},\boldsymbol{Y}) + \text{MMD}(\boldsymbol{Y},\tilde{\boldsymbol{Y}})$ is related to σ. A smaller σ leads to a lower estimation error (higher estimation accuracy) but weaker privacy protection. We can obtain a precise trade-off between accuracy and privacy by combining Theorem 4 with Theorem 5.

Theorem 5 (Differential privacy of FedDL with gradient perturbation)**.** *Suppose $\max_{p,j}\|(\boldsymbol{X}_p)_{:,j}\|_2 = \tau_X$, $\max_{p,i,j}\|(\boldsymbol{Y}_p)_{:,i} - (\boldsymbol{X}_p)_{:,j}\| = \Upsilon$, $\|\boldsymbol{Y}_p^s\|_{sp} \le \tau_Y \forall s$, let $\{\nabla f_p(\boldsymbol{Y}_p^s)\}_{p=1}^P$ for $s \in [S]$ be the sequence that is perturbed by noise drawn from $\mathcal{N}(0,\sigma^2)$ with variance $8S\Delta^2\log(e + (\varepsilon/\delta))/\varepsilon^2$ where $\Delta = \frac{8\sqrt{n_y}\gamma\tau_X}{n_p n_y}\{1 + 2\gamma(\tau_X + \tau_Y)(\tau_X + \Upsilon)\}$. Then, the Gaussian Mechanism that injects noise to $\{\nabla f_p(\boldsymbol{Y}_p^s)\}_{p=1}^S$ for $s \in [S]$ is $(\varepsilon,\delta)-$differentially private.*

Note that it is intuitively appropriate to choose a decreasing sequence of noise variance $\{\sigma_s^2\}_{s=1}^S$ adapted to the gradient norm, which may make the algorithm converge well.

		IID		non-IID	
Metric	tSNE	Fed-tSNE	Fed-tSNE+	Fed-tSNE	Fed-tSNE+
CA 1-NN	0.9618±0.0015	0.9400±0.0017	0.9364±0.0020	0.9412±0.0021	0.9189±0.0030
CA 10-NN	0.9656±0.0017	0.9477±0.0017	0.9443±0.0012	0.9483±0.0019	0.9307±0.0026
CA 50-NN	0.9609±0.0015	0.9401±0.0022	0.9354±0.0022	0.9406±0.0020	0.9209±0.0035
NPA 1-NN	0.4176±0.0016	0.2728±0.0022	0.2543±0.0016	0.2729±0.0022	0.1928±0.0019
NPA 10-NN	0.3905±0.0005	0.3373±0.0007	0.3263±0.0005	0.3375±0.0013	0.2827±0.0010
NPA 50-NN	0.3441±0.0007	0.3301±0.0007	0.3258±0.0007	0.3305±0.0006	0.3030±0.0012
NMI	0.7747±0.0243	0.7534±0.0202	0.7471±0.0073	0.7399±0.0109	0.7025±0.0149
SC	0.4226±0.0082	0.4407±0.0103	0.4478±0.0066	0.4321±0.0058	0.4441±0.0045
Metric	UMAP	Fed-UMAP	Fed-UMAP+	Fed-UMAP	Fed-UMAP+
CA 1-NN	0.9322±0.0053	0.9066±0.0031	0.9007±0.0034	0.9064±0.0026	0.8730±0.0041
CA 10-NN	0.9613±0.0048	0.9445±0.0018	0.9416±0.0023	0.9449±0.0022	0.9224±0.0036
CA 50-NN	0.9602±0.0049	0.9432±0.0020	0.9400±0.0025	0.9441±0.0022	0.9219±0.0037
NPA 1-NN	0.0308±0.0009	0.0293±0.0007	0.0277±0.0008	0.0298±0.0011	0.0218±0.0009
NPA 10-NN	0.1227±0.0010	0.1133±0.0008	0.1088±0.0009	0.1131±0.0012	0.0914±0.0006
NPA 50-NN	0.2226±0.0015	0.2099±0.0011	0.2053±0.0011	0.2095±0.0013	0.1860±0.0013
NMI	0.8285±0.0150	0.7844±0.0208	0.7812±0.0153	0.7919±0.0217	0.7368±0.0194
SC	0.6118±0.0207	0.5812±0.0261	0.5746±0.0229	0.5889±0.0248	0.5422±0.0173

