Federated, Fast, and Private Visualization of Decentralized Data

Debbrata K. Saha [1 2] **Vince Calhoun** [1 2] **Soo Min Kwon** [3] **Anand Sarwate** [4] **Rekha Saha** [2] **Sergey Plis** [2]

Abstract

Data visualization is an important step in many machine learning applications, as it allows for detecting outliers and discovering latent structure within data samples. In high-dimensional settings, visualization can be performed by embedding the samples into a low-dimensional space. There are several existing methods that do this embedding efficiently, but many of them rely on the assumption that all the data are locally available. In order to use such methods in a distributed setting, one would have to pool all of the datasets into a single site. However, in many domains, communication overhead and privacy concerns often preclude aggregating data from different data sources. To overcome this issue, we previously proposed decentralized Stochastic Neighbouring Embedding (dSNE), where one can embed high-dimensional data to a low-dimensional space in a decentralized manner. Yet, the dSNE algorithm still presents a couple challenges. Since dSNE communicates in an iterative manner, communication overhead may still be high. In addition, privacy is not formally guaranteed. In this paper, we introduce Faster AdaCliP dSNE (F-dSNE) that reduces communication among sites while satisfying (ϵ, δ)-differential privacy. Our experiments on two multi-site neuroimaging datasets demonstrate that we can still obtain promising results while addressing these remaining challenges.

[1]Georgia Institute of Technology [2]Tri-institutional Center for Translational Research in Neuroimaging and Data Science (TReNDS), Georgia State University, Georgia Institute of Technology, and Emory University, Atlanta, GA 30303 [3]University of Michigan [4]Rutgers, The State University of New Jersey. Correspondence to: Debbrata Saha <dsaha34@gatech.edu>.

Workshop of Federated Learning and Analytics in Practice, colocated with 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

The rapid growth and availability of large scale datasets are making machine learning (ML) algorithms more practical and feasible(Halevy et al., 2009). Unfortunately, many of these datasets may be "noisy" in the sense that data values may be missing or of low quality. Recent advances in ML methods (e.g. deep learning (Goodfellow et al., 2016; Schmidhuber, 2015)) can effectively average out problems with individual samples, but in the medical domain, the situation is drastically different. For example, consider a magnetic resonance image (MRI) data sample that consists of the entire brain, containing on the order of 100,000 volumetric pixels (voxels) (Huettel, 2014). Due to its size and overhead, MRI data collection process is expensive, and outliers in smaller datasets hinder statistical analysis. One potential solution may be to scan each data sample for quality evaluation, but it's impractical for larger datasets. A more effective solution would be to project the high-dimensional samples into a lower-dimensional space for visualization. This visualization would allow us to actually see what data samples may be outliers, and have been proven to be an effective tool in a neuroimaging study(Panta et al., 2016).

There are many existing methods that embed (or project) high-dimensional data to a lower-dimensional space. For example, principal component analysis (PCA)(Hotelling, 1933) is a tool that can be used to extract the underlying linear structure of data. However, PCA does not work well for data samples that have inherently non-linear structures. There are other notable non-linear methods, such as Sammon mapping(Sammon Jr, 1969), curvilinear component analysis(Demartines & Hérault, 1997), stochastic neighbor embedding(Hinton & Roweis, 2002), and maximum variance unfolding(Weinberger & Saul, 2006), but many of them struggle to retain global and local structure in high-dimensional settings. To resolve this issue, there are methods such as t-SNE(van der Maaten & Hinton, 2008), hierarchical stochastic neighbor embedding (HSNE)(van Unen et al., 2017), and uniform manifold approximation and projection (UMAP)(McInnes et al., 2018) that have been proven to be effective. However, all of these methods were developed under the assumption that all data samples require locally accessible data for analysis. If data samples were distributed across multiple sites, the sites would have to pool all of the data to a single site for analysis. This is not possible for large, sensitive datasets, such as certain fMRI data.

