frontiers in Genetics

METHODS
published: 06 October 2020
doi: 10.3389/fgene.2020.00786

NeuralEE: A GPU-Accelerated Elastic Embedding Dimensionality Reduction Method for Visualizing Large-Scale scRNA-Seq Data

Jiankang Xiong[1,2], *Fuzhou Gong*[1,2], *Lin Wan*[1,2]* *and Liang Ma*[3]*

[1] National Center for Mathematics and Interdisciplinary Sciences, Academy of Mathematics and Systems Science, Chinese Academy of Sciences (CAS), Beijing, China, [2] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China, [3] Key Laboratory of Zoological Systematics and Evolution, Institute of Zoology, Chinese Academy of Sciences (CAS), Beijing, China

The dramatic increase in amount and size of single-cell RNA sequencing data calls for more efficient and scalable dimensional reduction and visualization tools. Here, we design a GPU-accelerated method, NeuralEE, which aggregates the advantages of elastic embedding and neural network. We show that NeuralEE is both scalable and generalizable in dimensional reduction and visualization of large-scale scRNA-seq data. In addition, the GPU-based implementation of NeuralEE makes it applicable to limited computational resources while maintains high performance, as it takes only half an hour to visualize 1.3 million mice brain cells, and NeuralEE has generalizability for integrating newly generated data.

Keywords: single-cell RNA sequencing, elastic embedding, neural networks, large-scale, stochastic optimization, parametric models, generalizable models

OPEN ACCESS

Edited by:
Xianwen Ren,
Peking University, China

Reviewed by:
Jin Gu,
Tsinghua University, China
Yongcui Wang,
Northwest Institute of Plateau Biology (CAS), China

***Correspondence:**
Liang Ma
maliang@ioz.ac.cn
Lin Wan
lwan@amss.ac.cn

Specialty section:
This article was submitted to Genomic Assay Technology, a section of the journal Frontiers in Genetics

Received: *30 April 2020*
Accepted: *01 July 2020*
Published: *06 October 2020*

Citation:
Xiong J, Gong F, Wan L and Ma L (2020) NeuralEE: A GPU-Accelerated Elastic Embedding Dimensionality Reduction Method for Visualizing Large-Scale scRNA-Seq Data. Front. Genet. 11:786. doi: 10.3389/fgene.2020.00786

1. INTRODUCTION

Dimensionality reduction is one of the basic steps in machine learning algorithms and big-data analyses, especially in the analysis of high-throughput single cell RNA sequencing data (scRNA-seq data). scRNA-seq enables us to simultaneously profile thousands of genetic markers at single-cell resolution, which makes it an ideal tool to study the cell-cell heterogeneity in developmental biology, oncology, and immunology. Visualization of scRNA-seq data in a manageable dimension often plays as a pivotal first step prior to other downstream analyses such as cell type identification or cell developmental trajectory reconstruction.

Among the numerous dimensionality reduction and visualization methods, t-distributed stochastic neighbor embedding (t-SNE) (van der Maaten and Hinton, 2008) is most widely used in the single-cell community to visualize data structures. While t-SNE emphasizes the neighborhood information, which keeps the local affinity of the data, it tends to shatter the global structures (Becht et al., 2018). As an extension of stochastic neighbor embedding (SNE), elastic embedding (EE) algorithm penalizes, placing far apart latent points from similar data points and placing close together latent points from dissimilar data points (Carreira-Perpinán, 2010), thereby preserving the intrinsic data structure both locally and globally (Hie et al., 2020). EE has been recently proved to be well-performed in visualization and in reconstruction of the embedded structure of the cell developmental process (An et al., 2019; Chen et al., 2019).

Recent advances in automatic cell isolation and multiplex sequencing have led to an exponential growth in the number of cells (may reach the order of millions) sequenced for individual studies

(Svensson et al., 2018). Many modified version of dimensionality reduction and visualization methods, such as net-SNE (Cho et al., 2018), FIt-SNE (Linderman et al., 2019), and UMAP (Becht et al., 2018), have been introduced to deal with data in large scale. Net-SNE in particular optimizes the original t-SNE under a neural network (NN) framework. In addition, methods based on autoencode, which is in form of a specialized NN framework, have been proposed to deal with scRNA-seq data for dimensionality reduction. To list a few, scScope (Deng et al., 2019) reconstructs scRNA-seq data by a deep recurrent autoencoder. DCA (Eraslan et al., 2019) models scRNA-seq for count data by a deep count autoencoder. scVI (Lopez et al., 2018), based on a variational autoencoder (Kingma and Welling, 2014), also models count data and incorporates batch correction in addition to dimensionality reduction. Many other methods are also constructed under the deep learning framework (Ding et al., 2018; Wang and Gu, 2018) that take advantage of parallel and scalable features in deep neural networks (Lin et al., 2020). EE optimization procedures still lack sufficient scalability to mega-scale datasets.

Here, we develop neural elastic embedding termed NeuralEE, a scalable and generalizable method that trains a NN with a mini-batch trick, mapping from high-dimensional single-cell gene-expression profiles to a low-dimensional visualization. NeuralEE visualizes large-scale scRNA-seq data very efficiently and accurately on a conventional workstation with GPU-installed, making it applicable to biological labs with limited computational resources. We also validate the accuracies of visualization by NeuralEE on four benchmark datasets and a simulated dataset. Furthermore, NeuralEE has no computational restrictions on embedding dimensions, making it viable as a general purpose dimension reduction technique.