Table 1: Performance (mean±std) of dimensionality reduction on MNIST

		IID		non-IID	
Metric	tSNE	Fed-tSNE	Fed-tSNE+	Fed-tSNE	Fed-tSNE+
CA 1-NN	0.8112±0.0049	0.7473±0.0029	0.7198±0.0041	0.7453±0.0044	0.6669±0.0044
CA 10-NN	0.8260±0.0039	0.7892±0.0030	0.7706±0.0034	0.7898±0.0039	0.7280±0.0048
CA 50-NN	0.8064±0.0041	0.7754±0.0033	0.7631±0.0037	0.7760±0.0045	0.7280±0.0043
NPA 1-NN	0.3518±0.0018	0.1251±0.0021	0.0718±0.0013	0.1275±0.0017	0.0274±0.0006
NPA 10-NN	0.3635±0.0007	0.2551±0.0010	0.1954±0.0011	0.2571±0.0011	0.1090±0.0010
NPA 50-NN	0.3710±0.0003	0.3363±0.0006	0.3004±0.0006	0.3369±0.0008	0.2204±0.0017
NMI	0.5787±0.0212	0.5780±0.0154	0.5733±0.0149	0.5778±0.0044	0.5162±0.0129
SC	0.4049±0.0101	0.4382±0.0070	0.4638±0.0147	0.4389±0.0085	0.4564±0.0111
Metric	UMAP	Fed-UMAP	Fed-UMAP+	Fed-UMAP	Fed-UMAP+
CA 1-NN	0.7146±0.0029	0.6756±0.0036	0.6587±0.0055	0.6766±0.0043	0.6110±0.0037
CA 10-NN	0.7734±0.0039	0.7413±0.0045	0.7287±0.0041	0.7437±0.0030	0.6875±0.0041
CA 50-NN	0.7781±0.0039	0.7491±0.0052	0.7383±0.0039	0.7501±0.0040	0.7006±0.0033
NPA 1-NN	0.0356±0.0012	0.0218±0.0011	0.0156±0.0009	0.0223±0.0011	0.0071±0.0004
NPA 10-NN	0.1401±0.0013	0.1002±0.0015	0.0799±0.0012	0.1020±0.0010	0.0423±0.0007
NPA 50-NN	0.2518±0.0018	0.2152±0.0028	0.1907±0.0018	0.2167±0.0022	0.1226±0.0015
NMI	0.6187±0.0127	0.5915±0.0112	0.5755±0.0090	0.5877±0.0181	0.5191±0.0132
SC	0.5304±0.0286	0.5448±0.0264	0.5476±0.0176	0.5338±0.0252	0.5322±0.0191

Table 2: Performance (mean±std) of dimensionality reduction on Fashion-MNIST

In practice, we do not have to do this and can instead inject homoscedastic noise while incorporating a carefully chosen scaling factor into the step size of the gradient descent. By doing so, the differential privacy of our FedDL with gradient perturbation can be guaranteed by Theorem 5.

5.3 Fed-tSNE+ and Fed-UMAP+

Based on the above discussion, we propose the security-enhanced versions of Fed-tSNE and Fed-UMAP, denoted by Fed-tSNE+ and Fed-UMAP+, for which Algorithm 2 has noise injection in line 1 (Algorithm 1).

6 Experiments

6.1 Data Visualization

We applied the proposed Fed-tSNE and Fed-UMAP methods to the MNIST and Fashion-MNIST datasets. We designed the experiment with 10 clients, where IID (independent and identically distributed) refers to each client's data being randomly sampled from the MNIST dataset, thus including all classes. In contrast, non-IID means that each client's data contains only a single class. After reducing the data dimension to two, we visualized them. Figure 1 presents the results on MNIST, showing the data distribution under both IID and non-IID conditions. Additionally, we included

			IID		non-IID	
	Metric	SpeClust	Fed-SpeClust	Fed-SpeClust+	Fed-SpeClust	Fed-SpeClust+
MNIST	NMI	0.5415±0.0009	0.5240±0.0038	0.5220±0.0052	0.5235±0.0051	0.5025±0.0068
	ARI	0.3837±0.0008	0.3815±0.0076	0.3807±0.0088	0.3806±0.1123	0.3829±0.0102
COIL-20	NMI	0.8885±0.0016	0.8425±0.0218	0.8333±0.0173	0.8339±0.0216	0.8215±0.0163
	ARI	0.6066±0.0012	0.5113±0.0557	0.4793±0.0426	0.4895±0.0639	0.4551±0.0322
Mice-Protein	NMI	0.3241±0.0063	0.3233±0.0121	0.3220±0.0143	0.3222±0.0190	0.3198±0.0100
	ARI	0.1837±0.0037	0.1827±0.0033	0.1802±0.0154	0.1809±0.0024	0.1783±0.0016

