Ad-TSNE: Visualizing High Dimensional Data Based On Adversarial Neural Networks

Huixi He [1, a], Nawata Kazumitsu [2, b]

[1] The University of Tokyo, Tokyo 1130033, Japan

[2] The University of Tokyo, Tokyo 1130033, Japan

[a] huixihe@g.ecc.u-tokyo.ac.jp, [b] nawata@tmi.t.u-tokyo.ac.jp

Abstract

Visualization tools are one of the most significant kinds of techniques in the artificial intelligence domain. Since 2008, Stochastic Neighbor Embedding and its improved versions or variants showed powerful abilities for visualizing high dimensional data. However, complexity in high-dimensional manifold always confuse the optimization algorithms when urging a curtain optimum. Furthermore, extraneous data points require further training cost. In our work, we highlight our novelty in strengthening the optimization process via an adversarial training process rather than iteratively optimizing the kernel-based loss function monotonously. Meanwhile, low-dimensional representation of new inputs can be embedded without any further training process based on our model. By testing our algorithm on famous open source datasets, it is evident that our visualization tool can learn the latent distribution of the real-world datasets

Keywords

Visualization; Dimensionality Reduction; Adversarial;·TSNE.

1. Introduction

Artificial intelligence tools promote many aspects in research domain or engineering applications and create many new viewpoints and chances. For researchers and decision makers, Visualization tools can decrease the cognition difficulty with only a glance of visualization map. In our work, we propose an adversarial training process for the popular visualizing tool: T kernel based Stochastic Neighbor Embedding [1] [2] [3] to realize an online and real-time visualizing function. TSNE can automatically search the low dimensional representation of data in high dimensions. However, any new inputs will break the balance of trained results. Compare with the algorithm which can only be responsible to the local data, our approach tends to learn the real latent distribution of dataset, and any new input which is homologous generated from the same source will earn its low dimensional representation without any further learning.

For reaching this goal, we tend to use the adversarial training model to try to generate target low dimensional coordinates from original high-dimensional space.

Generative Adversarial Nets [4] [5] was proposed by Goodfellow et. al. since 2014, and its variants or improved versions present powerful abilities in generation tasks. There are two parts in the GAN model: generator for generating images from random noise and discriminator for judging whether the image is real data or generated fake one. Based on adversarial training, the generator in GAN model finally can fully cheat the discriminator by achieving the *Nash Equilibrium* [6], and this successful generator can generate meaningful images from random noise.

In our hypothesis, we treat the T-SNE model as the “generator” in GAN model, the *Gaussian kernel* based or *Student’s T kernel* based data’s neighbor probability as the input of the “Discriminator” to make the True or False identification. Generator holds the tendency to cheat the Discriminator that low-dimensional data have the same neighbor probability as the high dimensional data. While Discriminator must identify the identity of the source of the neighbor probability. If the speculation from low dimensional coordinates can entirely cheat the discriminator, the trained coordinates will become convinced enough as the low dimensional representation of the high dimensional manifold.

2. Relative Researches

Several relative researches or reports inspired us for this work. We will introduce them briefly and point out our promotions beyond past researches.

2.1 Stochastic Neighbor Embedding

SNE was proposed by Hinton et. al. since 2002 [1]. They define the similarity between high dimensional data and low dimensional data with probability, which means that the current might choose other points as its neighbors, the closer the higher probability. So SNE is based on distance measurement, and use kernel function to transfer the distance to the neighbor probability. For the distance measurement, because the final result (which refers to the visualization map) normally will be demonstrated to human observer, so the intuitive distance measurement, Euclidean distance [7], will be considered in visualization tools. For the kernel function, the Gaussian kernel, or called RBF kernel [8] can describe the Euclidean distance properly. Based on the Kullback-Leibler divergence [9], the similarity between two probabilistic distribution could be defined. By minimizing the KL divergence between high dimensional distribution and low dimensional distribution, SNE can find out proper coordinates in the low dimensional manifold, such as two dimensions.