2. MATERIALS AND METHODS

2.1. EE

Given $Y_{D\times N} = (y_1, \ldots, y_N)$ the $D \times N$ matrix of the scRNA-seq dataset with N cells in $\mathbb{R}^D$, where D is typically on the order of tens of thousands (number of genes), EE seeks to find the low-dimensional embedding on $\mathbb{R}^d$, $X_{d\times N} = (x_1, \ldots, x_N)$, with $d \ll D$. Formally, EE solves the following optimization problem:

$$\min_X E(X) = \min_X \sum_{n\neq m} w^+_{nm}||x_n - x_m||^2 + \lambda \sum_{n\neq m} w^-_{nm} \exp(-||x_n - x_m||^2), \quad (1)$$

where the attractive weights $W^+_{N\times N} = (w^+_{nm})$ and the repulsive weights $W^-_{N\times N} = (w^-_{nm})$ are both $N \times N$ symmetric and non-negative matrices, which are derived from Y. The parameter λ trades off between the attractive and repulsive terms. The W^+ can be defined as Gaussian affinities (Hinton and Roweis, 2003) or entropic affinities (Vladymyrov and Carreira-Perpinan, 2013), and the W^- can be simply defined as Euclidean distance. The objective function $E(X)$ is normally solved by fixed-point iteration (Carreira-Perpinán, 2010) or partial-Hessian strategies (Vladymyrov and Carreira-Perpinan, 2012). However, when N is large, the optimization can be computationally expensive.

2.2. NeuralEE

The original EE is not generalizable: it is unable to project new samples to existing embedding. A basic approach is to use a mapping **F** belonging to a parametric family $\mathcal{F}$ of mappings, then involve **F** in the learning from the beginning, by replacing x_n with $\mathbf{F}(y_n)$ in the embedding objective function and optimizing it over the parameters of **F**.

An NN with sufficient hidden layers (with non-linear activation functions) is capable of approximate functions with arbitrary complexity (Leshno et al., 1991). We can thus choose to use a standard feedforward NN as the parametric family $\mathcal{F}$ of mappings for EE. Standard feedforward NN architecture is a multilayer stack of simple modules that maps a fixed-size input (for example, a cell represented with D genes expression) to a fixed-size output (for example, coordinate in embedded space). To go from one layer to the next, a set of units (neurons) compute weighted sums of inputs from their previous layer and pass the results to the next layer through a non-linear function. Units that are not in the input (the first) or output (the last) layers are conventionally called hidden units. The hidden layers can be seen as distorting the input in a non-linear way so that coordinates in embedded space become linearly separable by the last layer. Multilayer architectures can also be trained by simple stochastic gradient descent. As long as the modules are relatively smooth functions of their inputs and of their internal weights, one can compute gradients using the backpropagation procedure (Lecun et al., 2015).

We propose NeuralEE, which applies the NN framework with mini-batch trick to mitigate the high computational intensity when dealing with large-scale datasets. NeuralEE defines a NN that maps data from the original space to the embedded space. The embedding represents as the function of the original data is fed into Equation (1) with attractive weight and the repulsive weight matrices which are calculated offline. By use of the backpropagation algorithm (Lecun et al., 1990), the parameters in the NN are optimized, and the mapping of the embedding is thus learned.

In detail, with arbitrary parametric mapping $Net_\theta : R^D \to R^d$ defined by NN, the objective $E(X)$ in Equation (1) can be written as $E(Net_\theta(Y))$, where $Net_\theta(Y) = (Net_\theta(y_1), \ldots, Net_\theta(y_N))$ (the specific structure of NN is detailed at **Supplementary Materials**).

The parameters θ can thus be optimized via gradient descent algorithm by applying the chain rule as follows:

$$\frac{\partial E(Net_\theta)}{\partial \theta} = \sum_{n=1}^{N} \left(\frac{\partial E(X)}{\partial x_n}\right)^T \frac{\partial Net_\theta(y_n)}{\partial \theta}, \quad (2)$$

where $\frac{\partial E(X)}{\partial x_n}$ has close-form expression.

$$\frac{\partial E(X)}{\partial x_n} = 4 \sum_{m\neq n} (w^+_{nm} - \lambda w^-_{nm} \exp(-||x_n - x_m||^2))(x_n - x_m). \quad (3)$$

And $\frac{\partial Net_\theta(y_n)}{\partial \theta}$ can be acquired by backpropagation algorithm.

2.3. Stochastic Optimization

In cases where N is large, the calculation of $\frac{\partial E(X)}{\partial x_n}$ can be time-consuming. Additionally, it will be memory-costly to store the attractive and the repulsive weight matrices. We therefore further propose a stochastic optimization version of NeuralEE, termed NeuralEE-SO, by applying the mini-batch trick. It first randomly partitions the full matrix into several batches $B = \{b_i \subset \{1, \ldots, N\}\}$ before it then calculates the attractive weight and the repulsive weight matrices for each batch. At each iteration of backpropagation, the gradients of NN parameters are calculated on each batch and averaged over all batches:

$$\frac{\partial E(Net_\theta)}{\partial \theta} \approx \gamma \frac{1}{|B|} \sum_{i=1}^{|B|} \sum_{n \in b_i} \left(\frac{\partial E(X_i)}{\partial x_n}\right)^T \frac{\partial Net_\theta(y_n)}{\partial \theta}, \quad (4)$$

where γ is an offset constant, which is finally integrated into the learning rate of the gradient descent algorithm. The flowchart of NeuralEE is shown at **Figure 1A**, and we give the pseudocode of NeuralEE at **Supplementary Algorithm 1**.

