entropy

Article

A Preprocessing Manifold Learning Strategy Based on t-Distributed Stochastic Neighbor Embedding

Sha Shi [1], Yefei Xu [1,*,†], Xiaoyang Xu [1], Xiaofan Mo [2] and Jun Ding [3,*]

[1] State Key Laboratory of Integrated Services Network, Xidian University, 2 South TaiBai Road, Xi'an 710071, China; sshi@xidian.edu.cn (S.S.); xyxu510@stu.xidian.edu.cn (X.X.)
[2] National Astronomical Observatories, Chinese Academy of Sciences, 20A Datun Road, Chaoyang District, Beijing 100101, China; moxiaofan@nao.cas.cn
[3] Institute of Information Sensing, Xidian University, 2 South TaiBai Road, Xi'an 710071, China
* Correspondence: yefei.xu@sap.com (Y.X.); dingjun@xidian.edu.cn (J.D.)
† Current address: SAP Labs, Shanghai 201203, China.

Abstract: In machine learning and data analysis, dimensionality reduction and high-dimensional data visualization can be accomplished by manifold learning using a t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm. We significantly improve this manifold learning scheme by introducing a preprocessing strategy for the t-SNE algorithm. In our preprocessing, we exploit Laplacian eigenmaps to reduce the high-dimensional data first, which can aggregate each data cluster and reduce the Kullback–Leibler divergence (KLD) remarkably. Moreover, the k-nearest-neighbor (KNN) algorithm is also involved in our preprocessing to enhance the visualization performance and reduce the computation and space complexity. We compare the performance of our strategy with that of the standard t-SNE on the MNIST dataset. The experiment results show that our strategy exhibits a stronger ability to separate different clusters as well as keep data of the same kind much closer to each other. Moreover, the KLD can be reduced by about 30% at the cost of increasing the complexity in terms of runtime by only 1–2%.

Keywords: manifold learning; t-SNE; dimensionality reducing; k-nearest neighbor

Citation: Shi, S.; Xu, Y.; Xu, X.; Mo, X.; Ding, J. A Preprocessing Manifold Learning Strategy Based on t-Distributed Stochastic Neighbor Embedding. *Entropy* **2023**, *25*, 1065. https://doi.org/10.3390/e25071065

Academic Editor: Philip Broadbridge

Received: 24 April 2023
Revised: 1 July 2023
Accepted: 5 July 2023
Published: 14 July 2023

1. Introduction

In machine learning and other computer-related areas, the demands for dimensionality reduction methods never vanish owing to the curse of dimensionality [1–3]. Generally, the amount of calculation often grows exponentially with the increase in dimensionality, hence the efficiency of machine learning algorithms will drop markedly if the dimension of the input data is enormous [2,4]. On account of the limited computing power at present, it is essential to devise dimensionality reduction methods to obtain a sound and reliable result. On the other hand, in many realms, it is also of great interest to reduce high-dimensional data to two or three dimensions for visualization purposes [5–9].

For decades, a large number of dimensionality reduction methods have been applied to different tasks, among them are Principal Component Analysis (PCA) [10–12], Multidimensional Scaling (MDS) [13,14], Sammon Mapping [15], Isomap [16], Locally Linear Embedding (LLE) [17], Laplacian Eigenmaps (LE) [18–20], t-Distributed Stochastic Neighbor Embedding (t-SNE) [21–24] and so on. It is well known that the first three algorithms mentioned above are linear dimensionality reduction methods, which usually break the inner data structure of real-world datasets, thus yielding a poor visualization map. The others are non-linear and can be concluded to be manifold learning algorithms.

Manifold learning tends to outperform linear dimensionality reduction methods in data visualization. Particularly, t-SNE is amongst the best-known manifold learning algorithms, as it can not only capture much of the local structure of the high-dimensional data but also reveal the associated global structure, such as by presenting clusters at many different scales [21].

As unprecedented as the performance of t-SNE is, however, several problems still remain to be addressed. Firstly, owing to the assumption in t-SNE that the high-dimensional data are in Gaussian distribution, the distribution of the mapped data in low dimensions is always uniform and loose, and the Kullback–Leibler divergence (KLD) often converges to a high value, which prevents the algorithm from generating a sound low-dimensional map. Secondly, separations between different natural clusters still need to be improved as some of the data are often inclined to be clustered into the wrong groups due to the obscure boundary. Thirdly, both the computation and space complexity of t-SNE increase quadratically with the number of data pairs, which severely limits the application of t-SNE on large datasets in reality.

In this paper, we significantly improve the standard t-SNE scheme by developing a preprocessing strategy for it. In our preprocessing strategy, Laplacian eigenmaps (LE) are first employed on the high-dimensional data. Thus, before they are input into t-SNE, each data cluster can be aggregated first and the data are no longer in Gaussian distribution.

In addition, aiming at magnifying the gaps between different clusters and enlarging the difference between data of different kinds, the K-nearest-neighbor (KNN) algorithm is also introduced into our preprocessing to shrink the Euclidean distance between each neighboring data pair. Moreover, compared to the standard t-SNE, KNN is also expected to reduce the computation and space complexity as only the neighboring data pairs are considered in our strategy, which can offer a balance between performance and complexity.