To address these remaining problems, we previously proposed decentralized stochastic neighbor embedding (dSNE)(Saha et al., 2017). dSNE is an iterative approach that is able to embed high-dimensional distributed datasets into a low-dimensional map for visualization and inspection. At each iteration, the local sites compute a "reference" gradient that is shared between the central and local nodes. However, since these gradients are passed at every iteration, communication overhead may be high. In addition, even though only reference gradients are passed, dSNE does not provide any formal privacy guarantees. In this paper, we propose a (ϵ, δ)-differentially private faster version of dSNE, F-dSNE. F-dSNE runs more local iterations than dSNE before passing the gradients to the central node. The gradients are also perturbed using a differentially private mechanism called AdaCliP(Pichapati et al., 2019). Since the noise is added after a certain number of local iterations, this method provides better utility than methods that add noise at every iteration. We evaluate our F-dSNE algorithm using the moments accountant(Abadi et al., 2016) to keep track of the privacy loss per iteration. We compare our F-dSNE algorithm to existing dSNE and DP-dSNE(Saha et al., 2022) algorithms on three different datasets. Our results show that reducing communication and adding privacy can still obtain promising results even in high privacy regimes ($\epsilon < 1$).

2. METHODS

The objective of centralized data embedding is to produce a dataset of N samples $\mathbf{Y} = [\mathbf{y}_1 \ldots, \mathbf{y}_N]$, where $\mathbf{y}_i \in \mathbb{R}^m$, from a dataset $\mathbf{X} = [\mathbf{x}_1 \ldots, \mathbf{x}_N]$, where $\mathbf{x}_i \in \mathbb{R}^n$, such that $m \ll n$. For the task of visualization, m is usually set to $m = 2$. A very effective method that can perform this embedding is t-SNE. In t-SNE, the nearby points in $\mathbf{X}$ of high-dimensional space must be as close to the points in $\mathbf{Y}$ of the low-dimensional space (van der Maaten & Hinton, 2008). At the beginning, t-SNE computes the distance between data points in high-dimensional space into conditional probabilities, referred to as pairwise affinities. These pairwise affinities represent the similarities between the data points (see Algorithm 1). To compute similarity of a datapoint x_j to datapoint x_i, the algorithm computes the weight of x_j given by a Gaussian kernel centered at x_i with bandwidth (variance) $\sigma_i(\rho)^2$, where ρ is the perplexity parameter. These values identify σ_i separately for each data point by performing binary search across a range of values until it can match the user-specified perplexity (ρ).

$$p_{j|i} = \begin{cases} 0 & j = i \\ \dfrac{\exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i(\rho)^2)}{\sum_{k \neq i} \exp(-||\mathbf{x}_i - \mathbf{x}_k||^2/2\sigma_i(\rho)^2)} & j \neq i \end{cases} \tag{1}$$

Algorithm 1 PairwiseAffinities

```
**Input:** $\rho$ (perplexity), $\mathbf{X} \in R^{N \times n}$
**Output:** $\mathbf{P} \in \mathbb{R}^{N \times N}$
1. Eq. (1) to compute $p_{j|i}$ with perplexity $\rho$
2. Set $p_{ij} = (p_{j|i} + p_{i|j})/(2N)$ for all $i, j$.
```

For the low-dimensional representation $\mathbf{Y}$, pairwise weights are computed in a similar fashion (equation 2). The only difference is that it uses the Student-t distribution with one degree of freedom (or a Cauchy distribution) instead of a Gaussian to compute joint distribution q_{ij}. The computation of gradient is shown in Appendix A.

$$q_{ij} = \begin{cases} 0 & j = i \\ \frac{(1+||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l}(1+||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}} & j \neq i. \end{cases} \tag{2}$$

Computing distances between data points is challenging when the data are distributed. Without this distance metric (see equation (1)), it is not possible to form a common embedding. dSNE is an algorithm that can overcome these challenges, performing t-SNE in a decentralized manner. dSNE leverages a public dataset so that the local sites can communicate without having to send their private data to the other sites. Fortunately, public datasets are available in many research fields such as neuroimaging (Hall et al., 2012). We use these datasets to form a common overall embedding from all of the sites.