2.4. Data

We test NeuralEE on five scRNA-seq datasets, including 3,005 mouse cortex cells (Zeisel et al., 2015) (hereinafter denoted as **CORTEX**), 4,016 hematopoietic stem and progenitor cells. Tusi et al. (2018) (hereinafter denoted as **HEMATO**), 12,039 human peripheral blood mononuclear cells (Zheng et al., 2017) (hereinafter denoted as **PBMC**), 27,499 mouse retinal bipolar neuron cells (Shekhar et al., 2016) (hereinafter denoted as **RETINA**), 1.3 million mouse brain cells (10x Genomics, 2017) (hereinafter denoted as **BRAIN-LARGE**), and also a simulated complex trajectory of embedded data from Moon et al. (2019) (hereinafter denoted as **ArtificialTree**) (data information is detailed at **Supplementary Materials**). The pre-filtering of cells and the labeling of cells follow the procedures in Lopez et al. (2018) for the first four data. We further filter and normalize the genes in following steps. First we apply $\log(1+x)$ transformation to each element of the cell-gene expression matrix. Then we exclude genes with low expression variance, and in most datasets retain the top 500 genes are ordered by variance (558 genes for **CORTEX**). Finally, we normalize the expression of each gene by subtracting its mean and dividing its standard deviation. We preprocess **BRAIN-LARGE** data following the procedures in Zheng et al. (2017) and, as Cho et al. (2018) does, retain the top 50 principal components (PCs) as features. For **ArtificialTree** data, we take the raw data as input.

2.5. Quantitative Evaluation

We quantitatively evaluate NeuralEE with other methods on simulation data by comparing their generalization error measured on K-nearest neighbor classifier. First, we apply different dimensionality reduction methods on the entire **ArtificialTree** dataset, where the ground-truth labels of the cells are known. We then conduct a 10-fold cross-validation procedure on the full dataset, and the cross-validation errors for different settings of K ($1 \leq K \leq 40$) are calculated for each dimensional reduction methods. The minimum cross validation errors of each methods are chosen as the corresponding generalization errors. Selected hyperparameters of NeuralEE and other methods are listed at **Supplementary Table 2**.

2.6. Implementation

NeuralEE is implemented in Python. It integrates the EE optimization protocol of Vladymyrov and Carreira-Perpinan (2012) and the NN framework based on the PyTorch (Paszke et al., 2019) module, which exploits parallel computation of GPU to accelerate the optimization. We provide freely available codes and detailed guidance on our Github site https://github.com/HiBearME/NeuralEE/tree/v0.1.6. In this study, NeuralEE is performed on a 64G computer memory workstation, with a NVIDIA GPU (GCeForce GTX 1080 Ti, 11G video RAM).

3. RESULTS

3.1. NeuralEE Preserves the Properties of EE

We first compare NeuralEE to EE and other common dimension reduction methods on **ArtificialTree** data, which contains 1,440 single cells and 60 genes. The **ArtificialTree** data have an embedded continuous tree structure with 10 branches. Each branch constitute of about 140 cells that have exclusive expression on a subset of genes, where the progression along the branch is modeled by gradually increase the expression of these genes (**Figure 2A**). The two versions of NeuralEEs together with EE resemble each other in results with EE results being more neatly presented (**Figures 2B–D**). NeuralEE inherits the merits of EE, which can preserve both global and local structure of data. The state-of-the-art methods, t-SNE and UMAP, are also good at keeping the local affinity; however, compared to other methods, they both tend to shatter the global structures (**Figures 2E,F**). PHATE (Moon et al., 2019) tends to stretch the branches, while PCA fails to resolve all the branches (**Figures 2G,H**). We further quantitatively compare the performance of NeuralEE to other methods on these simulation data. By performing the K-nearest neighbors classifiers on each embedded space, and with K varying from 1 to 40, we calculate their minimum generalization error (**Supplementary Table 1**). Except for PCA, the minimum generalization errors for other methods are all less than 0.1. NeuralEE, NeuralEE-SO, and EE exhibited the best performance with NeuralEE and EE ties at the top.

We also apply NeuralEE together with other dimension reduction methods on real biological data (**Supplementary Figure 1**). As expected, both versions of NeuralEE give very similar visualization structures on these datasets as EE. However, when the size of dataset is large, EE suffers from computation problems. For instance, it fails to run the **RETINA** dataset. NeuralEE, and especially NeuralEE-SO, on the other hand, overcome this issue and are able to carry forward the characteristics of EE on larger datasets. The parameter λ in EE (and thus in NeuralEE) trades off between the local affinity and the global structure (Carreira-Perpinán, 2010; Hie et al., 2020). Normally, the results of EE are relatively robust with the setting of λ within a certain range (Carreira-Perpinán, 2010; An et al., 2019; Chen et al., 2019). We use 1 as the

