# Improved Deep Embedded Clustering with Local Structure Preservation

**Xifeng Guo, Long Gao, Xinwang Liu, Jianping Yin**
College of Computer, National University of Defense Technology, Changsha, China
guoxifeng1990@163.com, 1017730430@qq.com, 1022xinwang.liu@gmail.com, jpyin@nudt.edu.cn

## Abstract

Deep clustering learns deep feature representations that favor clustering task using neural networks. Some pioneering work proposes to simultaneously learn embedded features and perform clustering by explicitly defining a clustering oriented loss. Though promising performance has been demonstrated in various applications, we observe that a vital ingredient has been overlooked by these work that the defined clustering loss may corrupt feature space, which leads to non-representative meaningless features and this in turn hurts clustering performance. To address this issue, in this paper, we propose the Improved Deep Embedded Clustering (IDEC) algorithm to take care of data structure preservation. Specifically, we manipulate feature space to scatter data points using a clustering loss as guidance. To constrain the manipulation and maintain the local structure of data generating distribution, an under-complete autoencoder is applied. By integrating the clustering loss and autoencoder's reconstruction loss, IDEC can jointly optimize cluster labels assignment and learn features that are suitable for clustering with local structure preservation. The resultant optimization problem can be effectively solved by mini-batch stochastic gradient descent and backpropagation. Experiments on image and text datasets empirically validate the importance of local structure preservation and the effectiveness of our algorithm.

## 1 Introduction

Unsupervised clustering is a vital research topic in data science and machine learning. Traditional clustering algorithms like k-means [MacQueen, 1967], gaussian mixture model [Bishop, 2006] and spectral clustering [Von Luxburg, 2007] group data on handcraft features according to intrinsic characteristics or similarity. However, when the dimension of input feature space (data space) is high, the clustering becomes ineffective due to unreliable similarity metrics. Transforming data from high dimensional feature space to lower dimensional space in which to perform clustering is an intuitive solution and has been widely studied [Von Luxburg, 2007]. This can be done by applying dimension reduction techniques like Principle Component Analysis (PCA), but the representation ability of these shallow models is limited. Thanks to the development of deep learning, such feature transformation can be achieved by using Deep Neural Networks (DNN). We refer to this kind of clustering algorithms as *deep clustering*.

Deep clustering is most recently proposed and leaves a lot of problems unsolved. For example, what types of neural networks are proper? How to provide guidance information i.e. to define clustering oriented loss function? Which properties of data should be preserved during transformation? The primitive work in deep clustering focuses on learning features that preserve some properties of data by adding prior knowledge to the subjective [Tian *et al.*, 2014][Peng *et al.*, 2016]. They are two-stage algorithms: feature transformation and then clustering. Latter, algorithms that jointly accomplish feature transformation and clustering come into being [Yang *et al.*, 2016][Xie *et al.*, 2016]. The Deep Embedded Clustering (DEC) [Xie *et al.*, 2016] algorithm defines an effective objective in a self-learning manner. The defined clustering loss is used to update parameters of transforming network and cluster centers simultaneously. While the cluster assignment is implicitly integrated to soft labels. However, the local structure preservation can not be guaranteed by the clustering loss. Thus the feature transformation may be misguided, leading to corruption of embedded space and learning unreasonable representations.

To deal with this problem, in this paper, we assume that both clustering oriented loss guidance and local structure preservation mechanism are essential for deep clustering. Inspired by [Peng *et al.*, 2016], we use under-complete autoencoder to learn embedded features and to preserve local structure of data generating distribution. We propose to incorporate autoencoder into DEC framework. In this way, the proposed framework can jointly perform clustering and learn representative features with local structure preservation. Our algorithm can be viewed as a regularized autoencoder [Goodfellow *et al.*, 2016] with regularizer being clustering loss [Xie *et al.*, 2016]. It can also be seen as an improved version of DEC. For convenience of description, we refer to our algorithm as Improved Deep Embedded Clustering (IDEC). The optimization of IDEC is straightforward. Only the forward and backward processes between embedded code and clustering loss need to take care of. The rest can directly perform mini-batch stochastic gradient descent and backpropagation. At last, some experiments are carefully designed and conducted. The results validate our assumption and the effectiveness of our IDEC.