We now formally describe the setting in which F-dSNE is used. There are a total of L sites, where each site ℓ has (local) data $\mathbf{X}^\ell = [\mathbf{x}_1^\ell, \mathbf{x}_2^\ell, \ldots \mathbf{x}_{N_\ell}^\ell]$ consisting of N_ℓ vectors with $\mathbf{x}^\ell \in \mathbb{R}^n$. In addition, we have a shared (public) dataset $\mathbf{X}^s = [\mathbf{x}_1^s, \mathbf{x}_2^s, \ldots, \mathbf{x}_{N_s}^s]$. The goal is to produce embeddings $\{\mathbf{Y}^\ell\}$ and $\mathbf{Y}^s$, where $\mathbf{Y}^\ell = [\mathbf{y}_1^\ell, \mathbf{y}_2^\ell, \ldots \mathbf{y}_{N_\ell}^\ell]$ for each site ℓ and $\mathbf{Y}^s = [\mathbf{y}_1^s, \mathbf{y}_2^s, \ldots \mathbf{y}_{N_s}^s]$ contain vectors in $\mathbb{R}^m$, where $m \ll n$. Here, we assume that each local site has access to the shared data $\mathbf{X}^s$ and its embedding $\mathbf{Y}^s$, and can modify it during the local computation.

For F-dSNE, the messages between the local and the central site are passed after k iterations rather than at every iteration. At time point t, the central site passes the reference embedding $\mathbf{Y}^s(t-1)$ to each of the local sites. Now, each site ℓ has $\mathbf{X}^\ell$, past values of $\mathbf{Y}^\ell(t-j)$ For $j = 1, 2, \ldots, t$, the reference data set $\mathbf{X}^s$, and the updated embedding $\mathbf{Y}^s(t-1)$. Each local site computes the gradient update (Algorithm 2) until a fixed number of iterations k. After k iterations, each site adds Gaussian noise to the gradients by using a recently proposed method called AdaCliP (Algorithm 4). Then, each local site has updates $\mathbf{Y}^\ell(t)$ and $\mathbf{Y}^{s,\ell}(t)$ for the local and shared data embeddings, respectively. Each local site passes the new embeddings of their local $\mathbf{Y}^{s,\ell}(t)$ to the central site. Lastly, the central site sends back the average of $\hat{\mathbf{Y}}^s$ to all of the local sites. The local sites update their shared and

Algorithm 2 LocalGradStep

```
Input: Data embeddings: $\mathbf{Y}^\ell(t-1)$, $\mathbf{Y}^\ell(t-2)$ (local), $\mathbf{Y}^s(t-1)$, $\mathbf{Y}^s(t-2)$ (shared), $\mathbf{P} \in \mathbb{R}^{(N_\ell+N_s)\times(N_\ell+N_s)}$
Optimization parameters: $\eta$, $\alpha$
Output: $\hat{\mathbf{Y}}^\ell(t)$ (local), $\hat{\mathbf{Y}}^s(t)$ (shared)
1. Eq. (2) on $[\mathbf{Y}^\ell(t), \mathbf{Y}^s(t)]$ to compute low-dimensional affinities $q_{ij}$
2. Compute gradient: $\frac{\partial J}{\partial \mathbf{y}_i} = 4\sum_j (p_{ij}-q_{ij})(\mathbf{y}_i-\mathbf{y}_j)(1+||\mathbf{y}_i-\mathbf{y}_j||^2)^{-1}$
3. $\hat{\mathbf{y}}_i(t) = \eta(\partial J/\partial \mathbf{y}_i(t-1)) + \alpha(\mathbf{y}_i(t-1) - \mathbf{y}_i(t-2))$
4. Group $\{\hat{\mathbf{y}}_i(t)\}$ into $\hat{\mathbf{Y}}^\ell(t)$ (local) and $\hat{\mathbf{Y}}^s(t)$ (shared)
```