T kernel based SNE *TSNE* and its variants or improved versions achieved great success over past decades. Since 2003, Hinton et. al. started to publish the probabilistic-based visualization method [1]. They obtained a clear distribution of handwriting dataset, each class could be clearly recognized. Later in 2008, Matten et. al. promoted SNE via the T-Kernel [2] which is derived from Student’s T distribution. The robust feature of the heavy-tailed distribution help coordinates in low dimensions overcome the crowded problem. In 2014, Matten et. al. improved the training efficiency [3]. Tree-based K nearest neighbor searching technique can efficiently find the K-NN topological structure. Remote points hold very low probability which could be treat as zero, and only neighbors keep the probability which is defined by K-NN structure.

The core part of T-SNE mainly focus on the probabilistic similarity between high and low dimensional data’s distribution of neighbor relationship. In their work, data in high dimensions will use Gaussian kernel to define the probability how possible the current will choose another point as its neighbor:

$$p_{i,j} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma^2\right)}{\sum_{k\neq l}\exp(-\|y_k - y_l\|^2)}. \tag{1}$$

Due to The *Crowding Problem* [1], the famous robust distribution Student’s T distribution can derive an important kernel called T-kernel to overcome this problem:

$$q_{i,j} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}}. \tag{2}$$

And *Kullback-Leibler divergence* [9] could be a nice choice to measure the loss between high and low dimensional neighbor probabilistic. Also take the symmetric version into consideration to concur the nonequivalence in KL divergence, the final gradient could be obtained as follow:

$$\frac{\delta C}{\delta_{y_i}} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}. \tag{3}$$

Gradient descent based algorithms (such as Adam [10], RMSprop [11], SGD [12], etc.) can approximate optimal results in low dimensional space.

2.2 Other visualization tools

Multidimensional Scaling (MDS) Using the distance between pairs of points in high-dimensional, *MDS* (Cox and Cox) [25] managed to construct a suitable low-dimensional space to contain the pairwise distances similar between these two spaces. It begins with a matrix $D = [d_{ij}]$, where d_{ij} stands for the distance between two points i and j in high-dimensional, and the distance here can be given by Euclidean distance,but not necessary. [25] The distance of pairwise points in low-dimensionality mapped form high-dimensionality can be described as a matrix $X = [x_{ij}]$ [29]. Usually, a solution can be found by minimizer a cost function such as

$$\min \sum_{ij} ||x_{ij} - d_{ij}||^2 . \quad (4)$$

Principal component analysis (PCA) In some cases, some of the given features of samples are redundant. *PCA* (Jolliffe) is such a method, which not only reduce the dimensional, but more importantly, remove the noise through dimension reduction, and recognize the patterns in data as well. Generally speaking, the main idea of *PCA* is to transfer the original data into a set of linearly uncorrelated values, remain the information with evident and essential features by mapping n-dimensional features to a new k-dimensional space, where $k < n$. The Orthogonal projection of data in low-dimensional linear space called the principal subspace can be defined as standards of the quality of features by maximizing the variance in the projection space. Equivalently, it can also be interpreted as minimizes the squared reconstruction error $\Sigma_{\mathrm{n}} \|\mathbf{t}_{\mathrm{n}} - \widehat{\mathbf{t}_{\mathrm{n}}}\|^2$, where $\mathbf{t}_n$ is a set of d-dimensional data vector. [26] [28]

ISOMAP Isomap (Tenenbaum, Silva, Langford), based on *PCA* and *MDS*, is a widely accepted and a traditional method used to dimensionality reduction. This method captures the main features of *MDS* and breaks its limits, expand dimensionality reduction to non-linear manifolds [27]. To deal with this question, *Isomap* focus on estimating the geodesic distance $d_G(i,j)$ between pairs of points i and j. Geodesic distance can be approximated by the Euclidean distance $d_X(j,j)$, because the Euclidean distance can replace the geodesic distance for neighboring points. And for the distance between faraway points, shortest path distance here used to estimate it. Using geodesic distance, it can construct a matrix $D_G = d_G(i,j)$, then apply *MDS* to this matrix to form the embedding coordinate representation of data in d-dimensional space $\mathbb{Y}$. For descend dimension, it is necessary to minimize the cost function [28] [29].