We apply our method on the MNIST dataset, which contains 70,000 handwritten numeric images that are 28 pixels by 28 pixels in size. The training set contains 60,000 images, while the test set 10,000 images. The experimental results show that our strategy can significantly improve the performance of the standard t-SNE and the recoveries of low-dimensional data structures are also reinforced, while the overall complexity only increases by about 1–2%.

The outline of this paper is as follows: Section 2 gives a quick review of the basic idea of the standard t-SNE. In Section 3, a preprocessing manifold learning strategy based on t-SNE, LE and KNN is proposed. The numerical results on the MNIST dataset are presented in Section 4. Finally, we draw some conclusions in Section 5.

2. Manifold Learning by Using a t-SNE

Generally speaking, dimensionality reduction methods convert the high-dimensional dataset $X = \{x_1, x_2, \cdots, x_n\}$ into two- or three-dimensional data $Y = \{y_1, y_2, \cdots, y_n\}$ that can be displayed in a scatterplot. It is argued in [25] that a set of similar data is neither randomly nor uniformly distributed in this space, but instead lies on or near a submanifold of much lower dimension. Manifold learning is a sort of non-linear technique that aims to find a non-linear mapping to extract the intrinsic dimensionality of the original data and to realize dimensionality reduction.

For traditional dimensionality reduction methods, such as the Locally Linear Embedding algorithm, the similarity between data is typically modeled by Euclidean distance. Thus, it is difficult for those traditional dimensionality reduction methods to unfold "many-to-one" mappings, in which a single ambiguous object really belongs in several disparate locations in the low-dimensional space [26]. To solve the problem, the t-SNE algorithm proposed by Maaten and Hinton [21] employs a probabilistic model to visualize the structure of complex datasets. Specifically, t-SNE converts high-dimensional Euclidean distances between data points into joint probabilities to characterize the similarities between data.

In t-SNE, the conditional probabilities $p_{j|i}$ that a data point $\mathbf{x}_i$ would pick $\mathbf{x}_j$ as its neighbor is given by [27]

$$p_{j|i} = \frac{exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{j\neq i} exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)} \tag{1}$$

with $p_{i|i} = 0$. The variance σ_i in the Gaussians centered around x_i is determined by a binary search procedure. The data density is likely to vary. Thus, in dense regions, a smaller value of σ is more appropriate than that in sparse regions.

Let P_i be the conditional probability distribution over all other data points at a given point $\mathbf{x}_i$. The entropy of P_i will grow along with the increase in σ_i. Then, with a fixed perplexity specified by the user, a binary search for σ_i is performed in t-SNE to produce a probability distribution P_i. In a sense, the perplexity can be interpreted as a smooth measure of the effective number of neighbors. The performance of t-SNE is fairly robust to changes in the perplexity, which is typically between 5 and 50, depending on the size of the datasets. The joint probabilities p_{ij} can be obtained easily as follows:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2}, \tag{2}$$

which ensures that $\sum_j p_{ij} > \frac{1}{2n}$ for all data points $\mathbf{x}_i$. Thus, each data point $\mathbf{x}_i$ can make a significant contribution to the cost function [21].

In the lower dimension, t-SNE employs a Student t-distribution with one degree of freedom as the heavy-tailed distribution to separate different clusters from each other. The joint probabilities of map points are given by [27]

$$q_{j|i} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_{k \neq l} exp(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}. \tag{3}$$

KLD is typically adopted to characterize the mismatch between p_{ij} and $q_{j|i}$. t-SNE minimizes the sum of KLD over all data points using a gradient descent method. The cost function C and the gradient of t-SNE are given by [27]

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log \frac{p_{ij}}{q_{ij}}, \tag{4}$$

and

$$\frac{\delta C}{\delta \mathbf{y}_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}, \tag{5}$$

respectively. Once the KLD decreases to an appropriate value, a faithful low-dimensional map will be obtained.

3. The Proposed Preprocessing Manifold Learning Strategy

We noticed that the drawbacks of the t-SNE mentioned above are partly caused by the Gaussian distribution of the high-dimensional data. In other words, due to the algorithm design of t-SNE, the mapped data will be uniformly but loosely distributed in a low dimension given that the high-dimensional data are in Gaussian distribution. Thus, a natural way to handle such a problem is to modify the original data distribution in advance in a reasonable way to fit the standard t-SNE. By reasonable, we mean the updated data distribution tailored to t-SNE also inherits the characteristics of the original data distribution.