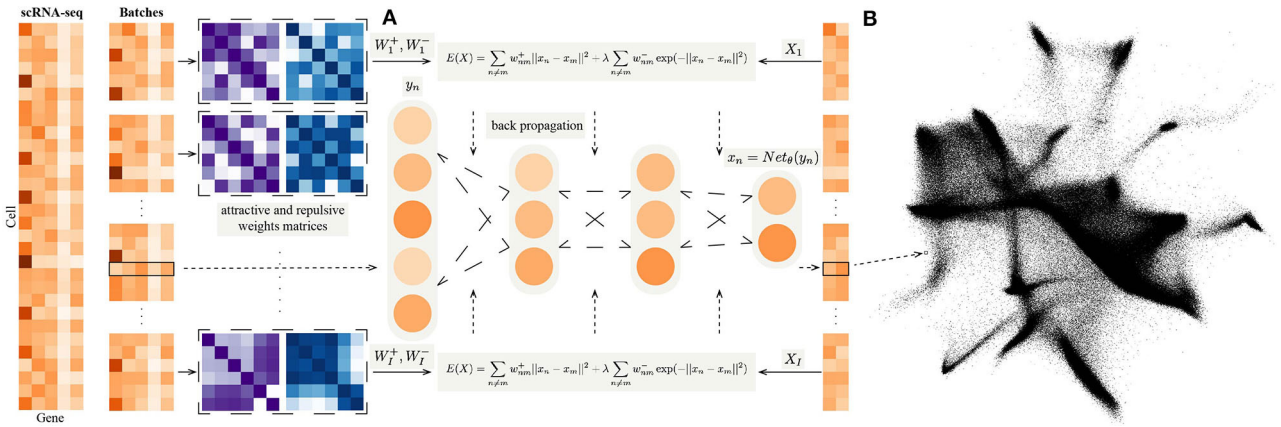

FIGURE 1 | (A) The flow chart of NeuralEE. In brief, NeuralEE constructs an NN that defines a parametric mapping from the original space to the embedded space. The full dataset is first randomly partitioned into several batches (only one batch is also acceptable, which means not applied with mini-batch trick or stochastic optimization). On each batch the attractive weight and the repulsive weight matrices are calculated and fed into the loss function of EE, which is represented as a composite function of the original data. By backpropagation algorithm, the parameters in the NN are optimized, and the mapping of the embedding is thus learned. **(B)** Visualization results of 1.3 million mouse brain cells by NeuralEE-SO.

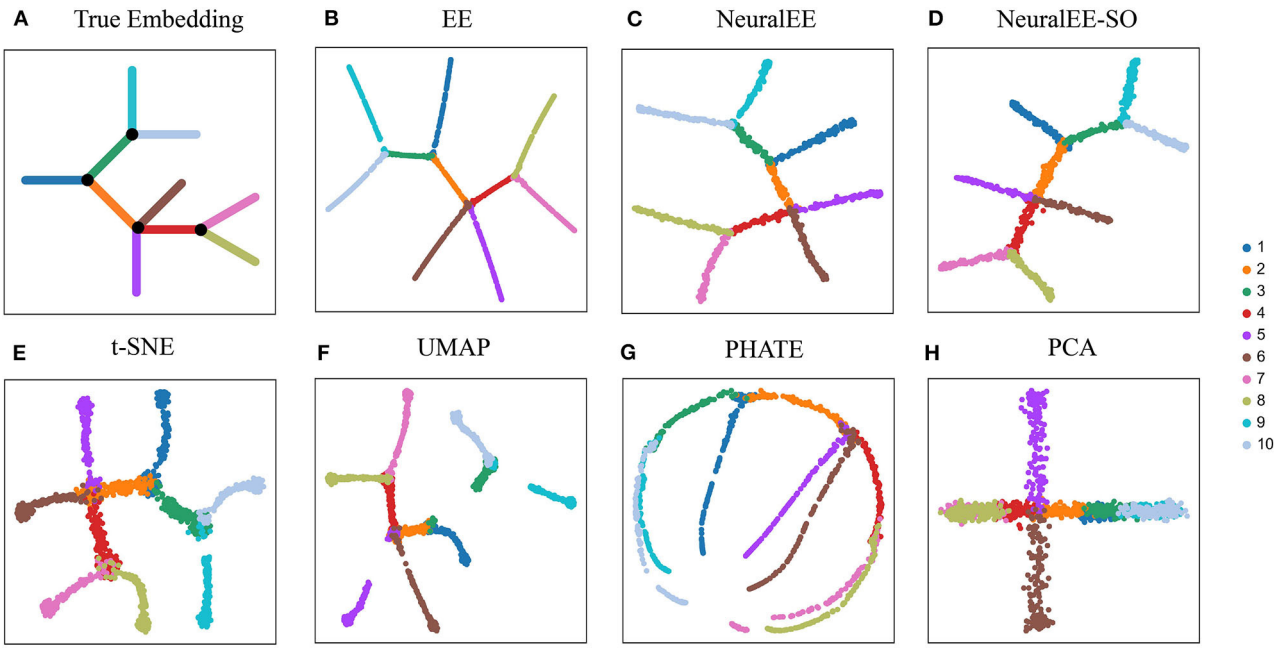

FIGURE 2 | Comparison of NeuralEE to other visualization methods on **ArtificialTree** data. **(A)** True embedding. **(B)** EE. **(C)** NeuralEE. **(D)** NeuralEE-SO. **(E)** t-SNE. **(F)** UMAP. **(G)** PHATE. **(H)** PCA. Different color corresponds different branch of artificial tree.