Major difference Based on past dimensionality reduction algorithms above, we are easy to visualize high-dimensional data. Each algorithm owns its unique characteristics qualified for different kinds of datasets. However, none of these algorithms focuses on some particular cases such as immediate visualizing requirement. As we know, big data brings a massive amount of information. On the other hand, big data also requires a rigorous computational cost. Data renews everywhere and anytime. Past visualization tools can visualize high-dimensional data based on different theorems, but all of them have to compute again when new inputs are coming. In our model, the successfully trained dimension reducer will learn the regularity among data dimensions, and no more training process is needed when new inputs are coming, only if they are *i.i.d.* when compared with the training dataset.

2.3 Generative Adversarial Nets

Adversarial training can promote the quality of generated results, such as the vague problem in Variational Auto-encoder, etc. In our work, we want to use the framework of the generative adversarial networks to improve the precise of visualization results.

Since 2014, Goodfellow developed a new viewpoint for generative models, known as Generative Adversarial Nets(GAN), now has been widely applied in image generation. The advantage of GAN is that is can generate samples without approximate inference and Markov chains so that some complicated sum calculations can be avoided, which make it quite convenient in the case of complex data. The framework of GAN mainly contains at least two parts: a generative model G and a discriminative model D. By giving random noise to G, the new sample $G(z)$ is generated. Then, the generated $G(z)$ or original sample will be treated as inputs of D. From these inputs, D tries to identify the real sample and the generated sample as correctly as possible. When D cannot distinguish the difference between the true data and the fake data, *G* will achieve its Nash equilibrium against *D*. [6] That is, the primary purpose of *G* is to generate samples that consistent with original data without being rejected by *D*, while the goal of *D* is to determine whether a sample is generated by *G* or not, in other words, they must compete with each other. This process can be expressed in the following function:

$$\min_{G,D}\max \mathrm{V}\,(D,G) = \mathbb{E}_{\mathrm{x}\sim \mathrm{p}_{\mathrm{data}}(\mathrm{x})}[\log D(\mathrm{x})] + \mathbb{E}_{\mathrm{z}\sim \mathrm{p}_{\mathrm{z}}(\mathrm{z})}\left[\log\left(1 - D\big(G(z)\big)\right)\right] \quad (5)$$

Where *D*(*x*) refers to the probability that *x* came from the data as Goodfellow et al. explained in their paper. [4] [24] By training *G* and *D* simultaneously, they will influence each other and reach equilibrium. This process can be described as a two-player minimax game, both *G* and *D* are managed to maximize their income so that the value function can be converged.

3. Algorithms

Dimensionality Reducer The first part is designed similar to, but not exactly same as, the generator in GAN models. Consider the original generator in GAN models: they can generate meaningful images from random noise. So, generator could be treated as a converter which can convert random noise to the target distribution of real data manifolds. However, in our research, our novel method will treat the generator as the converter from high dimensions to the low dimensional manifold. Especially, based on the Whitney Embedding Theorem, high-dimensions can always be embedded into low-dimensional space smoothly. So it is considered to use neural networks to approximate that mapping function. Particularly, the input of the generator will be set as data coordinates in high dimensional space. And the output will be set as low-dimensional coordinates.