Laplacian eigenmaps is another efficient manifold learning method, which maps the high-dimensional data into a lower dimension by solving a generalized eigenvector problem [19]. It shares some similar properties with LLE, e.g., it also employs weights rather than the possibility to realize dimensionality reduction, hence a tighter map compared with t-SNE could be obtained. Thus, the idea of LE can be introduced here to preprocess the original high-dimensional data. Following the dimensionality reduction strategy of LE [28], we first find the k-nearest neighbors for each data point by using KNN and characterize their relations with W_{ij} as follows:

$$W_{ij} = \begin{cases} e^{-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{t}} & \text{if } i,j \text{ are neighbors,} \\ 0 & \text{otherwise.} \end{cases}$$

Then the cost function is given by

$$\sum_{i=1}^{N} \sum_{j=1}^{N} ||\mathbf{y}_i - \mathbf{y}_j||^2 W_{ij}. \tag{6}$$

To minimize the cost function, there are three matrices **D**, **W** and **L** defined in LE, in which **D** is a diagonal matrix where $D_{ii} = \sum_j W_{ij}$. **W** is composed of W_{ij}, where W_{ij} is just the element at row i and column j. The Laplacian matrix [28] is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{7}$$

Thus, the first dimensionality reduction can be accomplished by solving a generalized eigenvector problem, i.e.,

$$\mathbf{L}\mathbf{y} = \lambda \mathbf{D}\mathbf{y}, \tag{8}$$

and the m-dimensional mapped data are corresponding to the smallest m non-zero eigenvalues in Equation (8) [28].

It is easy to see that by using LE the data points can be aggregated more tightly with their neighbors and the mapped data are not in Gaussian distribution anymore. Thus, if we take the mapped data that were preprocessed by LE as the input of t-SNE, the loose distribution problem associated with t-SNE can be significantly alleviated.

In Figures 1 and 2, we show the visualization of 5000 data selected from the MNIST randomly by using t-SNE and t-SNE with LE as the preprocessing strategy, respectively. The procedure of the latter strategy is as follows: We first use KNN to obtain k_0 neighbors for each data point; then, we reduce the dimensionality of the original data to N_0 using the concept of LE. Finally, we take those N_0-dimensional data as the input for t-SNE to further accomplish the dimensionality reduction.

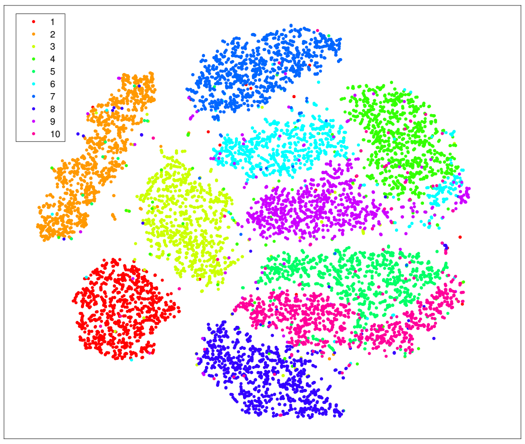

Figure 1. Visualization of 5000 data randomly selected from MNIST by using t-SNE, the iteration number is 500.

Just as we expect, the t-SNE implements t-distribution to solve the crowd problem and a relatively obvious gap between different data clusters can be formed. However, the clusters themselves are still loose to some extent, which undermines the ability of t-SNE to form tight clusters. Worse yet, the entropy of each cluster is too high to yield a small K–L divergence. On the other hand, from Figure 2, it is evident that not only the gaps between each of the ten data clusters can be formed by using the preprocessing strategy, but each data point also tends to be gathered with its neighboring points.

However, we notice that the gaps between different clusters are not satisfactory, both for the standard t-SNE and the one with LE as the preprocessing strategy, as shown in Figures 1 and 2. Moreover, as both the computation and space complexity of t-SNE grow quadratically with the number of data pairs, it will be of great interest to reduce the number of data pairs involved in the final t-SNE.

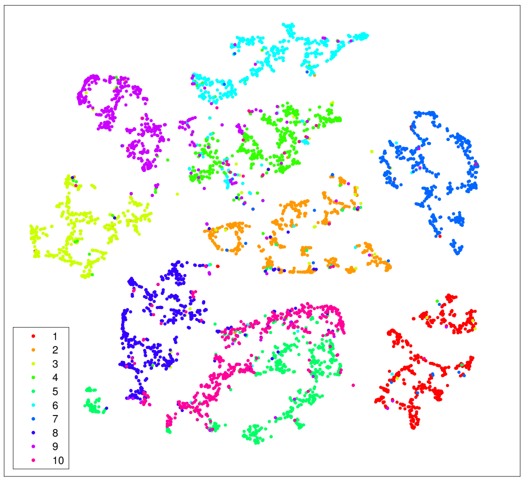

Figure 2. Visualization of 5000 data randomly selected from MNIST by using t-SNE with LE as the preprocessing strategy. Here we take $k_0 = 40$ and $N_0 = 80$. The iteration number of t-SNE is also 500.

Here, we continue to preprocess the data that experienced dimensionality reduction by LE before they are processed by t-SNE. Inspired by the sparse approximation strategy proposed in [29], we again use the KNN algorithm to find out the neighbors for each data point, then aggregate neighboring similar data points and weaken the relationships between dissimilar data pairs. As in [29], the pairwise similarities between data points, p_{ij}, is redefined by

$$p_{j|i} = \begin{cases} \frac{exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)}{\sum_{j\neq i} exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/2\sigma_i^2)} & \text{if } i,j \text{ are neighbors} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In this way, we propose another strategy by which we can decrease the distance between neighboring data in order to increase the possibility that a data point chooses its real neighbor.