Table 3: Performance (mean±std) of spectral clustering

results using Fed-tSNE+ and Fed-UMAP+, where the variance of noise is the same as that of the gradients. Due to space limitations, the results on Fashion-MNIST are provided in the Appendix (Figure 4). Based on the visualization results, our proposed methods perform very well in all settings, with only minor differences compared to the non-distributed results. They preserved nearly all the essential information and structure of the data. Tables 1 and 2 provide quantitative evaluations using the following metrics (detailed in Appendix A): **CA (Classification Accuracy)** with k-NN, **NPA (Neighbor Preservation Accuracy)** with k-NN, **NMI (Normalized Mutual Information)** of k-means, and **SC (Silhouette Coefficient)** of k-means. It can be observed that the performance of our proposed method shows a slight decline in various metrics compared to the nondistributed results, which is unavoidable. However, the overall differences remain within an acceptable range. Notably, the method performs slightly better on distributed data when the distribution is IID compared to non-IID. Moreover, the performance of Fed-tSNE+ and Fed-UMAP+ with added noise to protect privacy is somewhat inferior to the performance without noise, which is expected, as the non-IID scenario and the introduction of noise both impact the accuracy of $\boldsymbol{Y}$'s learning on whole $\boldsymbol{X}$, thereby affecting the final results.

Convergence Analysis We also conducted experiments to test the convergence of our methods. In Figure 2, the relevant metrics reached convergence after approximately 50 epochs. Figure 3 provides a more intuitive demonstration that, with the increase in epochs, the learning of $\boldsymbol{Y}$ significantly improves the final results of Fed-tSNE and Fed-UMAP, further confirming the feasibility of our method. (The full process visualization is included in Figure 5 of Appendix A.)

In addition, we also studied the impact of n_y and noise level β on NMI (Figures 6 and 7 in Appendix A). We see, regardless of the method or conditions, the larger the $\boldsymbol{Y}$ volume or the smaller the noise level β (indicating a lower privacy protection requirement), the better the NMI results.

6.2 Clustering performance

We utilized three datasets MNIST, COIL-20, and Mice-Protein (detailed in Appendix) to evaluate the effectiveness of our Fed-SpeClust, and the corresponding results are presented in Table 3. In addition to the NMI metric used previously, we also employed the ARI (Adjusted Rand Index) metric, detailed in Appendix. We see that both NMI and ARI indicate that Fed-SpeClust achieves results comparable to the original spectral clustering, despite a slight decrease in performance, demonstrating the feasibility of our method.

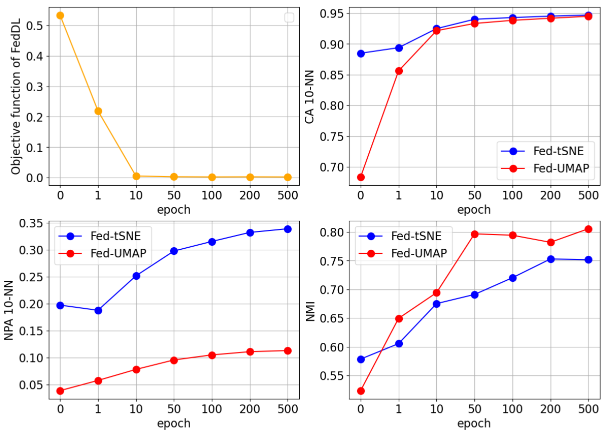

Figure 2: Convergence Performance on MNIST

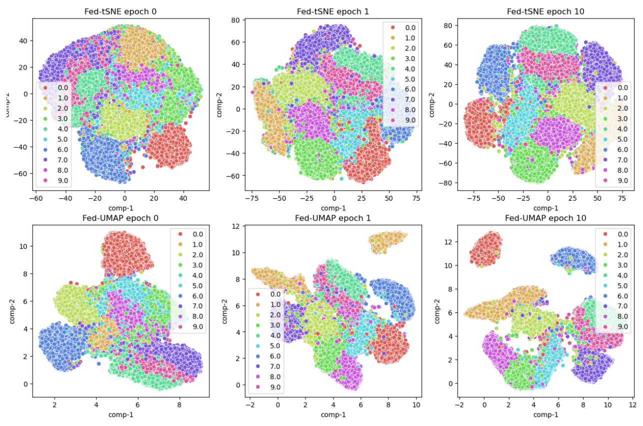

Figure 3: Visualization of Fed-tSNE and Fed-UMAP Convergence from epoch 1 to 10 (MNIST)

7 Conclusion

This work proposed FedDL and applied it to t-SNE and UMAP to visualize distributed data. The idea was also extended for spectral clustering to cluster distributed data. We provided theoretical guarantees such as differential privacy. Experimental results demonstrated that the accuracies of our federated algorithms are close to the original algorithms.