Parameterization We need a standard to evaluate the quality of coordinates generated from the dimensionality reducer. Based on the mature dimensionality reduction tool T-SNE, the stochastic neighbor embedding algorithm performs excellently on many datasets. So, we will first transfer algebraic coordinates in *Reproducing Kernel Hilbert Space* (RKHS) [13] measurements in *Probability Space*. As explained above, Student's T kernel shows the power of low-dimensional coordinates modeling. So, we decided to use the Gaussian Kernel to convert high dimensional coordinates and T kernel for low-dimensional coordinates. Firstly, coordinates with shape like $\mathbb{C}_{\mathrm{high}}^{(*,\mathrm{n})}$ will be transfer to the diagonal matrix

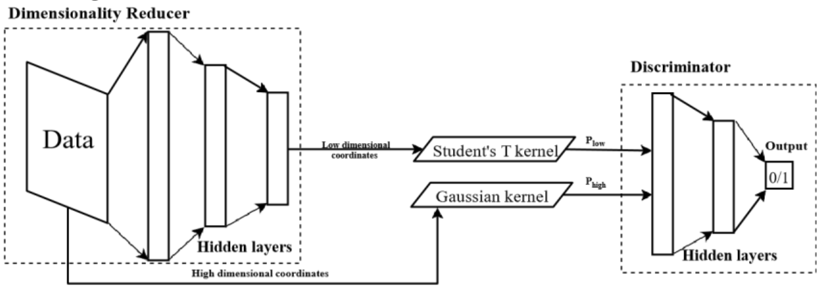


Fig.1. This is the architecture of our model. As shown in this graph, two major part construct our model. For the Dimensionality Reducer, the input data will be reshaped to $\mathbb{C}^{(*,n)}$ at first, then pass through feed-forward layers and down to l features. Where l refers to the number of target dimensions in the low dimensional space and l << n (convenient to visualization, the default value of l is two). The output of Dimensionality Reducer part is the low dimensional coordinate, which is the answer of our target. For the Discriminator part, it receives probabilistic input transferred from different source of coordinates via corresponding kernels. The discriminator will rearrange information from inputs and will judge the source of input and return Boolean answers.

like$\mathbb{P}_{\mathrm{high}}^{(*,*)}$, where the uncertain "*" will be decided by the batch size in the training process; in validation, it could be any size depends on the size of test data. The value *n* refers to the number of features in the original space. This diagonal matrix reveals the inner relationship between data coordinates, showing higher probability for smaller pair-wise Euclidean Distance based on second ordered norm between data points. The physics meaning of $P_{<i,j>} \in \mathbb{P}_{\text{high}}^{(*,*)}$ points out how possible point *i* will choose *j* as its neighbor based on the value of $\|C_i - C_j\|_2$.

$$\mathrm{f}_{\text{Gaussian}}: \mathbb{C}_{\text{low}}^{(*,\mathrm{n})} \mapsto \mathbb{P}_{\text{low}}^{(*,*)} . \tag{6}$$

Secondly, similar to the case in high-dimensional space, just convert the coordinates generated from Algorithm 1 to probabilistic matrix via the *Student's T kernel* function, the input for next step could be obtained.

$$\mathrm{f}_{\text{Student's-T}}: \mathbb{C}_{\text{low}}^{(*,\mathrm{n})} \mapsto \mathbb{P}_{\text{low}}^{(*,*)} . \tag{7}$$

However, data in large scale limit the computational cost. If we tend to find the whole pair-wise similarity matrix, the complexity will become to $\mathcal{O}(\mathrm{N}^2)$. Similar to the tree-based TSNE, we also use the K-NN algorithm [14] to seek the neighbor points for each point, and use zero to instead further computation for remote points. However, we are not satisfied with current idea. Mini-batch technique [15] is one of the most significant techniques among deep learning algorithms. It can grantee the training efficiency and convergence. So, we treat each point and its k-neighbor points as one mini-batch to train the model. Here the shape of the input of *Algorithm 1* will becomes to $\mathbb{C}_{\text{high}}^{(\mathrm{k,n})}$ for every mini-batch, where *k* refers to k- neighbors and *n* is the dimensions of features. Synchronously, the shape of

Algorithm 1 Dimensionality Reducer: $\mathcal{R}$

Require:

Aligned Coordinates in original space: $\mathbb{C}_{\text{high}}^{(*,\mathrm{n})}$;

Parameters initialization.