After we implement LE, we perform the KNN algorithm again to find the neighbors of each data point, and then we introduce a coefficient α as below

$$d_{ij} = \begin{cases} \alpha||\mathbf{x}_i - \mathbf{x}_j||^2 & \text{if } i,j \text{ are neighbors} \\ \frac{1}{\alpha}||\mathbf{x}_i - \mathbf{x}_j||^2 & \text{otherwise} \end{cases} \tag{10}$$

It makes sense and does little harm to the original data structure since we have already implemented Laplacian eigenmaps to preprocess the data. With regard to the MNIST dataset, we set α as 1×10^{-5}.

Now, we present our dimensionality reduction strategy with preprocessing as shown in Algorithm 1.

Algorithm 1 A preprocessing manifold learning strategy based on t-SNE.

```
Input: The high-dimensional data set X.
Output: The two-dimensional data Y after the dimensionality reduction.
 1: Compute the Euclidean distances for each of the high-dimensional data pairs;
 2: Apply a KNN algorithm to find out $k_0$ neighbors for each data;
 3: Apply Laplacian Eigenmaps on the original data and reduce its dimensionality to $N_0$;
 4: Apply the KNN algorithm again to find out $k_1$ neighbors for each data, and for two
    data points that are not neighbors $i, j$, set their conditional probability $p_{j|i}$ to 0;
 5: Decrease the distance between neighboring data points by applying the coefficient $\alpha$ on
    the neighboring data points;
 6: Compute the joint probabilities with $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2}$;
 7: Get the low-dimensional mapped data $Y^{(0)}$;
 8: Compute the joint probabilities of the mapped points with $q_{j|i} = \frac{(1+||\mathbf{y}_i-\mathbf{y}_j||^2)^{-1}}{\sum_{k\neq l} exp(1+||\mathbf{y}_k-\mathbf{y}_l||^2)^{-1}}$;
 9: Compute $\frac{\delta C}{\delta \mathbf{y}_i} = 4\sum_j (p_{ij} - q_{ij})(\mathbf{y}_i - \mathbf{y}_j)(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}$;
10: Compute $Y^{(t)} = Y^{(t-1)} + \eta\frac{\delta C}{\delta \mathbf{Y}} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$, where $\eta$ represents the learning
    rate and $\alpha(t)$ the momentum;
11: Repeat steps 8, 9 and 10 until $t$ reach the number of iterations to implement t-SNE
    on the pre-processed data or the corresponding dimensionality of the mapped data is
    reduced to be 2.
```

In Figure 3, we show the visualization of 5000 data points selected from MNIST randomly with our preprocessing strategy. The procedure of our preprocessing strategy is as follows: first, we use KNN to obtain k_0 neighbors for each data point; then, we apply LE to reduce the dimensionality of the original data; after that, we implement the KNN algorithm again to find k_1 neighbors for each data point; and finally, we apply t-SNE on the preprocessed data and reduce their dimensionality to 2.

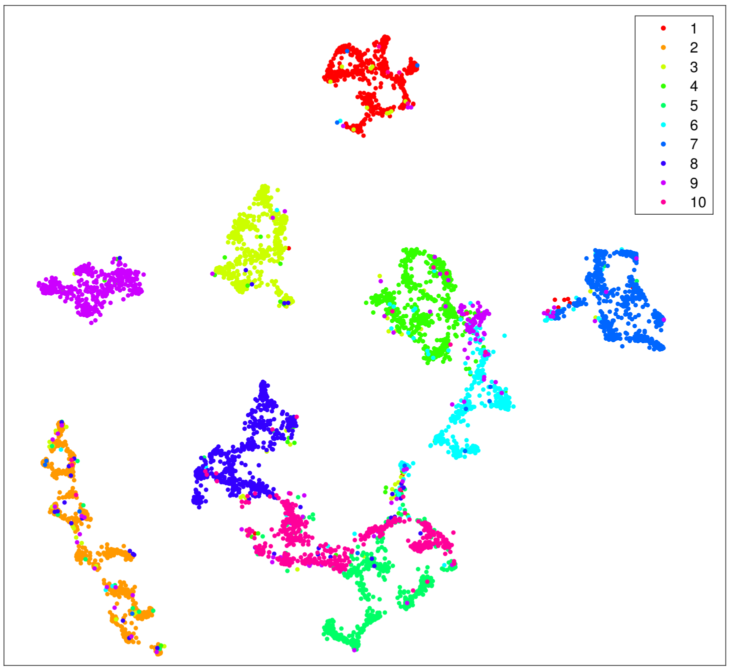

Figure 3. Visualization of 5000 data selected from MNIST randomly with our preprocessing strategy as shown in Algorithm 1. Here, we take $k_0 = k_1 = 40$, $N_0 = 80$ and the iteration number is 500. During the first 100 iteration of gradient descent, early exaggeration is exploited.

We have also applied our strategies on datasets Coil-100 and the Fashion-MNIST. The former is a collection of color images of 100 objects taken from different angles and comprises 72 images per object in total. The size of each image is uniformly processed to be 128 × 128. The latter is a clothing image dataset from Zalando, Berlin, Germany, containing a training set of 60,000 samples and a test set of 10,000 samples. Each sample is a 28 × 28 grayscale image associated with 10 categories of labels. The performance of the t-SNE algorithm and our strategy on these datasets are shown in Figures 4–7. According to the simulation results, our strategy has better performance on the MNIST dataset and the Fashion-MNIST dataset. For the Coil-100 dataset, though the performance of our strategy is slightly unsatisfactory due to the high dimensionality of the data, it is better than the standard t-SNE algorithm.