The contributions of this work are summarized as below:

- We propose a deep clustering algorithm that can jointly perform clustering and learn representative features with local structure preservation.
- We empirically prove the importance of local structure preservation in deep clustering.
- The proposed IDEC outperforms the newest opponent in a large margin.

Our IDEC implementation based on Keras [Chollet, 2015] is publicly available at `https://github.com/XifengGuo/IDEC`.

# 2 Related Work

## 2.1 Deep Clustering

Existing deep clustering algorithms broadly fall into two categories: (i) two-stage work that applies clustering after having learned a representation, and (ii) approaches that jointly optimize the feature learning and clustering.

The former category of algorithms directly take advantage of existing unsupervised deep learning frameworks and techniques. For example, [Tian *et al.*, 2014] uses autoencoder to learn low dimensional features of original graph, and then runs k-means algorithm to get clustering results. [Chen, 2015] layer-wisely trains a Deep Belief Network (DBN) and then applies non-parametric maximum-margin clustering to learned intermediate representation. [Peng *et al.*, 2016] uses autoencoder with sparsity prior to learn representations in nonlinear latent space that are adaptive to local and global subspace structure simultaneously, and then traditional clustering algorithms are employed to get label assignment.

The other category of algorithms try to explicitly define a clustering error, simulating classification error in supervised deep learning. [Yang *et al.*, 2016] proposes a recurrent framework in deep representations and image clusters, which integrates two processes into a single model with a unified weighted triplet loss and optimizes it end-to-end. DEC [Xie *et al.*, 2016] learns a mapping from the observed space to a low-dimensional latent space with deep neural networks, which can obtain feature representations and cluster assignments simultaneously.

The proposed algorithm intrinsically is a modified version of DEC with incorporating an under-complete autoencoder to preserve local structure. It excels [Yang *et al.*, 2016] by simplicity without recurrent and outperforms DEC in terms of clustering accuracy and feature's representativeness. Since IDEC mainly depends on autoencoder and DEC, we will introduce them in more detail in the following sections.

## 2.2 Autoencoder

An autoencoder is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer $z$ that describes a code used to represent the input. The network consists of two parts: an encoder function $z = f_W(x)$ and a decoder $x' = g_{W'}(z)$ that produces a reconstruction. The reconstruction error $\|x' - x\|$ is used to train the network and the goal is to let latent code $z$ represent input data well. To avoid identity mapping $x = g_{W'}(f_W(x))$, we review two types of autoencoders that are utilized in our algorithm.

**Under-complete autoencoder**. The simplest way of avoiding identity mapping is to control the dimension of latent code $z$ lower than input data $x$. Learning such under-complete representations force the autoencoder to capture the most salient features of the data. When activations of encoder and decoder are linear and reconstruction error is mean square error $\|x' - x\|_2^2$, an under-complete autoencoder learns to span the same subspace as PCA. Autoencoders with nonlinear activation can thus learn a more powerful nonlinear generalization of PCA. So the latent code $z$ is compact and preserves salient structure of data.

**Denoising autoencoder**. Instead of reconstructing $x$ given $x$, denoising autoencoder minimizes the following objective:

$$L = \|x - g_{W'}(f_W(\tilde{x}))\|_2^2 \tag{1}$$

where $\tilde{x}$ is a copy of $x$ that is corrupted by some form of noise. Therefore, denoising autoencoder has to recover $x$ from this corruption rather than simply copying their input. In this way, denoising autoencoder can force encoder $f_W$ and decoder $g_{W'}$ to implicitly capture the structure of data generating distribution even when latent code is not low-dimensional and networks have large capacities, as shown in [Bengio *et al.*, 2013].

In our algorithm, the denoising autoencoder is used for pretraining and under-complete autoencoder is added to DEC framework after initialization.

## 2.3 Deep Embedded Clustering

Deep Embedded Clustering (DEC) [Xie *et al.*, 2016] starts with pretraining an autoencoder and then removes the decoder. The remaining encoder is finetuned by optimizing the following objective:

$$L = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{2}$$

where $q_{ij}$ is the similarity between embedded point $z_i$ and cluster center $\mu_j$ measured by Student's $t$-distribution [Maaten and Hinton, 2008]:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}} \tag{3}$$

And $p_{ij}$ in (2) is the target distribution defined as

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j \left( q_{ij}^2 / \sum_i q_{ij} \right)} \tag{4}$$

As we can see, the target distribution $P$ is defined by $Q$, so minimizing $L$ is a form of self-training [Nigam and Ghani, 2000].