References

Baker, C. T. 1977. *The numerical treatment of integral equations*. Oxford University Press.

Becht, E.; McInnes, L.; Healy, J.; Dutertre, C.-A.; Kwok, I. W.; Ng, L. G.; Ginhoux, F.; and Newell, E. W. 2019. Dimensionality reduction for visualizing single-cell data using UMAP. *Nature biotechnology*, 37(1): 38–44.

Briguglio, W.; Yousef, W. A.; Traoré, I.; and Mamun, M. 2023. Federated Supervised Principal Component Analysis. *IEEE Transactions on Information Forensics and Security*.

Carreira-Perpinán, M. A. 2010. The Elastic Embedding Algorithm for Dimensionality Reduction. In *ICML*, volume 10, 167–174. Citeseer.

Chatzimparmpas, A.; Martins, R. M.; and Kerren, A. 2020. t-viSNE: Interactive Assessment and Interpretation of t-SNE Projections. *IEEE transactions on visualization and computer graphics*, 26(8): 2696–2714.

Drineas, P.; and Mahoney, M. W. 2005. On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research*, 6(72): 2153–2175.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4): 211–407.

Fisher, R. A. 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188.

Fu, D.; Zhang, Z.; and Fan, J. 2024. Dense Projection for Anomaly Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(8): 8398–8408.

Gisbrecht, A.; Schulz, A.; and Hammer, B. 2015. Parametric nonlinear dimensionality reduction using kernel t-SNE. *Neurocomputing*, 147(147): 71–82.

Grammenos, A.; Mendoza Smith, R.; Crowcroft, J.; and Mascolo, C. 2020. Federated principal component analysis. *Advances in neural information processing systems*, 33: 6453–6464.

Gretton, A.; Borgwardt, K. M.; Rasch, M. J.; Schölkopf, B.; and Smola, A. 2012. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1): 723–773.

Grootendorst, M. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. *arXiv preprint arXiv:2203.05794*.

Hao, Y.; Hao, S.; Andersen-Nissen, E.; Mauck, W. M.; Zheng, S.; Butler, A.; Lee, M. J.; Wilk, A. J.; Darby, C.; Zager, M.; et al. 2021. Integrated analysis of multimodal single-cell data. *Cell*, 184(13): 3573–3587.

Hinton, G. E.; and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786): 504–507.

Jolliffe, I. T.; and Cadima, J. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065): 20150202.

Kairouz, P.; McMahan, H. B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A. N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. 2021. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2): 1–210.

Kairouz, P.; Oh, S.; and Viswanath, P. 2015. The composition theorem for differential privacy. In *International conference on machine learning*, 1376–1385. PMLR.

Kohonen, T. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1): 59–69.

Kumar, S.; Mohri, M.; and Talwalkar, A. 2009a. Ensemble nystrom method. *Advances in Neural Information Processing Systems*, 22.

Kumar, S.; Mohri, M.; and Talwalkar, A. 2009b. Sampling techniques for the nystrom method. In *Artificial intelligence and statistics*, 304–311. PMLR.

Laurent, B.; and Massart, P. 2000. Adaptive estimation of a quadratic functional by model selection. *Annals of statistics*, 1302–1338.

Li, M.; Kwok, J. T.; and Lu, B.-L. 2010. Making large-scale nyström approximation possible. In *Proceedings of the 27th International Conference on Machine Learning, ICML 2010*, ICML'10, 631–638. Madison, WI, USA: Omnipress. ISBN 9781605589077.

Linderman, G. C.; Rachh, M.; Hoskins, J. G.; Steinerberger, S.; and Kluger, Y. 2019. Fast interpolation-based t-SNE for improved visualization of single-cell RNA-seq data. *Nature methods*, 16(3): 243–245.

Linderman, G. C.; and Steinerberger, S. 2019. Clustering with t-SNE, Provably. *SIAM Journal on Mathematics of Data Science*, 1(2): 313–332.

McInnes, L.; Healy, J.; Saul, N.; and Großberger, L. 2018. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*, 3(29): 861.

McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; and y Arcas, B. A. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, 1273–1282. PMLR.

Nolet, C. J.; Lafargue, V.; Raff, E.; Nanditale, T.; Oates, T.; Zedlewski, J.; and Patterson, J. 2021. Bringing UMAP closer to the speed of light with GPU acceleration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 418–426.