Algorithm 3 noiseAddition

```
Input: Gradient Matrix: $G(t) = [g_1^t, \dots, g_N^t]^\top$ (gradient at iteration $t$ for $N$ samples)
Noise Parameters: $m_i^t$, $b_i^t$ (vector parameter for each gradient), $\sigma$ (noise scale)
Output: $\tilde{G}(t)$ (privacy-preserving approximation of $G(t)$)
for $i = 1$ to $N$ do
    Transform each gradient: $w_i^t = \frac{g_i^t - m_i^t}{b_i^t}$
    Clip transformed gradient: $\hat{w}_i^t = \frac{w_i^t}{\max(1, ||w_i^t||_2)}$
    Add noise to gradient: $\tilde{w}_i^t = \hat{w}_i^t + \mathcal{N}(0, \sigma^2 I)$
    Rescale the gradient: $\tilde{g}_i^t = b_i^t \tilde{w}_i^t + m_i^t$
end for
```

Algorithm 4 AdaCliP

```
1: Input: Gradient matrix: $G(t) \in \mathbb{R}^{n\times N}$ (gradient at iteration $t$ for $N$ samples)
2: Parameters: $h_1, h_2, \beta_1, \beta_2$, $M(t) = [m_1^t, \dots, m_N^t]$, $S(t) = [s_1^t, \dots, s_N^t]$ (noise parameters for each gradient)
3: Output: $\tilde{G}(t)$ (noisy gradient matrix)
4: for $i = 1$ to $N$ do
5:     for $j = 1$ to $n$ do
6:         $b_i^t = \sqrt{s_i^t} \cdot \sqrt{\sum_{j=1}^n s_j^t}$
7:     end for
8:     Add noise to each gradient in matrix: $\tilde{g}_i^t$ = noiseAddition$(g_i^t, m_i^t, b_i^t, \sigma)$
9:     Update $m_i^t$: $m_i^{t+1} = \beta_1 m_i^t + (1-\beta_1)\tilde{g}_i^t$
10:    Compute variance $v_i^t$: $v_i^t$ = min(max$((\tilde{g}_i^t - m_i^t)^2 - (b_i^t)^2\sigma^2, h_1), h_2)$
11:    Update $s_i^t$: $s_i^{t+1} = \beta_2(s_i^t)^2 + (1-\beta_2)v_i^t$
12: end for
```

local embeddings and averages them for recentering.

They send this average to the coordinator (or central node), who averages across the sites and sends back a global mean. Each site uses the mean to center their local and shared embeddings to get $\mathbf{Y}^{(\ell)}(t)$ and $\mathbf{Y}^{(s)}(t)$ for the next iteration. Each local site keeps track of the parameters from AdaCliP to report the final (ϵ, δ) parameters. This process then repeats for a total of T iterations. The pseudocode and overall procedure for F-dSNE are shown in Appendix C(Algorithm 5). Note that at each iteration, the embedding vector for the shared dataset Y^s should be same at the each local site to ensure that the values at each local site will be updated using the same reference data at each iteration.

3. DATA

For our experiments, we use three different datasets: (1) MNIST[1], (2) Pediatric Imaging, Neurocognition, and Genetics (PING)[2] and (3) A local structural magnetic resonance imaging (sMRI) dataset. The data acquisition and preprocessing information are described in Appendix B.

4. EXPERIMENTS

For each of the experiments listed below, we compare three different algorithms: (1) dSNE, (2) DP-dSNE, and (3) F-dSNE. For dSNE, we kept the same setup as described in (Saha et al., 2017). For DP-dSNE, we added noise to the reference gradients at every iteration, where for F-dSNE, we added noise every k iterations. In addition, F-dSNE runs k local iterations before passing the gradients to the central node, whereas DP-dSNE sends them after every local iteration. The objective of these experiments are to highlight that F-dSNE can provide stronger privacy guarantees while increasing utility compared to DP-dSNE.