Let $f_W$ be the encoder mapping, i.e. $z_i = f_W(x_i)$ where $x_i$ is input example from dataset $X$. After pretraining, all embedded points $\{z_i\}$ can be extracted using $f_W$. Then employ k-means on $\{z_i\}$ to get initial cluster centers $\{\mu_j\}$. Afterwards, $L$ can be computed according to (2), (3) and (4). And the predicted label of sample $x_i$ is $\arg\max_j q_{ij}$.

During backpropagation, $\partial L/\partial z_i$ and $\partial L/\partial \mu_j$ can be easily computed. Then $\partial L/\partial z_i$ is passed down to update $f_W$ and $\partial L/\partial \mu_j$ is used to update cluster center $\mu_j$:

$$\mu_j = \mu_j - \lambda \frac{\partial L}{\partial \mu_j} \tag{5}$$

The biggest contribution of DEC is the clustering loss (or target distribution $P$, to be specific). It works by using high confidential samples as supervision and then making samples in each cluster distribute more densely. However, there is no guarantee of pulling samples near margins towards the correct cluster. We deal with this problem by explicitly preserving the local structure of data. Under this condition, the supervision information of high confidential samples can help the marginal samples walk to the correct cluster.

## 3 Improved Deep Embedded Clustering

Consider a dataset $X$ with $n$ samples and each sample $x_i \in \mathbb{R}^d$ where $d$ is the dimension. The number of clusters $K$ is a priori knowledge and the $j$th cluster center is represented by $\mu_j \in \mathbb{R}^d$. Let the value of $s_i \in \{1, 2, \ldots, K\}$ represent the cluster index assigned to sample $x_i$. Define nonlinear mapping $f_W : x_i \to z_i$ and $g_{W'} : z_i \to x'_i$ where $z_i$ is the embedded point of $x_i$ in the low dimension feature space and $x'_i$ is the reconstructed sample for $x_i$.

We aim to find a good $f_W$ which makes embedded points $\{z_i\}_{i=1}^n$ more suitable for clustering task. To this end, two components are essential: the autoencoder and clustering loss. The autoencoder is used to learn representations in unsupervised manner and the learned features can preserve intrinsic local structure in data. The clustering loss, borrowed from [Xie *et al.*, 2016], is responsible for manipulating embedded space in order to scatter embedded points. The whole network structure is illustrated in Fig. 1. And the objective is defined as

$$L = L_r + \gamma L_c \tag{6}$$

where $L_r$ and $L_c$ are reconstruction loss and clustering loss respectively, and $\gamma > 0$ is a coefficient that controls the degree of distorting embedded space. When $\gamma = 1$ and $L_r \equiv 0$, (6) reduces to the objective of DEC [Xie *et al.*, 2016].

### 3.1 Clustering loss and Initialization

The clustering loss is proposed by [Xie *et al.*, 2016]. It is defined as KL divergence between distributions $P$ and $Q$, where $Q$ is the distribution of soft labels measured by $t$-student distribution and $P$ is the target distribution derived from $Q$. That is to say, the clustering loss is defined as

$$L_c = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{7}$$

where $KL$ is KullbackLeibler divergence that measures the non-symmetric difference between two probability distributions, $P$ and $Q$ are defined by (4) and (3). Details can be found in Section 2.3 and [Xie *et al.*, 2016].

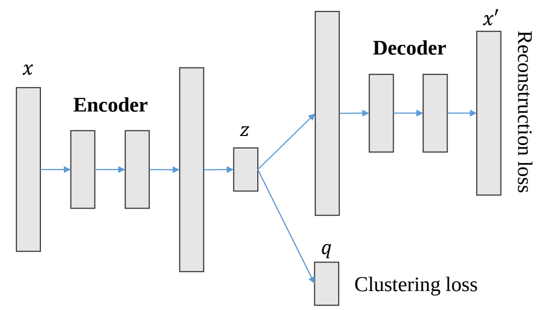


Figure 1: The network structure of IDEC. The encoder and decoder are composed of fully connected layers. Clustering loss is used to scatter the embedded points $z$ and the reconstruction loss makes sure that the embedded space preserves local structure of data generating distribution.