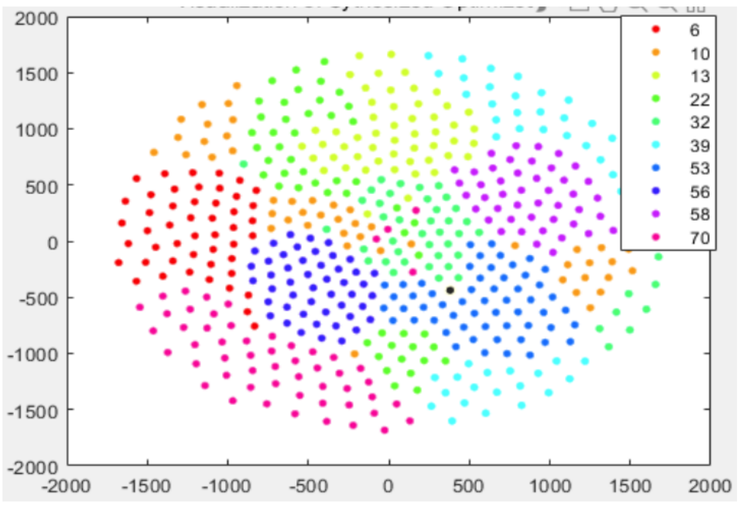

Figure 4. Visualization of 720 data from 10 objects randomly selected from Coil−100 by using t−SNE.

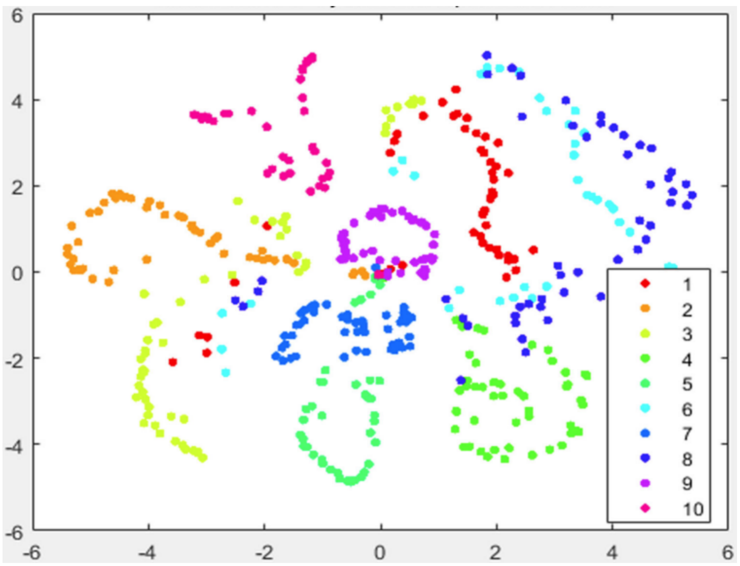

Figure 5. Visualization of 720 data from 10 objects randomly selected from Coil−100 with our preprocessing strategy.

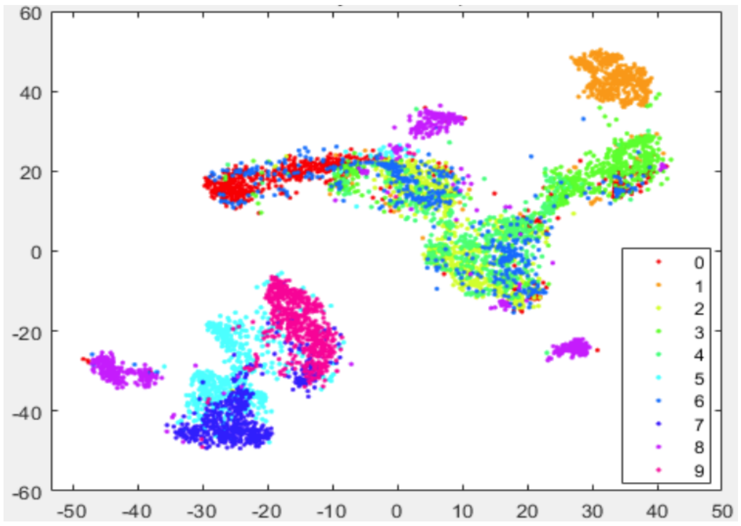

Figure 6. Visualization of 5000 data randomly selected from Fashion−MNIST by using t−SNE.