Novoa-Paradela, D.; Fontenla-Romero, O.; and Guijarro-Berdiñas, B. 2023. Fast deep autoencoder for federated learning. *Pattern Recognition*, 143: 109805.

Pearson, K. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11): 559–572.

Pezzotti, N.; Lelieveldt, B. P.; Van Der Maaten, L.; Höllt, T.; Eisemann, E.; and Vilanova, A. 2016. Approximated and user steerable tSNE for progressive visual analytics. *IEEE transactions on visualization and computer graphics*, 23(7): 1739–1752.

Roweis, S. T.; and Saul, L. K. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500): 2323–2326.

Saha, D. K.; Calhoun, V. D.; Du, Y.; Fu, Z.; Kwon, S. M.; Sarwate, A. D.; Panta, S. R.; and Plis, S. M. 2022. Privacy-preserving quality control of neuroimaging datasets in federated environments. *Human Brain Mapping*, 43(7): 2289–2310.

Sainburg, T.; McInnes, L.; and Gentner, T. Q. 2021. Parametric UMAP embeddings for representation and semisupervised learning. *Neural Computation*, 33(11): 2881–2907.

Sammon, J. W. 1969. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5): 401–409.

Schölkopf, B.; Smola, A.; and Müller, K.-R. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5): 1299–1319.

Sun, Y.; Han, Y.; and Fan, J. 2023. Laplacian-based Cluster-Contractive t-SNE for High-Dimensional Data Visualization. *ACM Transactions on Knowledge Discovery from Data*, 18(1): 1–22.

Tenenbaum, J. B.; De Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500): 2319–2323.

Van Der Maaten, L. 2014. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1): 3221–3245.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Van Der Maaten, L.; Postma, E. O.; Van Den Herik, H. J.; et al. 2009. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10(66-71): 13.

Williams, C.; and Seeger, M. 2001. Using the Nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13: 682–688.

Wu, Z.; Zhang, Z.; and Fan, J. 2023. Graph Convolutional Kernel Machine versus Graph Convolutional Networks. In Oh, A.; Naumann, T.; Globerson, A.; Saenko, K.; Hardt, M.; and Levine, S., eds., *Advances in Neural Information Processing Systems*, volume 36, 19650–19672. Curran Associates, Inc.

Xie, B.; Mu, Y.; Tao, D.; and Huang, K. 2011. m-SNE: Multiview stochastic neighbor embedding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(4): 1088–1096.

Yang, Z.; King, I.; Xu, Z.; and Oja, E. 2009. Heavy-tailed symmetric stochastic neighbor embedding. *Advances in neural information processing systems*, 22: 2169–2177.

Zhang, K.; Tsang, I. W.; and Kwok, J. T. 2008. Improved Nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, 1232–1239.

Reproducibility Checklist

This paper

- Includes a conceptual outline and/or pseudocode description of AI methods introduced (**YES**/partial/no/NA).
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (**YES**/no).
- Provides well marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (**YES**/no).

Does this paper make theoretical contributions? (**YES**/no)

If yes, please complete the list below:

- All assumptions and restrictions are stated clearly and formally. (**YES**/partial/no).
- All novel claims are stated formally (e.g., in theorem statements). (**YES**/partial/no).
- Proofs of all novel claims are included. (**YES**/partial/no).
- Proof sketches or intuitions are given for complex and/or novel results. (**YES**/partial/no).
- Appropriate citations to theoretical tools used are given. (**YES**/partial/no).
- All theoretical claims are demonstrated empirically to hold. (yes/**PARTIAL**/no/NA).
- All experimental code used to eliminate or disprove claims is included. (yes/no/**NA**).

Does this paper include computational experiments? (**yes**/no)

If yes, please complete the list below:

- Any code required for pre-processing data is included in the appendix. (**YES**/partial/no).
- All source code required for conducting and analyzing the experiments is included in a code appendix. (**YES**/partial/no).
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. (**YES**/partial/no).
- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (**YES**/partial/no).
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. (**YES**/partial/no/NA).
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. (**YES**/partial/no).
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. (**YES**/partial/no).
- This paper states the number of algorithm runs used to compute each reported result. (**YES**/no).

- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. (**YES**/no).
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). (**YES**/partial/no).
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. (**YES**/partial/no/NA).
- This paper states the number and range of values tried per (hyper-)parameter during development of the paper, along with the criterion used for selecting the final parameter setting. (**YES**/partial/no/NA).