According to (3), soft labels depend on cluster centers and embedded points. If the network is randomly initialized, embedded points do not correspond to the structure of input data. Thus the soft labels are meaningless and the clustering loss (2) can not provide useful information to tune the network parameters. So before using clustering loss, we need to pretrain the network to get reasonable embedded representations. Any unsupervised neural networks are viable choices, such as Deep Belief Nets (DBN) [Hinton *et al.*, 2006], Stacked Autoencoder (SAE) [Vincent *et al.*, 2010] and Generative Adversarial Nets (GAN) [Goodfellow *et al.*, 2014][Chen *et al.*, 2016]. Since the pretraining is not our focus in this paper, we simply follow the same setting in [Xie *et al.*, 2016], i.e. finetuning a stacked denoising autoencoder after layer wise pretraining.

After pretraining, embedded points are valid feature representations for input samples. Then cluster centers $\{\mu_j\}_{j=1}^K$ can be initialized by employing k-means on $\{z_i = f_W(x_i)\}_{i=1}^n$

### 3.2 Local structure preservation

The embedded points obtained in Section 3.1 are not necessarily suitable for clustering task. To this end, DEC [Xie *et al.*, 2016] abandons the decoder and finetunes encoder using clustering loss $L_c$. However, we suppose that this kind of finetuning could distort the embedded space, weaken the representativeness of embedded features and thereby hurt clustering performance. Therefore, we propose to keep the decoder untouched and directly attach the clustering loss to embedded space.

To ensure effectiveness of clustering loss, the stacked denoising autoencoder used in pretraining is not appropriate any more. Instead, we utilize the standard autoencoder with low embedded dimension constraint. This is called *Undercomplete Autoencoder* in Section 14.1 of [Goodfellow *et al.*, 2016]. The reconstruction loss is measured by Mean Squared Error (MSE):

$$L_r = \sum_{i=1}^n \|x_i - g_{W'}(z_i)\|_2^2 \tag{8}$$

where $z_i = f_W(x_i)$ and $f_W$ and $g_{W'}$ are encoder and decoder mappings respectively. As shown in [Peng *et al.*, 2016] and [Goodfellow *et al.*, 2016], autoencoders can preserve local structure of data generating distribution. Under this condition, manipulating embedded space slightly using clustering loss will not cause corruption. So the coefficient $\gamma$ is set to 0.1.

### 3.3 Optimization

We optimize (6) using mini-batch stochastic gradient decent (SGD) and backpropagation. To be specific, there are three kinds of parameters to optimize or update: autoencoder's weights, cluster centers and target distribution $P$.

**Update autoencoder's weights and cluster centers**. Fix target distribution $P$, then the gradients of $L_c$ with respect to embedded point $z_i$ and cluster center $\mu_j$ can be computed as:

$$\frac{\partial L_c}{\partial z_i} = 2\sum_{j=1}^{K} \left(1 + \|z_i - \mu_j\|^2\right)^{-1} (p_{ij} - q_{ij})(z_i - \mu_j) \quad (9)$$

$$\frac{\partial L_c}{\partial \mu_j} = 2\sum_{i=1}^{n} \left(1 + \|z_i - \mu_j\|^2\right)^{-1} (q_{ij} - p_{ij})(z_i - \mu_j) \quad (10)$$

Then given a mini batch with $m$ samples and learning rate $\lambda$, $\mu_j$ is updated by

$$\mu_j = \mu_j - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\partial L_c}{\partial \mu_j} \quad (11)$$

The decoder's weights are updated by

$$W' = W' - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\partial L_r}{\partial W'} \quad (12)$$

The encoder's weights are updated by

$$W = W - \frac{\lambda}{m}\sum_{i=1}^{m} \left(\frac{\partial L_r}{\partial W} + \gamma\frac{\partial L_c}{\partial W}\right) \quad (13)$$

**Update target distribution**. The target distribution $P$ serves as "groundtruth" soft label but also depends on predicted soft label. Therefore, to avoid instability, $P$ should not be updated at each iteration (one update for autoencoder's weights using a batch of samples is called an iteration) using only a batch of data. In practice, we update target distribution using all embedded points every $T$ iterations. See (3) and (4) for the update rules. When update target distribution, the label assigned to $x_i$ is obtained by

$$s_i = \arg\max_j q_{ij} \quad (14)$$

where $q_{ij}$ is computed by (3). We will stop training if label assignment change (in percentage) between two consecutive updates for target distribution is less than a threshold $\delta$.