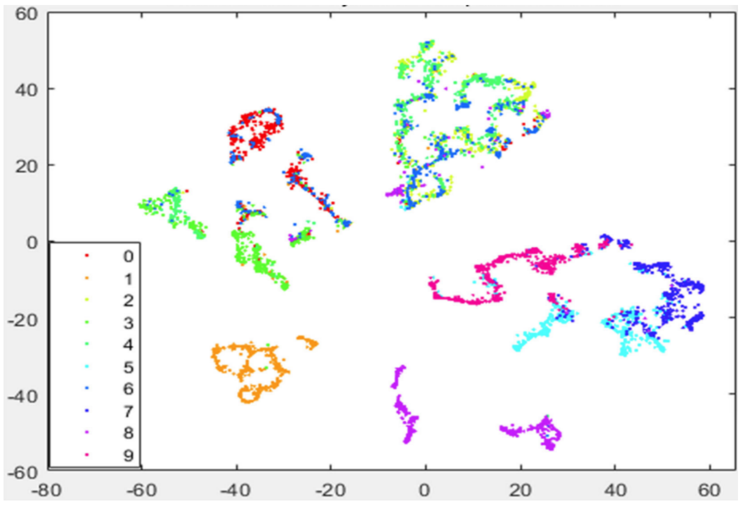

Figure 7. Visualization of 5000 data randomly selected from Fashion−MNIST with our preprocessing strategy.

4. Discussion

As in previous simulations, in Figure 3 we again show the visualization of 5000 data selected from MNIST randomly but by using our approach (Algorithm 1). It can be seen that, compared with Figure 2, our strategy shows more pronounced benefits since data of the same kind are aggregated far more tightly. In addition, the gaps between different clusters are also enlarged. Thus, it provides many more advantages for the process of extracting each cluster respectively.

Just as we mentioned in the last section, the strategy by which we decrease the distance between neighboring data destroys the data structure in some cases, leading to an increase in KLD to some extent. However, the KLD can be significantly reduced by using the idea of LE in step 3 in Algorithm 1. In Figure 8, we show the changing trend of KLD along with the increase in iterations. It is evident that by using our preprocessing strategy, the KLD plateaus around "1" after 500 iterations, while the standard t-SNE plateaus at around 1.5 and the SNE plateaus at 2.8. In other words, the KLD yielded by our approach is about 33% lower than the standard t-SNE and almost 66% lower than the SNE.

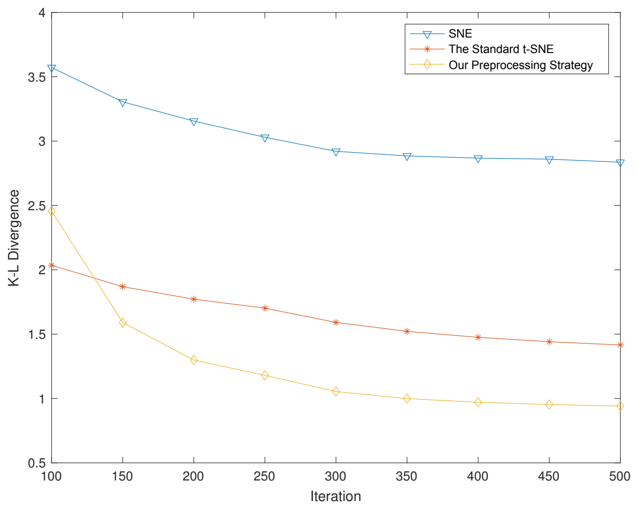

Figure 8. The gradient descent process of the SNE, the standard t-SNE and our preprocessing strategy.

We also compare the computation complexity of the three approaches mentioned in the context in terms of running time. As shown in Table 1, the running time caused by our approach is increased by only 1.82% when compared with the standard t-SNE. There are two reasons for this. Firstly, LE is a highly efficient algorithm; when it is introduced into our preprocessing strategy to accomplish the preliminary dimensionality reduction, the extra complexity can almost be ignored. Secondly, though KNN algorithms are used in our approach twice, the number of data pairs considered in t-SNE can also be reduced correspondingly. In other words, the extra complexity caused by KNN can be offset by itself.

Table 1. Average running time of SNE, t−SNE and our preprocessing approach on the visualization of 5000 data randomly selected from MNIST. The simulation platform is a laptop with Intel CPU i7−6700HQ.

Algorithm	**SNE**	**t-SNE**	**Our Preprocessing Strategy**
Running Time	530.09 s	278.62 s	283.71 s

For the space complexity, since we have added a preprocessing process to the standard t-SNE algorithm, some extra space is required, which is mainly caused by the Laplacian eigenmaps and the KNN algorithm. Firstly, for the KNN algorithm, each data point is stored as a separate object or array, and its space complexity is $O(n \times d)$, where n represents the number of data points and d represents the dimensionality of each data point. Secondly, Laplacian eigenmaps need to construct the graph Laplacian matrix, which is an $n \times n$ matrix, so the space complexity of Laplacian eigenmaps is $O(n^2)$. As the standard t-SNE needs to compute and store the joint probabilities of mapped points, its space complexity is $O(n^2)$. In conclusion, our strategy requires approximately twice as much space compared to the standard t-SNE algorithm.

To show the effect of different parameters k_0, k_1 and α on the performance of our algorithm, we performed simulations with different parameters and gave the corresponding gradient descent process. Since both k_1 and k_2 are the neighbor numbers of the KNN algorithm, we also made them consistent to maintain their consistency. In Figure 9, we show that a suitable parameter α can bring good performance to our strategy. In Figure 10, we show that the performance does become better along with the increase in k_0 and k_1, but at a cost of higher complexity as more neighbor nodes are involved. On the other hand, we find that when k_0 and k_1 are increased to 80, the performance reaches a saturated state.