For the MNIST dataset, We have designed two different experiments: (1) MNIST 4 digits and (2) MNIST all digits. For MNIST 4 digit experiment, we have created 3 local sites and 1 remote site. The remote site consists of 800 reference samples, where each digit $(0, 1, 8,$ and $9)$ contains 200 samples. Each local site contains 40 samples, where site 1 contains 8 and 9, site 2 carries 0 and 9, and site 3 contains 0 and 1 digits, respectively. For the MNIST all digits experiment, there are 3 local sites and 1 remote site. The remote site contains all of the digits (0 to 9), where each digit has 200 samples. The local site also contains all of the digits but in a smaller amount. Here, all local sites consist of 20 samples of each digit. We run dSNE, DP-dSNE, and F-dSNE experiments on the same datasets and validate our results.

We collected the PING dataset from five different data sites. For the PING experiment, we consider each data source as an individual local site (total of five local sites) and prepared our reference samples by taking some small samples from each local site.

[1]https://www.kaggle.com/c/digit-recognizer
[2]http://pingstudy.ucsd.edu/Data.php

The sMRI dataset consists of subjects with four different age groups: below 11, 11 to 17, 30 to 34, and above 64. We used each age group as an individual site (total of four local sites) and formed the reference samples by taking 100 samples from each age group.

5. RESULTS

The experimental results of the MNIST 4 digits experiment are shown in Figure 1. In all of the experiments, we observe four distinct clusters, showing that all of the methods work efficiently. Figure 2 presents the results of MNIST for the all digits experiment. In Figure 2, we get ten distinct clusters for all three experiments. We can see that all of the decentralized methods obtain similar utility compared to the centralized case, reported in (van der Maaten & Hinton, 2008). In this experiment, DP-dSNE satisfies $(1.35, 10^{-5})$-DP, whereas F-dSNE satisfies $(0.13, 10^{-5})$-DP. Even with the stronger privacy guarantee (hence more noise) and less overhead, F-dSNE obtains good results.

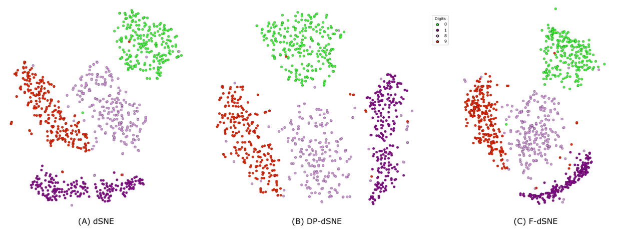

Figure 1. Experiment for the quality control (QC) metrics of the MNIST 4 digits dataset. (A), (B) and (C) are the layouts of dSNE, DP-dSNE and F-dSNE. In the layout, the samples are colored by the digits, and we can see four distinct clusters.

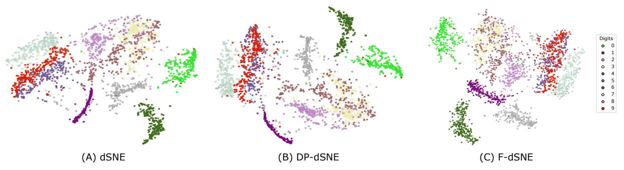

Figure 2. Experiment for the QC metrics of the MNIST all digits dataset. (A), (B) and (C) presents the output of dSNE, DP-dSNE and F-dSNE. In the layout, we can see 10 clusters, where each digit is represented by a different color.

Figure 3 presents the results of the PING experiment. For all of the algorithms, we get four distinct clusters. The privacy guarantee resulting from the DP-dSNE algorithm was $(1.39, 10^{-5})$-DP, whereas F-dSNE satisfies $(0.14, 10^{-5})$-DP. Similar to the MNIST experiment, the clusters from F-dSNE are as close to the clusters formed in the centralized setting (Saha et al., 2022), even with the increase in privacy.