default setting of λ in this study; however, there should be little influence in the resulting visualization with λ ranging from 1 to 10 (**Supplementary Figure 2**). We have used the top 500 variable genes in the above visualizations of real datasets. We also demonstrate that when enough variation retained, rising the number of initializing gene do not alter the main structure of the visualization (**Supplementary Figure 3**). In practice, one may also choose to use another dimensional reduction method to extract the main features in advance to visualization (Lin et al., 2020; Wu and Zhang, 2020). The dimensionality reduction method can be in linear form such as PCA (Kobak and Berens, 2019) or in non-linear form such as scVI (Lopez et al., 2018; Wu and Zhang, 2020). We compare the visualization performance of NeuralEE with FIt-SNE and UMAP based on 50 PCs or latent variables learned by scVI, and we show that NeuralEE still preserves global structure better than t-SNE and UMAP (**Supplementary Figure 4**).

3.2. NeuralEE Is Generalizable to Newly Generated Data

Another important feature of NeuralEE, owing to the parametric framework, is its generalizability, i.e., ability of mapping

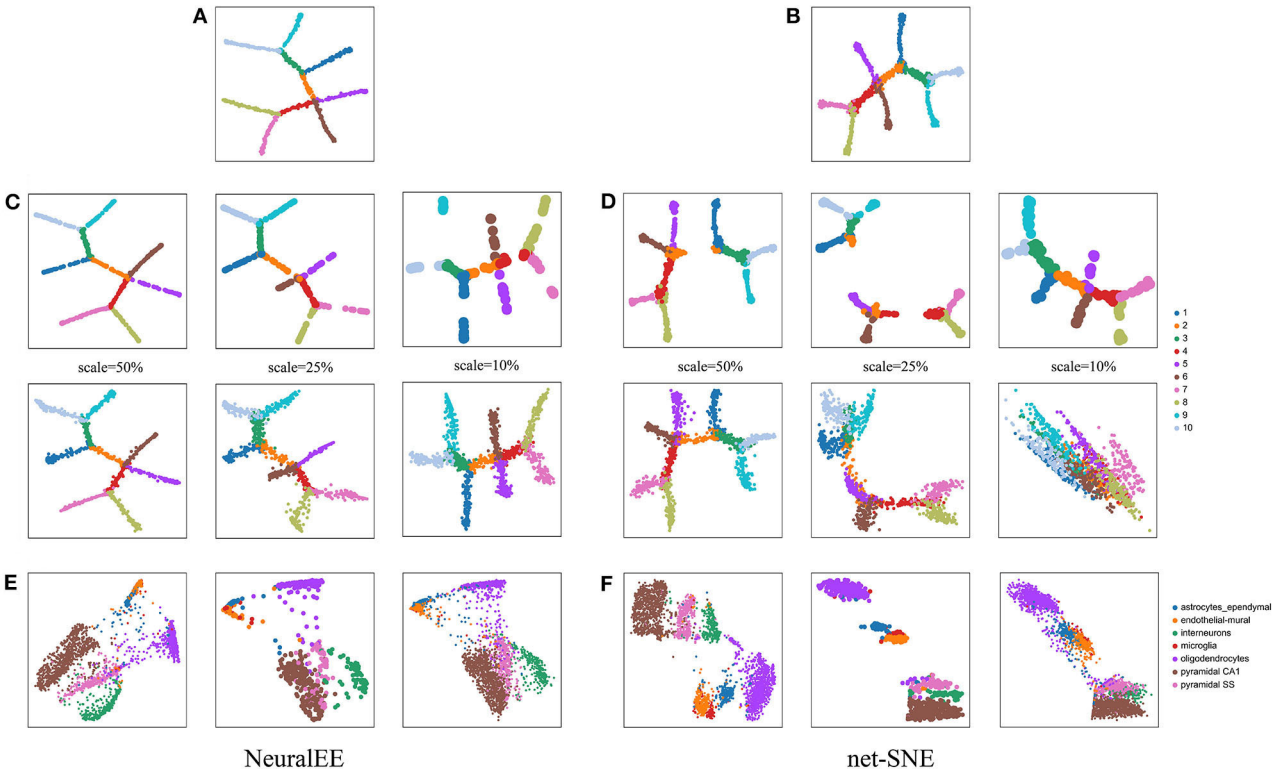

FIGURE 3 | (A) NeuralEE or **(B)** net-SNE on the full **ArtificialTree** data. Their embeddings with stochastic optimization are showed at **Supplementary Figure 6**. **(C)** The top line panel is the NeuralEE based on the sub-samples with a sub-sampling scale as the index *scale*, and the bottom line panel is the mapping of all samples to the embedded space based on the NN trained on the top panel. **(D)** net-SNE under the similar experiments as **(C)**. **(E)** From left to right is the NeuralEE on the entire **CORTEX** data, NeuralEE based on the sub-samples with a sub-sampling scale of 25%, and the mapping of all samples to the embedded space based on the NN trained on sub-samples. **(F)** net-SNE under the similar experiments as **(E)**.