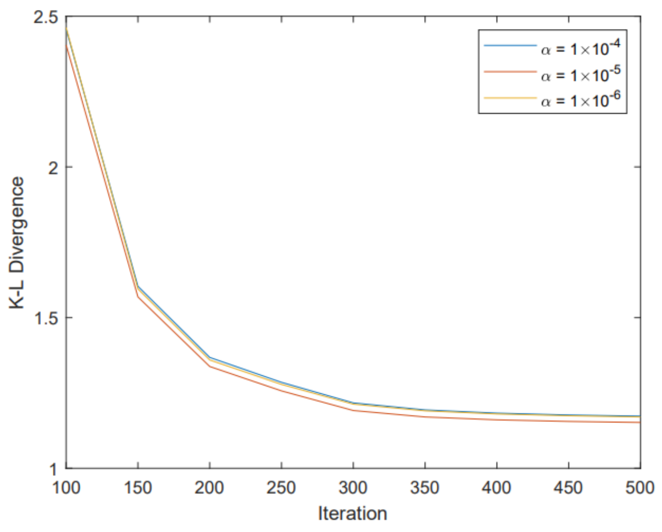

Figure 9. The gradient descent process of our preprocessing strategy when using different parameters α.

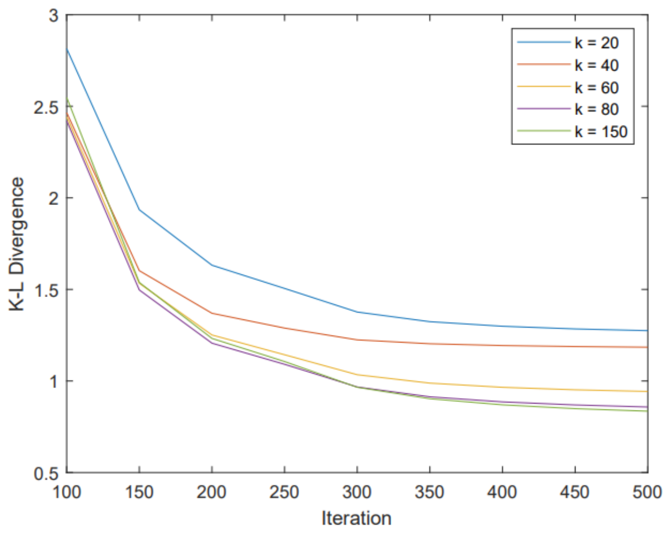

Figure 10. The gradient descent process of our preprocessing strategy when using different parameters k, where we let k = k_0 = k_1.

Moreover, as we can set those parameters (k_0, k_1 and N_0) according to different scenarios, our approach offers a flexible balance between complexity and performance as required. Now, we can safely arrive at the conclusion that our strategy significantly improves the performance of the standard t-SNE while the complexity almost remains the same.

5. Conclusions

We have developed a preprocessing t-SNE manifold learning algorithm whose preprocessing strategy employs the idea of LE to aggregate each data cluster and the KNN to enlarge the gaps between different clusters. Our approach significantly improves the capability of dimensionality reduction and high-dimensional data visualization of the standard t-SNE while the computation complexity remains almost the same.

We also want to point out that the parameters used in Algorithm 1 (such as k_0, k_1, α and N_0) are determined based on experience. Obviously, different parameters lead to different dimensionality reduction performances and also different complexities. More precisely, the larger the k_0 or k_1 the more complex the algorithm; on the other hand, if k_0

and k_1 are too small, the structure of the low-dimensional data may be distorted. Thus, it would be interesting to study how to set those parameters to an optimal value to balance performance and complexity. We will consider this as a part of our next work.