Figure 4 depicts the experimental results from the sMRI dataset. For all three algorithms, we identify four clusters,

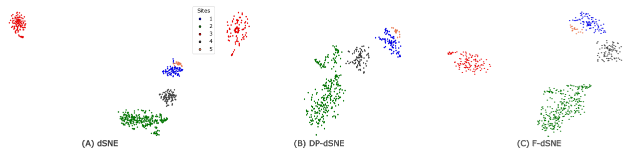

Figure 3. Experiment for QC metrics of the PING dataset. (A), (B) and (C) represent the layouts of dSNE, DP-dSNE and F-dSNE. In all of the experiments, we get four distinct clusters, similar to the centralized case. Each point in the layout represents a single individual from the dataset.

one for each age group. In the figure, we notice that children less than 11 always form a separate distinct cluster. Meanwhile, the other age groups, although connected together in a single contiguous cluster, are ordered according to age. Again, the same type of behavior is reported when the data is centralized (Saha et al., 2022).

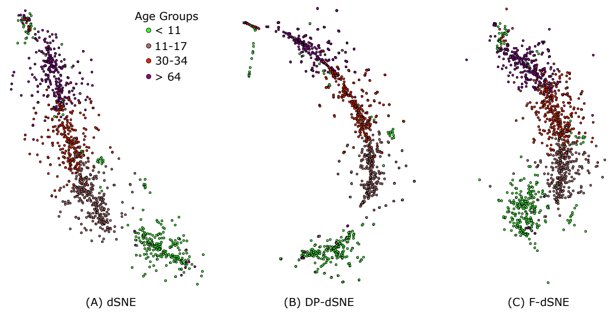

Figure 4. Experiment for the QC metrics of the sMRI dataset. (A), (B) and (C) presents the dSNE, DP-dSNE, and F-dSNE layouts, respectively. In the layout, we can see a continuous cluster of each age group. Each age group is colored differently, and each point represents a data sample.

6. DISCUSSION & CONCLUSION

In this paper, we proposed a faster, more private version of dSNE, called F-dSNE. F-dSNE can effectively embed high-dimensional distributed data to a low-dimensional space for manual visualization. By performing this visualization, we can measure the quality of the data across multiple sites. Throughout the whole computation, the private local data never leaves their respective sites, and minimal information is exchanged between the central and local sites. As a formal guarantee of privacy, we incorporated differential privacy through an algorithm called AdaCliP. F-dSNE reduced the communication overhead by 10% and increased privacy (in terms of ϵ). Our experiments showed that even though we increased privacy and reduced overhead, we can achieve the same results with F-dSNE compared to dSNE. We hypothesize that this communication overhead can be reduced more, but we leave this problem for the future work. In conclusion, F-dSNE is a valuable quality control tool for decentralized virtual consortia working with private data.

7. Acknowledgements

This work was supported by NIH 1R01DA040487, R01DA049238, R01MH121246, 2R01EB006841, 2RF1MH121885, and NSF 2112455.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

Ashburner, J. and Friston, K. J. Unified segmentation. *NeuroImage*, 26(3):839 – 851, 2005. ISSN 1053-8119. doi: 10.1016/j.neuroimage.2005.02.018.

Demartines, P. and Hérault, J. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on neural networks*, 8(1):148–154, 1997.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT Press, 2016.

Halevy, A., Norvig, P., and Pereira, F. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2): 8–12, 2009.

Hall, D., Huerta, M. F., McAuliffe, M. J., and Farber, G. K. Sharing heterogeneous data: the national database for autism research. *Neuroinformatics*, 10(4):331–339, 2012.

Hinton, G. and Roweis, S. Stochastic neighbor embedding. In *NIPS*, volume 15, pp. 833–840, 2002.

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:498–520, 1933.

Huettel, S. *Functional magnetic resonance imaging*. Sinauer Associates, Inc., Publishers, Sunderland, Massachusetts, U.S.A, 2014. ISBN 9780878936274.