Ensure:

Corresponding coordinates in target space: $\mathbb{C}_{\text{low}}^{(*,\mathrm{l})}$.

```
1: while Alg.1 has not converged do
2:     for $mini-batch \in batches$ do
3:         feed-forward propagation;
4:         back propagate gradients;
5:         renew parameters;
6:     end for
7: end while
```

The output of *Algorithm 1* will becomes to $\mathbb{C}_{\text{low}}^{(\mathrm{k,l})}$. And in the following step, the parameterization for the output will also changes:

$$\mathrm{f}_{\text{Gaussian}}^{*}: \mathbb{C}_{\text{high}}^{(\mathrm{k,n})} \mapsto \mathbb{P}_{\text{high}}^{(1,\mathrm{k})} , \tag{8}$$

$$\mathrm{f}_{\text{Student's-T}}^{*}: \mathbb{C}_{\text{low}}^{(\mathrm{k,n})} \mapsto \mathbb{P}_{\text{low}}^{(1,\mathrm{k})} . . \tag{9}$$

Pay attention that we only take out one column from the diagonal matrix $\mathbb{P}^{(\mathrm{k,k})}$. Because we set each point and its neighbor points together as one mini-batch. So, this mini-batch reflect information about this current point. But for the information about its neighbor points, though they will also hold their own specific neighbors (neighbors of neighbors), it will not be considered in the current iteration. As a result, we will only consider the information about the current point, which means we will only compute the $\mathbb{P}^{(1,\mathrm{k})}$ but not the whole $\mathbb{P}^{(\mathrm{k,k})}$. Based on parameterization above, we can use transfer the original algebraical coordinates to the probabilistic based measurement space for further analysis.

Discriminator Obviously, it is hard to keep information from high dimensions to low-dimensional space. So, we need another DNN to help *Algorithm 1*. Via the parameterization, coordinates were transferred to probabilistic inputs of *Algorithm 2*, and *Algorithm 2* will justify the source of input: return true for high-dimensional coordinates and false for low-dimensional ones. The input contains only one high-dimensional sample, so data-wised normalization sills are no more worked. As a result, we tend to use “SeLU” [16] activating function to normalize each layer instead of popular Batch Normalization technique. And for the Dimensionality reducer, we used the ReLU [18] activation function.

Adversarial training Now we have architectures of dimensionality reducer and discriminator, also, the parameterizing skill for space transferring. Next step is for optimization. Loss functions contain two parts, include *Reducer* loss and *Discriminator* loss separately. First, we introduce the Discriminator loss. Similar to the generative adversarial nets, different sources will be labeled with

Algorithm2 Discriminator$\mathcal{D}$

```
Require:
Neighboring probability: $\mathbb{P}^{(1,n)}$;
parameters initialization.
Ensure:
Identify True or False.
1: while Alg.2 has not converged do
2:     for $mini - batch \in batches$ do
3:         feed-forward propagation;
4:         back propagate gradients;
5:       renew parameters;
6:     end for
7: end while
```

different Boolean values. For Gaussian kernel based input, the output will be set as **1**; for Student’s T kernel based input, the output will be set as **0**. So, the cross-entropy loss [19] of discriminator could be written as follow:

$$\mathcal{L}_D = -\frac{1}{n}\sum_{i=1}^{n}\left(log\left(1-D_{low}^{(i)}+\epsilon\right)+log\left(D_{high}^{(i)}+\epsilon\right)\right) \tag{10}$$

Where $\mathbf{D}_*$ are outputs activated via the Sigmoid function. ϵ is a very small number in order to prevent the division by zero.