Author Contributions: Conceptualization, Y.X. and S.S.; methodology, S.S.; software, X.X.; validation, X.M. and J.D.; formal analysis, S.S.; investigation, Y.X.; data curation, X.X.; writing—original draft preparation, Y.X.; writing—review and editing, S.S.; visualization, X.X.; funding acquisition, J.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by NSFC (61771377, 61502376); the Key Research and Development Project of Shannxi Province (2023-YBGY-206); GuangDong Basic and Applied Basic Research Foundation (2023A1515010671); the Fundamental Research Funds for the Central Universities of China.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: No new data were created.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Keogh, E.; Mueen, A. Curse of dimensionality. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2011; pp. 257–258.
2. Anowar, F.; Sadaoui, S.; Selim, B. Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE). *Comput. Sci. Rev.* **2021**, *40*, 100378. [CrossRef]
3. Cheridito, P.; Jentzen, A.; Rossmannek, F. Efficient Approximation of High-Dimensional Functions with Neural Networks. *IEEE Trans. Neural Networks Learn. Syst.* **2022**, *33*, 3079–3093. [CrossRef] [PubMed]
4. An, P.; Wang, Z.; Zhang, C. Ensemble unsupervised autoencoders and Gaussian mixture model for cyberattack detection. *Inf. Process. Manag.* **2022**, *59*, 102844. [CrossRef]
5. Gorban, A.N.; Kégl, B.; Wunsch, D.C.; Zinovyev, A.Y. (Eds.). *Principal Manifolds for Data Visualization and Dimension Reduction*; Springer: Boston, MA, USA, 2008.
6. Yin, S.; Ding, S.X.; Xie, X.; Luo, H. A review on basic data-driven approaches for industrial process monitoring. *IEEE Trans. Ind. Electron.* **2014**, *61*, 6418–6428. [CrossRef]
7. Reddy, G.T.; Reddy, M.P.K.; Lakshmanna, K.; Kaluri, R.; Rajput, D.S.; Srivastava, G.; Baker, T. Analysis of Dimensionality Reduction Techniques on Big Data. *IEEE Access* **2020**, *8*, 54776–54788. [CrossRef]
8. Kiarashinejad, Y.; Abdollahramezani, S.; Adibi, A. Deep learning approach based on dimensionality reduction for designing electromagnetic nanostructures. *NPJ Comput. Mater.* **2020**, *6*, 1–12. [CrossRef]
9. Wang, Y.; Huang, H.; Rudin, C.; Shaposhnik, Y. Understanding how dimension reduction tools work: An empirical approach to deciphering t-SNE, UMAP, TriMAP, and PaCMAP for data visualization. *J. Mach. Learn. Res.* **2021**, *22*, 9129–9201.
10. Li, X.; Li, P.; Zhang, H.; Zhu, K.; Zhang, R. Pivotal-Aware Principal Component Analysis. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, 1–10. [CrossRef] [PubMed]
11. Ejaz, M.S.; Islam, M.R.; Sifatullah, M.; Sarker, A. Implementation of Principal Component Analysis on Masked and Non-masked Face Recognition. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–5.
12. Tran, L.; Liu, X. On Learning 3D Face Morphable Model from In-the-Wild Images. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 157–171.
13. Tenreiro Machado, J.; Lopes, A.M. Multidimensional scaling locus of memristor and fractional order elements. *J. Adv. Res.* **2020**, *25*, 147–157. [CrossRef]
14. Hägele, D.; Krake, T.; Weiskopf, D. Uncertainty-Aware Multidimensional Scaling. *IEEE Trans. Vis. Comput. Graph.* **2022**, *29*, 23–32. [CrossRef] [PubMed]
15. Sammon, J.W. A nonlinear mapping for data structure analysis. *IEEE Trans. Comput.* **1969**, *100*, 401–409. [CrossRef]
16. Liang, X.; Tang, Z.; Wu, J.; Li, Z.; Zhang, X. Robust image hashing with isomap and saliency map for copy detection. *IEEE Trans. Multimed.* **2021**, *25*, 1085–1097. [CrossRef]
17. Roweis, S.T.; Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef]
18. Zhang, H.; Ding, Y.; Meng, H.; Ma, S.; Long, Z. Component preserving and adaptive Laplacian Eigenmaps for data reconstruction and dimensionality reduction. In Proceedings of the 15th International FLINS Conference (FLINS 2022), Tianjin, China, 26–28 August 2022; pp. 642–649.
19. Tai, M.; Kudo, M.; Tanaka, A.; Imai, H.; Kimura, K. Kernelized Supervised Laplacian Eigenmap for Visualization and Classification of Multi-Label Data. *Pattern Recognit.* **2022**, *123*, 108399. [CrossRef]

20. Zhang, Y.; Li, B.; Liu, Y.; Wang, H.; Miao, C. Initialization matters: Regularizing manifold-informed initialization for neural recommendation systems. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Virtual Event, 14–18 August 2021; pp. 2263–2273.
21. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
22. Cai, T.T.; Ma, R. Theoretical foundations of t-sne for visualizing high-dimensional clustered data. *J. Mach. Learn. Res.* **2022**, *23*, 13581–13634.
23. Poličar, P.G.; Stražar, M.; Zupan, B. Embedding to reference t-SNE space addresses batch effects in single-cell classification. *Mach. Learn.* **2023**, *112*, 721–740. [CrossRef]
24. Wang, Q.; Xia, W.; Tao, Z.; Gao, Q.; Cao, X. Deep self-supervised t-SNE for multi-modal subspace clustering. In Proceedings of the 29th ACM International Conference on Multimedia, Virtual Event, 20–24 October 2021; pp. 1748–1755.
25. Bhatia, K.K.; Rao, A.; Price, A.N.; Wolz, R.; Hajnal, J.V.; Rueckert, D. Hierarchical manifold learning for regional image analysis. *IEEE Trans. Med. Imaging* **2014**, *33*, 444–461. [CrossRef] [PubMed]
26. Hinton, G.; Roweis, S. Stochastic neighbor embedding. In Proceedings of the NIPS, Vancouver, BC, Canada, 9–14 December 2002; Volume 15, pp. 833–840.
27. Guo, Y.; Guo, H.; Yu, S.X. Co-sne: Dimensionality reduction and visualization for hyperbolic data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 21–30.
28. Belkin, M.; Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.* **2003**, *15*, 1373–1396. [CrossRef]
29. Van der Maaten, L. Accelerating t-SNE using tree-based algorithms. *J. Mach. Learn. Res.* **2014**, *15*, 3221–3245.