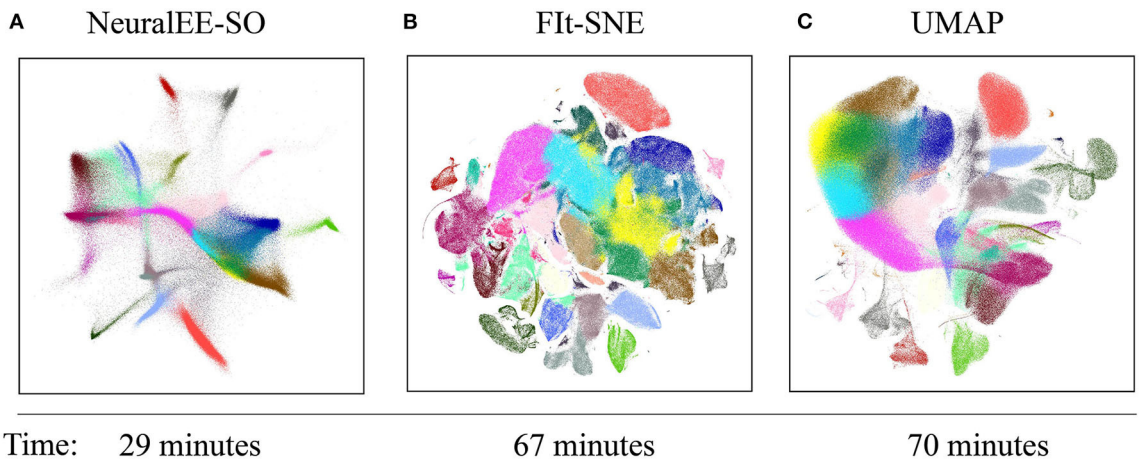

FIGURE 4 | Embedding results on 1.3 million mouse brain cells dataset. **(A)** NeuralEE-SO with batch size as 5,000. **(B)** FIt-SNE. **(C)** UMAP. Labels represent different clusters of Louvain community detection algorithm (Blondel et al., 2008). As NeuralEE belongs to parametric methods, when applied on million-size data, it could return several outliers on the embedded space, we manually delete them to make layout tighter. The raw embedding is showed at **Supplementary Figure 7**.

newly observed points from the original expression space to the embedded space (Cho et al., 2018). To validate the generalizability, we compare NeuralEE with net-SNE. We first apply and train NeuralEE on a subset of samples (a quarter sample in size). With the trained NN, we then directly map the left-out samples onto the embedded space and compare the

visualizations. We test on three scales of sub-sampling using 10%, 25%, and 50% of the original sample size. **Figures 3A,B** shows the embedding trained on the full dataset with NeuralEE and net-SNE. The top line panels of **Figures 3C,D** are the visualization of the embeddings based on the sub-samples. The bottom line panels of **Figures 3C,D** are the mapping of all samples to the embedded space based on the trained NN that corresponds top panels. We see that even when training on 10% of samples and with the subsample visualization shattered into broken clusters, the mapping visualization of NeuralEE (the right column in **Figure 3C**) is still rather robust and consistent with the result at **Figure 3A**. In contrast, the mapping visualization of net-SNE is only comparable to its full sample visualization on a 50% scale. Explicit distortions to the true structure are observed with mappings based on 10% and 25% subsample trainings (**Figure 3D**). We also test the generalizability on real biological data by setting the sub-sampling scale to 25%. On **HEMATO**, **PBMC**, and **RETINA** datasets, both NeuralEE and net-SNE show consistent results of mapping visualization to full sample embedding (**Supplementary Figure 5**). On the **CORTEX** data, NeuralEE performs well; however, net-SNE has a poor subsample embedding and, thus, an unsatisfactory mapping visualization (**Figures 3E,F**).

3.3. NeuralEE Is Scalable and Efficient

Next, we demonstrate the scalability of NeuralEE, which is another desired property with ever-growing data size nowadays. We apply NeuralEE, UMAP, and FIt-SNE to the **BRAIN-LARGE** dataset. Since there are over 25,000 genes in the **BRAIN-LARGE** dataset, we apply PCA initialization to the data; that is, we retain the top 50 PCs for further analysis. As the data is huge, we will apply NeuralEE-SO instead to reduce memory consumption. In this case, we set the batch size to 5,000 cells.

Figure 4 shows comparable results among the three methods. The colors are annotated according to Wolf et al. (2018). All methods visualize clusters clearly, with the clusters arranged by NeuralEE-SO in more connected manner, while in more separate and shatter layout manner by FIt-SNE. Besides the visualization, NeuralEE-SO also performs more efficiently, which only takes 29 min to visualize these 1.3 million mice brain cells, and the running times for FIt-SNE and UMAP are 67 and 70 min, respectively.

4. DISCUSSION

We develop NeuralEE, a GPU-accelerated dimensionality reduction method for visualization of large-scale scRNA-seq data.

NeuralEE applies a NN framework to parameterize the embedding, where the coordinate of data point in the original space is mapped to the embedded space. The mapping function thus trained allows NeuralEE to be generalizable, where the newly observed data can be directly mapped to the embedded manifold based on its features in the original space. After training NerualEE on a small set of sub-samples, the mapping visualization of remaining samples is comparable to the result of display based on full sample optimization.

We also provide a version of NeuralEE with application of the mini-batch trick. This is especially useful when dealing with large-scale dataset since a significant portion of cells are redundant in large-scale scRNA-seq data as experimentally demonstrated by Cho et al. (2018). In this way, the attractive and repulsive weights in the loss function of NeuralEE are optimized within each random batch which will greatly reduce the memory consumption and make NeuralEE scalable to datasets with millions of samples. Although we show that the batch size has minor effect on the resulting embedding, theoretically however, mini-batch trick applied on NeuralEE cannot guarantee the performance of the embedding, specifically on data with median or small size. As a result, we recommend using the mini-batch trick only when necessary, while in situations with small- or moderate-scale working with NeuraEE instead of NeuralEE-SO might be more robust. We also offer the option of online learning in NeuralEE-SO, where, by forming the newly observed data as new batches, the trained NeuralEE model can be updated.