McInnes, L., Healy, J., Saul, N., and Grossberger, L. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3:861, 09 2018. doi: 10.21105/joss.00861.

Panta, S. R., Wang, R., Fries, J., Kalyanam, R., Speer, N., Banich, M., Kiehl, K., King, M., Milham, M., Wager, T. D., et al. A tool for interactive data visualization: Application to over 10,000 brain imaging and phantom mri data sets. *Frontiers in neuroinformatics*, 10, 2016.

Pichapati, V., Suresh, A., Yu, F. X., Reddi, S., and Kumar, S. Adaclip: Adaptive clipping for private SGD. *CoRR*, abs/1908.07643, 2019. URL http://arxiv.org/abs/1908.07643.

Saha, D. K., Calhoun, V. D., Panta, S. R., and Plis, S. M. See without looking: joint visualization of sensitive multi-site datasets. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pp. 2672–2678, 2017.

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

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

Schmidhuber, J. Deep learning. *Scholarpedia*, 10(11): 32832, 2015. doi: 10.4249/scholarpedia.32832. URL https://doi.org/10.4249/scholarpedia.32832.

van der Maaten, L. and Hinton, G. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

van Unen, V., Höllt, T., Pezzotti, N., Li, N., Reinders, M., Eisemann, E., Koning, F., Vilanova, A., and Lelieveldt, B. Visual analysis of mass cytometry data by hierarchical stochastic neighbour embedding reveals rare cell types. *Nature Communications*, 8, 12 2017. doi: 10.1038/s41467-017-01689-9.

Weinberger, K. Q. and Saul, L. K. An introduction to nonlinear dimensionality reduction by maximum variance unfolding. In *AAAI*, volume 6, pp. 1683–1686, 2006.

A. Gradient Computation

We perform gradient descent on the Kullback-Leibler (KL) divergence (or relative entropy) between the joint distribution P and the joint distribution Q:

$$J(Y) = \sum_i \sum_{j \neq i} p_{ij} \ln \frac{p_{ij}}{q_{ij}} \tag{3}$$

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q is expressed in equation (4):

$$\frac{\partial J}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \tag{4}$$

B. Data Acquisition and Preprocessing

MNIST was collected from the Kaggle competition and consists of 60000, 28×28 grayscale images. This dataset consists of 10 unique digits from 0 to 9. From the dataset, we randomly picked 2600 samples while maintaining class balance.

PING is a multi-site study consisting neural development histories, information about developing mental and emotional functions, multimodal brain imaging data, and genotypes for well over 1000 adolescents between the ages of 3 to 20. We acquired a total of 632 subjects and their fMRI data for our experiment. The software SPM (Ashburner & Friston, 2005) was used to preprocess the data. Steps included slice time and motion correction and warping to the standard MNI brain template. The obtained image is used for extracting the data for the experiment. In our experiment, a total of five different data sites participated in the computations. Sites 1, 2, 3, 4, and 5 contained 59, 215, 79, 59, and 10 different subjects, respectively.

sMRI scans (3D T1-weighted pulse sequences) are pre-processed through the voxel based morphometry (VBM) pipeline using SPM. For each scan, the voxel values at each location from all the brain slices are first added across slices, resulting in a matrix size of 91×109. All of the voxel values from each image scan are converted into a single row vector of size 9919 for each data point and passed as inputs to the dSNE, DP-dSNE and F-dSNE algorithms. This dataset consists of people from four different age groups.