Next is the Reducer loss. One of the major difference between our dimensionality reducer and the conventional generator in GAN model is that we also applies the True/False concept into the generative part. However, this is not for identifying the data source. In chapter III-B, we talked about using neighbor points instead of using total points to reduce the computational cost. For past research, they only keep neighbor points and treat remote points as zero probability. However, based on our experiment, GAN model cannot obtain a good result when only train the model with neighbor points and treat remote points as zero probability. So, we artificially made some “remote points” to help the training process achieve convergence. One remote point equals to the opposite number of the original number by adding some small noise.

$$\mathbb{C}_r = -\mathbb{C}_o + Noise, \tag{11}$$

$$\text{Noise} \sim \alpha \times \mathcal{N}(0,1)\,. \tag{12}$$

α refers to the scaling index of noise, we set it as 10^{-2}. And The loss function for Generator is defined based on the result from trained Discriminator:

$$\mathcal{L}_R = -\frac{1}{n}\sum_{i=1}^{n}\left(\log\left(1-D_{R(C_r)}^{(i)}+\epsilon\right)+\log\left(D_{R(C_o)}^{(i)}+\epsilon\right)\right) \tag{13}$$

We first train the Discriminator, then use this new hand Discriminator to help to train the Dimensionality Reducer. Then we use the newly trained *R* to generate more deceptive samples to mislead Discriminator. Then train Discriminator again to recover its ability of judgment, and keep these adversarial training loops until convergence. Finally, we can use the successfully trained Dimensionality Reducer *R* to reduce the dimensions of samples in test dataset.

Many kinds of loss function or regularization methods were published in order to overcome the training problem in GAN model, such as BEGAN [20], WGANGP [21], etc. They are quite fancy in different situations. We test several kinds of famous models while results do not have too much variations among these methods. So, experiments were implemented via the original GAN.

4. Experiment

4.1 Datasets

We use several famous open sources to test the feasibility of our model because they can be easily downloaded and already been analyzed by many researchers.

MNIST dataset MNIST hand writing dataset [22] is one of the most famous open source dataset which contains 60k samples for training and 10k for testing. In our work, we train our model on the training set and visualize the testing set.

CIFAR-10 dataset CIFAR dataset [23] is the image dataset with clear labels. It has several versions, each contains different numbers of labels. We use the 10-label version in CIFAR dataset. In our work, we train our model with 60,000 images and visualize other 10k images.

4.2 Experiment design

First, we collect and pre-process the original data. Images are matrix of pixels, so we normalize values of pixels between zero and one to create a benign initial state for the model training.

Next, we iteratively train our model. It is hard to decide some hyper-parameters in artificial neural network architectures. We set hyper-parameters as default at first, and adjust them gradually.

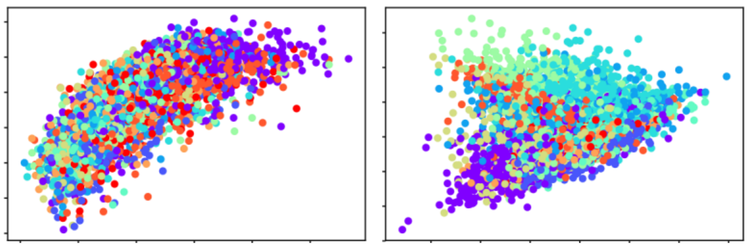
Finally, we compute the 2-dimensional representation of the test dataset, plot them to the 2-D map and compute the error rate.

4.3 Results and evaluation

We demonstrate visualization maps from our experiments in Fig. 2. We also evaluate the quality of the generated points by comparing their pair-wise distances to the pair-wise distances in the original space:

$$Error\ matrix=\begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n,1} & d_{n,2} & \cdots & d_{n,n} \end{bmatrix} \tag{14}$$

C^* refer to the low or high dimensional coordinates of one point. We compute the second order norm value for all data points' pairs as their Euclidean Distance. And $d_{i,j}$ refers to the difference of distance between high and low dimensional space. In this work, we set the normalization parameter as the maximum number among the whole distance matrix for high-dimensional coordinates and low-dimensional coordinates separately. As a result, $error \in (0,1]$.