The whole algorithm is summarized in Algorithm 1.

**Algorithm 1:** Improved Deep Embedded Clustering

**Input:** Input data: $X$; Number of clusters: $K$; Target distribution update interval: $T$; Stopping threshold: $\delta$; Maximum iterations: $MaxIter$.
**Output:** Autoencoder's weights $W$ and $W'$; Cluster centers $\mu$ and labels $s$.

Initialize $\mu, W'$ and $W$ according to Section 3.1.
**for** $iter \in \{0, 1, \ldots, MaxIter\}$ **do**
    **if** $iter\%T == 0$ **then**
        Compute all embedded points $\{z_i = f_W(x_i)\}_{i=1}^n$
        Update $P$ using (3), (4) and $\{z_i\}_{i=1}^n$.
        Save last label assignment: $s_{old} = s$.
        Compute new label assignments $s$ via (14).
        **if** $sum(s_{old} \neq s)/n < \delta$ **then**
            Stop training.
    Choose a batch of samples $S \in X$.
    Update $\mu$, $W'$ and $W$ via (11), (12) and (13) on $S$.

Table 1: Datasets statistics

| Dataset | # examples | # classes | Dimension |
|---|---|---|---|
| MNIST | 70000 | 10 | 784 |
| USPS | 9298 | 10 | 256 |
| REUTERS-10K | 10000 | 4 | 2000 |

## 4 Experiments

### 4.1 DataSets

The proposed IDEC method is evaluated on two image datasets and one text dataset:

- **MNIST**: The MNIST dataset [LeCun *et al.*, 1998] consists of total 70000 handwritten digits of 28x28 pixel size. We reshaped each gray image to a 784 dimensional vector.
- **USPS**: The USPS dataset contains 9298 gray-scale handwritten digit images with size of 16x16 pixels. The features are floating point in [0, 2].
- **REUTERS-10K**: Reuters contains around 810000 English news stories labeled with a category tree [Lewis *et al.*, 2004]. Following DEC [Xie *et al.*, 2016], we used 4 root categories: corporate/industrial, government/social, markets and economics as labels and excluded all documents with multiple labels. Restricted by computational resources, we randomly sampled a subset of 10000 examples and computed tf-idf features on the 2000 most frequent words. The sampled dataset is referred as to REUTERS-10K.

The statistics of these datasets are summarized in Table 1. For all algorithms, we preprocessed datasets as same as DEC, i.e. normalizing each example $x_i \in X$ to $\frac{1}{d}\|x_i\|_2^2 \approx 1$.

### 4.2 Experiment Setup

**Comparing methods**. We demonstrate the effectiveness of our IDEC method mainly by comparing with DEC [Xie *et al.*, 2016] which can be viewed as a special case of IDEC when the reconstruction term is set to zero. we use the publicly available code released by the author to report the performance of DEC. The two-stage deep clustering algorithm is denoted as AE+k-means, which means performing k-means algorithm on embedded features of pretrained autoencoder. This is same as results of DEC and IDEC before training with