The deep-learning-based design has been more popularly introduced in dimensionality reduction methods for scRNA-seq data owning to its scalable property. The variational autoencoder proposes a distinct family of generative process where the high-dimensional observed data can be generated from the low-dimensional latent space. For instance, scVI follows Zero-Inflated Negative Binomial for generative process (Risso et al., 2018) and assumes a decomposable form for approximated posterior inference. The posterior distributions parameters of low-dimensional latent variables are inferred by the learned parametric mapping from NN. These methods are powerful and work well when data approximately fits the model assumptions (Wu and Zhang, 2020). NeuralEE, on the other hand, focuses on preserving the intrinsic structure both locally and globally of the data, by simply utilizing the EE objective function and taking the advantage of scalability and parametric property of NN. We believe that such design may extend the capacity of applying EE to larger scale data, well preserving good properties for embedding and visualization.

NeuralEE also takes the advantage of utilizing GPU to accelerate the optimization. The optimization of NeuralEE mainly consists of matrix computation, and it can therefore be dramatically accelerated if its computation can be parallelizable. Although some of these methods leverage the merits of parallel computation, however, the optimizers for t-SNE and most of its variant methods, including EE, are only applicable on CPU. The number of CPU cores on a personal computer or even on a computation server is limited and normally incomparable to that of a GPU chip. By applying a GPU-based code design, NeuralEE can be easily implemented on a workstation or personal computer equipped with a regular NVIDIA GPU chip. In our experiment, it takes only half an hour to visualize 1.3 million mice brain cells on a NVIDIA GeForce GTX 1080 Ti GPU device and 64G computer memory (Memory consumption is illustrated at **Supplementary Table 3**).

In summary, NeuralEE is a scalable, generalizable, and, more importantly, GPU-accelerated dimensionality reduction method for visualization of scRNA-seq data.

DATA AVAILABILITY STATEMENT

NeuralEE is freely available at https://github.com/HiBearME/NeuralEE and its corresponding documentation is available at https://neuralee.readthedocs.io. All datasets analyzed in this paper are public. **CORTEX**, **HEMATO**, **PBMC**, **RETINA** and **BRAIN-LARGE** can be referenced at https://github.com/romain-lopez/scVI-reproducibility, and **ArtificialTree** can be referenced at https://github.com/KrishnaswamyLab/PHATE.

AUTHOR CONTRIBUTIONS

LM, LW, and FG conceived the NeuralEE models. JX, LM, and LW designed the NeuralEE models and performed the simulation study and real data set analysis. JX programmed the NeuralEE package. LM, LW, and JX wrote the whole manuscript. All authors read and approved the final manuscript.

FUNDING

This work was supported by the National Key R&D Program of China under Grant 2019YFA0709501, 2018YFB0704304, NSFC grants (Nos. 81673833, 11971459, and 12071466), NCMIS of CAS, LSC of CAS, and the Youth Innovation Promotion Association of CAS.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fgene.2020.00786/full#supplementary-material

REFERENCES

10x Genomics (2017). *Support: Single Cell Gene Expression Datasets*. 10x Genomics.

An, S., Ma, L., and Wan, L. (2019). TSEE: an elastic embedding method to visualize the dynamic gene expression patterns of time series single-cell RNA sequencing data. *BMC Genomics* 20:224. doi: 10.1186/s12864-019-5477-8

Becht, E., McInnes, L., Healy, J., Dutertre, C.-A., Kwok, I. W. H., Ng, L. G., et al. (2018). Dimensionality reduction for visualizing single-cell data using UMAP. *Nat. Biotechnol.* 37, 38–44. doi: 10.1038/nbt.4314

Blondel, V. D., Guillaume, J. L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *J. Stat. Mech.* 2008, 155–168. doi: 10.1088/1742-5468/2008/10/P10008

Carreira-Perpinán, M. A. (2010). "The elastic embedding algorithm for dimensionality reduction," in *27th International Conference on Machine Learning* (Haifa), Vol. 10, 167–174. Available online at: https://icml.cc/Conferences/2010/papers/123.pdf

Chen, Z., An, S., Bai, X., Gong, F., Ma, L., and Wan, L. (2019). DensityPath: an algorithm to visualize and reconstruct cell state-transition path on density landscape for single-cell RNA sequencing data. *Bioinformatics* 35, 2593-2601. doi: 10.1093/bioinformatics/bty1009

Cho, H., Berger, B., and Peng, J. (2018). Generalizable and scalable visualization of single-cell data using neural networks. *Cell Syst.* 7, 185–191. doi: 10.1016/j.cels.2018.05.017

Deng, Y., Bao, F., Dai, Q., Wu, L. F., and Altschuler, S. J. (2019). Scalable analysis of cell-type composition from single-cell transcriptomics using deep recurrent learning. *Nat. Methods* 16, 311–314. doi: 10.1038/s41592-019-0353-7