C. Pseudocode

Algorithm 5 Faster-dSNE ($\mathsf{F-dSNE}$)

```
 1: Input:
 2: Objective Parameters: $\rho$ (perplexity)
 3: Optimization Parameters: $T$, $\eta$, $\alpha$
 4: Shared Data: $\mathbf{X}^s = [\mathbf{x}_1^s, \mathbf{x}_2^s \dots \mathbf{x}_{N_s}^s], \mathbf{x}_i^s \in \mathbb{R}^n$
 5: Local Data: $\mathbf{X}^\ell = [\mathbf{x}_1^\ell, \mathbf{x}_2^\ell \dots \mathbf{x}_{N_\ell}^\ell], \mathbf{x}_i^\ell \in \mathbb{R}^n$, $\ell = 1, 2, \dots, L\}$
 6: Output: $\{\mathbf{Y}^\ell \colon \ell = 1, 2, \dots, L\}, \mathbf{Y}^s$
 7: Sample $\mathbf{Y}^s(0)$ i.i.d. from $\mathcal{N}(0, 10^{-4}\mathbf{I}_m)$ ▷ Initialize from Gaussian
 8: Coordinator sends $\mathbf{X}^s$, $\mathbf{Y}^s(0)$ to all sites
 9: for $\ell = 0$ to $L$ do
10:     $\mathbf{P}^\ell = \mathsf{PairwiseAffinities}(\rho, [\mathbf{X}_p, \mathbf{X}_s])$
11:     Sample $\mathbf{Y}^\ell(0)$, $\mathbf{Y}^\ell(-1)$ from $\mathcal{N}(0, 10^{-4}\mathbf{I}), \mathbf{I} \in R^{m \times m}$
12: end for
13: for $t = 1$ to $T$ do
14:     Coordinator sends $\mathbf{Y}^\ell(t-1)$ to all sites
15:     for $\ell = 1$ to $L$ do
16:         for $i = 1$ to $k$ do
17:             $\hat{\mathbf{Y}}^\ell(t), \hat{\mathbf{Y}}^{s,\ell}(t) = \mathsf{LocalGradStep}(\mathbf{Y}^\ell(t-1), \mathbf{Y}^\ell(t-2), \mathbf{Y}^s(t-1), \mathbf{Y}^\ell(t-2), \eta, \alpha)$
18:         end for
19:         $\breve{\mathbf{Y}}^\ell(t)$ = $\mathsf{AdaCliP}(\hat{\mathbf{Y}}^\ell(t), M(t), S(t), h_1, h_2, \beta_1, \beta_2)$
20:         $\breve{\mathbf{Y}}^{s,\ell}(t)$ = $\mathsf{AdaCliP}(\hat{\mathbf{Y}}^{s,\ell}(t), M(t), S(t), h_1, h_2, \beta_1, \beta_2)$
21:         Site $\ell$ sends $\breve{\mathbf{Y}}^{s,\ell}$ to Cooordinator and updates the local embeddings
22:     end for
23:     $\breve{\mathbf{Y}}^s = \frac{1}{L}\breve{\mathbf{Y}}^{s,\ell}$ ▷ Average local shared embeddings
24:     Coordinator sends $\breve{\mathbf{Y}}^s$ to all sites
25:     for $l = 0$ to $L$ do
26:         $\tilde{\mathbf{Y}}^\ell = \mathbf{Y}^\ell(t-1) + \breve{\mathbf{Y}}^\ell$
27:         $\tilde{\mathbf{Y}}^s = \mathbf{Y}^s(t-1) + \breve{\mathbf{Y}}^s$
28:         $\bar{\mathbf{y}}^\ell = \frac{1}{N_\ell + N_s}\left(\sum_{i=1}^{N_\ell} \tilde{\mathbf{y}}_i^\ell + \sum_{i=1}^{N_s} \tilde{\mathbf{y}}_i^s\right)$ ▷ Mean embedding
29:         Send $\bar{\mathbf{y}}^\ell$ to Coordinator
30:     end for
31:     $\bar{y} = \frac{1}{L}\sum_{\ell=1}^{L} \bar{\mathbf{y}}^\ell$
32:     Coordinator sends $\bar{y}$ to all sites
33:     for $\ell = 1$ to $L$ do
34:         Set $\mathbf{y}_i^\ell(t) = \tilde{y}_i^\ell - \bar{y}$ for all $i$
35:         Set $\mathbf{y}_i^s(t) = \tilde{y}_i^s - \bar{y}$ for all $i$
36:     end for
37: end for
```