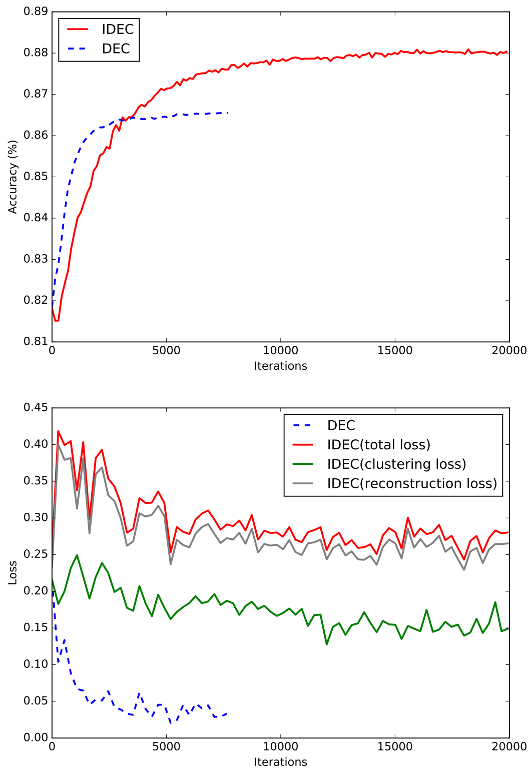


Figure 2: Accuracies (top) and losses (bottom) during training on MNIST. DEC converges much faster but to a much lower accuracy than IDEC. The degradation of IDEC at very beginning is caused by changing objective and switching optimizer from SGD to Adam. The clustering loss of IDEC is larger than DEC's but IDEC still reaches higher accuracy, which implies that clustering loss without constraint tends to break intrinsic structure in order to decrease itself.

clustering loss. For the sake of completeness, two traditional and classic clustering algorithms, K-means and Spectral Embedded Clustering (SEC) [Nie *et al.*, 2011], are also included in comparison. k-means is run 20 times with different initialization and the result with best objective value is chosen. SEC is a variant of spectral clustering with a linearity regularization explicitly added and outperforms traditional spectral clustering methods on a wide range of datasets according to [Nie *et al.*, 2011]. The parameters of SEC are fixed as default value in the code provided by the authors.

**Parameters setting**. Following settings in DEC [Xie *et al.*, 2016], the encoder network is set as a fully connected multilayer perceptron (MLP) with dimensions $d-500-500-2000-10$ for all datasets, where $d$ is the dimension of input data (features). And the decoder network is a mirror of encoder, i.e. a MLP with dimensions $10-2000-500-500-d$. Expect for input and output layers, all internal layers are activated by ReLU nonlinearity function [Glorot *et al.*, 2011]. The autoencoder network pretraining is set exactly the same as [Xie *et al.*, 2016], please refer to the paper for detail. After pretraining, the coefficient $\gamma$ of clustering loss is set

Table 2: Comparison of clustering performance in terms of accuracy (%) and NMI (%, in bracket). The proposed IDEC has best performance on all datasets. The improvement from DEC to IDEC demonstrates the effectiveness of reconstruction loss term, i.e. preserving local structure of data generating distribution is helpful for clustering performance.

| Methods | MNIST | USPS | REUTERS-10K |
|---|---|---|---|
| k-means | 53.24 | 66.82 | 51.62 |
| SEC | 80.37 | N/A | 60.08 |
| AE+k-means | 81.82(74.73) | 69.31(66.20) | 70.52(39.79) |
| DEC | 86.55(83.72) | 74.08(75.29) | 73.68(49.76) |
| IDEC | **88.06(86.72)** | **76.05(78.46)** | **75.64(49.81)** |

to 0.1 and batch size to 256 for all datasets. The optimizer Adam [Kingma and Ba, 2014] with init learning rate $\lambda = 0.001$, $\beta_1 = 0.9, \beta_2 = 0.999$ is applied for MNIST dataset and SGD with learning rate $\lambda = 0.1$ and momentum $\beta = 0.99$ is used for USPS and REUTERS-10K datasets. The convergence threshold is set to $\delta = 0.1\%$. And the update intervals $T$ are 140, 30, 3 iterations for MNIST, USPS and REUTERS-10K respectively. Our implementation is based on Python and Keras [Chollet, 2015] and is available at `https://github.com/XifengGuo/IDEC`.

**Evaluation Metric**. We mainly evaluate clustering methods by clustering accuracy, defined as the best match between ground truth and predicted labels:

$$ACC(\mathbf{y}, \mathbf{c}) = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{y_i = g(c_i)\}}{n} \tag{15}$$

where $y_i$ and $c_i$ are the ground truth and predicted label of sample $x_i$ respectively, and $g$ is a one to one mapping from predicted label to ground truth label. The best mapping can be efficiently computed by the Hungarian algorithm [Kuhn, 1955]. We also report Normalized Mutual Information (NMI) which is also widely used in unsupervised learning scenario:

$$NMI(\mathbf{y}, \mathbf{c}) = \frac{2I(\mathbf{y}, \mathbf{c})}{H(\mathbf{y}) + H(\mathbf{c})} \tag{16}$$

where $I$ and $H$ are mutual information and entropy respectively.