Ding, J., Condon, A., and Shah, S. P. (2018). Interpretable dimensionality reduction of single cell transcriptome data with deep generative models. *Nat. Commun.* 9:2002. doi: 10.1038/s41467-018-04368-5

Eraslan, G., Simon, L. M., Mircea, M., Mueller, N. S., and Theis, F. J. (2019). Single-cell RNA-seq denoising using a deep count autoencoder. *Nat. Commun.* 10:390. doi: 10.1038/s41467-018-07931-2

Hie, B., Peters, J., Nyquist, S. K., Shalek, A. K., Berger, B., and Bryson, B. D. (2020). Computational methods for single-cell RNA sequencing. *Annu. Rev. Biomed. Data Sci.* 3, 339–364. doi: 10.1146/annurev-biodatasci-012220-100601

Hinton, G., and Roweis, S. (2003). Stochastic neighbor embedding. *Adv. Neural Inform. Process. Syst.* 15, 857–864.

Kingma, D. P., and Welling, M. (2014). "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, eds Y. Bengio, and Y LeCun (Banff, AB). Available online at: https://dblp.org/db/conf/iclr/iclr2014.html

Kobak, D., and Berens, P. (2019). The art of using T-SNE for single-cell transcriptomics. *Nat. Commun.* 10:5416. doi: 10.1038/s41467-019-13056-x

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature* 521:436. doi: 10.1038/nature14539

Lecun, Y., Boser, B. E., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1990). Handwritten digit recognition with a back-propagation network. *Adv. Neural Inform. Process. Syst.* 2, 396–404.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1991). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Netw.* 6, 861–867. doi: 10.1016/S0893-6080(05)80131-5

Lin, E., Mukherjee, S., and Kannan, S. (2020). A deep adversarial variational autoencoder model for dimensionality reduction in single-cell RNA sequencing analysis. *BMC Bioinformatics* 21:64. doi: 10.1186/s12859-020-3401-5

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. *Nat. Methods* 16, 243–245. doi: 10.1038/s41592-018-0308-4

Lopez, R., Regier, J., Cole, M. B., Jordan, M. I., and Yosef, N. (2018). Deep generative modeling for single-cell transcriptomics. *Nat. Methods* 15, 1053–1058. doi: 10.1038/s41592-018-0229-2

Moon, K. R., van Dijk, D., Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., et al. (2019). Visualizing structure and transitions in high-dimensional biological data. *Nat. Biotechnol.* 37, 1482–1492. doi: 10.1038/s41587-019-0336-3

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, eds H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc.), 8026–8037. Available online at: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S., and Vert, J.-P. (2018). A general and flexible method for signal extraction from single-cell RNA-seq data. *Nat. Commun.* 9:284. doi: 10.1038/s41467-017-02554-5

Shekhar, K., Lapan, S. W., Whitney, I. E., Tran, N. M., Macosko, E. Z., Kowalczyk, M., et al. (2016). Comprehensive classification of retinal bipolar neurons by single-cell transcriptomics. *Cell* 166, 1308–1323. doi: 10.1016/j.cell.2016.07.054

Svensson, V., Vento-Tormo, R., and Teichmann, S. A. (2018). Exponential scaling of single-cell RNA-seq in the past decade. *Nat. Protoc.* 13, 599–604. doi: 10.1038/nprot.2017.149

Tusi, B. K., Wolock, S. L., Weinreb, C., Hwang, Y., Hidalgo, D., Zilionis, R., et al. (2018). Population snapshots predict early haematopoietic and erythroid hierarchies. *Nature* 555, 54–60. doi: 10.1038/nature25741

van der Maaten, L., and Hinton, G. (2008). Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9, 2579–2605.

Vladymyrov, M., and Carreira-Perpinan, M. (2012). "Partial-hessian strategies for fast learning of nonlinear embeddings," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, eds J. Langford, and J. Pineau (Edinburgh: Omnipress), 345–352, Available online at: https://icml.cc/Conferences/2012/papers/199.pdf

Vladymyrov, M., and Carreira-Perpinan, M. (2013). "Entropic affinities: properties and efficient numerical computation," in *30th International Conference on Machine Learning* (Atlanta, GA), 477–485. Available online at: http://proceedings.mlr.press/v28/vladymyrov13.pdf

Wang, D., and Gu, J. (2018). VASC: Dimension reduction and visualization of single-cell RNA-seq data by deep variational autoencoder. *Genomics Proteomics Bioinformatics* 16, 320–33. doi: 10.1016/j.gpb.2018.08.003

Wolf, F. A., Angerer, P., and Theis, F. J. (2018). SCANPY: large-scale single-cell gene expression data analysis. *Genome Biol.* 19:15. doi: 10.1186/s13059-017-1382-0

Wu, Y., and Zhang, K. (2020). Tools for the analysis of high-dimensional single-cell RNA sequencing data. *Nat. Rev. Nephrol.* 16, 408–421. doi: 10.1038/s41581-020-0262-0

Zeisel, A., Munozmanchado, A. B., Codeluppi, S., Lonnerberg, P., La Manno, G., Jureus, A., et al. (2015). Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science* 347, 1138–1142. doi: 10.1126/science.aaa1934

Zheng, G. X. Y., Terry, J. M., Belgrader, P., Ryvkin, P., Bent, Z. W., Wilson, R., et al. (2017). Massively parallel digital transcriptional profiling of single cells. *Nat. Commun.* 8:14049. doi: 10.1038/ncomms14049

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.