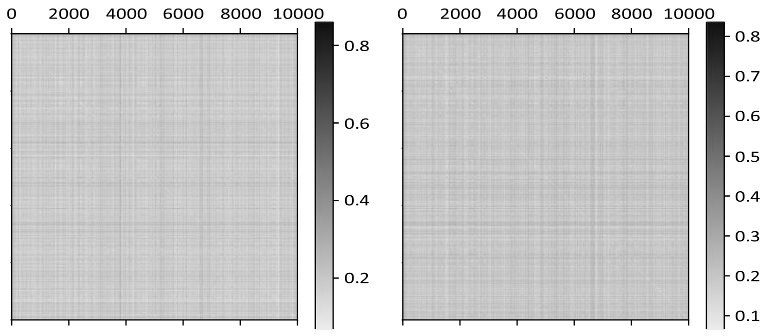
(a) CIFAR-10 (b) MNIST

Fig.2. Demonstration of visualization maps on test datasets. The first figure is based on the CIFAR-10 dataset and the second one shows the MNIST dataset.

Where,

$$norm^{(i,j)} = \frac{\left\|\mathbb{C}^i - \mathbb{C}^j\right\|_2}{normalization\ parameter}, \tag{15}$$

$$d_{i,j} = \left|norm_{high}^{(i,j)} - norm_{low}^{(i,j)}\right|. \tag{16}$$



(a) CIFAR-10 (b) MNIST

Fig.3. Error matrix of the pair-wise distance for pairs between each point. The first figure is based on the case in the 10,000 samples' CIFAR-10 testing dataset and the second one shows the case of the 10,000 samples' MNIST testing dataset. The mean value of the error matrix is 18.84% for CIFAR-10 testing dataset and 19.51% for MNIST testing dataset.

Both two error matrixes obtained a mean value below 20%, which means the lowdimensional representation is trustful more 80%.

As for other visualization tools, though they can display beautiful maps, however, they cannot realize similar function as our work: visualizing new inputs without further training. So, our work can help some online or real time application scenarios especially.

5. Conclusion and Discussion

In this paper, we purposed a new perspective for optimization process, makes it possible to embed new data in low-dimensional without further training. To achieve this goal, we introduced the adversarial training model to a popular visualizing tool named TSNE. The reason why this model was chosen is that it is capable of learning the exact relationship between training dataset, which can directly generate target data in low-dimensional form original data.

One of the innovative points in our work is that we regard the neighbor probability of data as the goal of the discriminator *D*, which can improve the performance of TSNE by using the framework of GAN. We use the data from original space instead of the random noise as the input of generator in GAN model, so it is easier for our model to achieve convergence than GAN models. Our experiments show that this method can learn the latent distribution of original data manifold. Furthermore, new incomes can also observe its low-dimensional representation directly. As we mentioned above, online and real-time scenarios are advantages. Therefore, some relevant applications could be expected. For instance, our method can help some website to visualize the *real-time flow measurement* (RTFM). Likewise, the visualization for the real-time traffic condition on the mobile terminal is also necessary for daily life.

This work may provide a new direction for other researches and own theoretical and application-based meaning as well. Some points are remained to be explored in the future, and we will consider the feasibility of applying other kernel functions to our model.

References

[1] Hinton, Geoffrey E., and Sam T. Roweis. ”Stochastic neighbor embedding.” Advances in neural information processing systems. 2003.

[2] Maaten, Laurens van der, and Geoffrey Hinton. ”Visualizing data using t-SNE.” Journal of machine learning research 9.Nov (2008): 2579-2605.

[3] Van Der Maaten, Laurens. ”Accelerating t-SNE using tree-based algorithms.” Journal of machine learning research 15.1 (2014): 3221-3245.
[4] Goodfellow, Ian, et al. ”Generative adversarial nets.” Advances in neural information processing systems. 2014.
[5] Goodfellow, Ian. ”NIPS 2016 tutorial: Generative adversarial networks.” arXiv preprint arXiv:1701.00160 (2016).
[6] Nash, John. ”Non-cooperative games.” Annals of mathematics (1951): 286-295.
[7] Danielsson, Per-Erik. ”Euclidean distance mapping.” Computer Graphics and image processing 14.3 (1980): 227-248.
[8] Vert, Jean-Philippe, Koji Tsuda, and Bernhard Schlkopf. ”A primer on kernel methods.” Kernel methods in computational biology 47 (2004): 35-70.
[9] Kullback, Solomon, and Richard A. Leibler. ”On information and sufficiency.” The annals of mathematical statistics 22.1 (1951): 79-86.
[10] Kingma, Diederik P., and Jimmy Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).
[11] Tieleman, Tijmen, and Geoffrey Hinton. ”Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.” COURSERA: Neural networks for machine learning 4.2 (2012): 26-31.
[12] Amari, Shun-ichi. ”Backpropagation and stochastic gradient descent method.” Neurocomputing 5.4-5 (1993): 185-196.
[13] Berlinet, Alain, and Christine Thomas-Agnan. Reproducing kernel Hilbert spaces in probability and statistics. Springer Science and Business Media, 2011.
[14] Larose, Daniel T. ”knearest neighbor algorithm.” Discovering knowledge in data: An introduction to data mining (2005): 90-106.
[15] Li, Mu, et al. ”Efficient mini-batch training for stochastic optimization.” Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.
[16] Klambauer, Gnter, et al. ”Self-normalizing neural networks.” Advances in Neural Information Processing Systems. 2017.
[17] Ioffe, Sergey, and Christian Szegedy. ”Batch normalization: Accelerating deep network training by reducing internal covariate shift.” arXiv preprint arXiv:1502.03167 (2015).
[18] Nair, Vinod, and Geoffrey E. Hinton. ”Rectified linear units improve restricted boltzmann machines.” Proceedings of the 27th international conference on machine learning (ICML-10). 2010.
[19] Shore, John, and Rodney Johnson. ”Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy.” IEEE Transactions on information theory 26.1 (1980): 26-37.
[20] Berthelot, David, Tom Schumm, and Luke Metz. ”Began: Boundary equilibrium generative adversarial networks.” arXiv preprint arXiv:1703.10717 (2017).
[21] Gulrajani, Ishaan, et al. ”Improved training of wasserstein gans.” Advances in Neural Information Processing Systems. 2017.
[22] LeCun, Yann, Corinna Cortes, and C. J. Burges. ”MNIST handwritten digit database.” AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist 2 (2010).
[23] Krizhevsky, Alex, Vinod Nair, and Geoffrey Hinton. ”The CIFAR-10 dataset.” online: http://www. cs. toronto. edu/kriz/cifar. html (2014).
[24] Goodfellow, Ian, et al. Deep learning. Vol. 1. Cambridge: MIT press, 2016.
[25] Cox, Trevor F., and Michael AA Cox. Multidimensional scaling. CRC press, 2000.
[26] Wold, Svante, Kim Esbensen, and Paul Geladi. ”Principal component analysis.” Chemometrics and intelligent laboratory systems 2.1-3 (1987): 37-52.
[27] Balasubramanian, Mukund, and Eric L. Schwartz. ”The isomap algorithm and topological stability.” Science 295.5552 (2002): 7-7.
[28] Roweis, Sam T., and Lawrence K. Saul. ”Nonlinear dimensionality reduction by locally linear embedding.” science 290.5500 (2000): 2323-2326.

[29] Bengio, Yoshua, et al. ”Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering.” Advances in neural information processing systems. 2004.