### 4.3 Results

We report the results of all comparing algorithms on 3 datasets in Table 2. As it shows, deep clustering algorithms AE+k-means, DEC and IDEC outperform traditional clustering algorithms k-means and Spectral Embedded Clustering (SEC) [Nie *et al.*, 2011] with a large margin, which indicates the fascinating potentials of deep learning in unsupervised clustering field. The performance gap between AE+k-means and DEC reflects the effect of clustering loss. And the outperformance of IDEC over DEC demonstrates that the autoencoder can help improve clustering performance in some way.

Figure 2 illustrates the behavior of DEC and IDEC during training on MNIST. We observe the following phenomena. First, the final accuracies comply with results in Table 2, i.e. IDEC outperforms DEC. Second, IDEC converges slower than DEC because of the fluctuation of reconstruction loss.

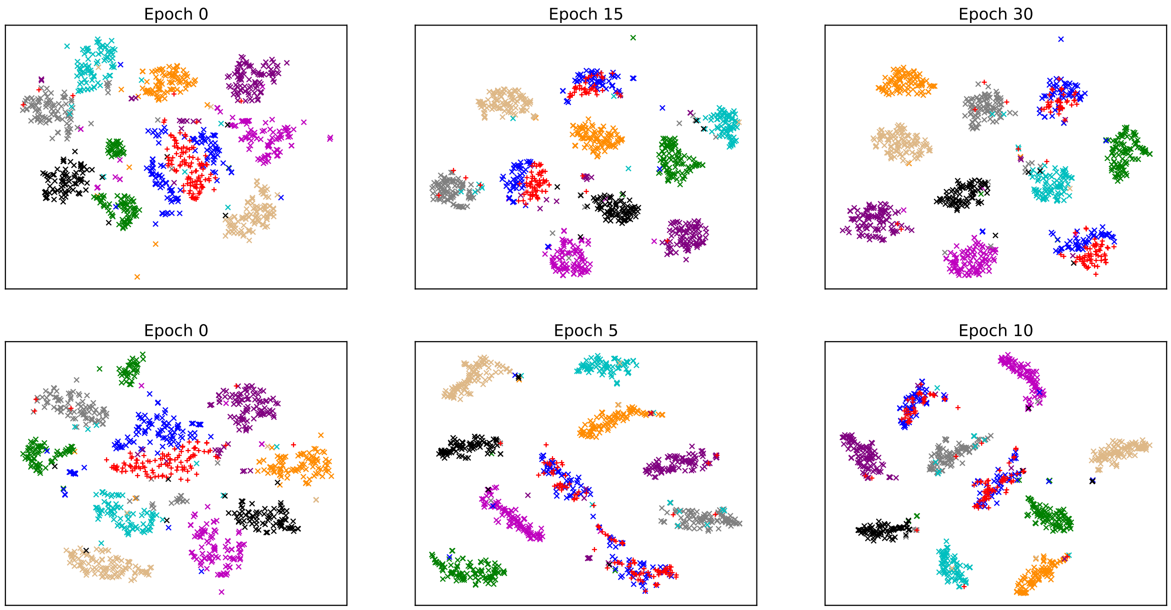


Figure 3: Visualization of clustering results on subset of MNIST during training. Different colors mark different clusters. The first row is ours and second row corresponds to DEC. The proposed IDEC converges slower since it optimizes reconstruction loss as well. Both methods separate clusters well but the data structure in the first row is preserved better than DEC. Note points with red and blue color, they are totally mixed together in DEC while still somehow separable in our IDEC.

Third, IDEC has larger clustering loss and higher clustering accuracy than DEC. This implies that the objective of DEC may mislead the clustering procedure by distorting the embedded feature space and breaking the intrinsic structure of data. Finally, the reconstruction losses at last few iterations approximately equal the loss at beginning. It implies that the performance improvement from DEC to IDEC is not likely due to the clustering ability of autoencoder. Actually, we did conduct an experiment that fine tune the autoencoder only using reconstruction loss $L_r$ (by setting coefficient $\gamma$ in (6) to 0) via various optimizers, and no improvement in terms of clustering accuracy was observed. So we assume that the autoencoder plays the role of preserving local structure of data, and under this condition clustering loss can manipulate embedded space to get better clustering accuracy.

We further prove our assumption about the role autoencoder acts by visualizing the embedded feature space during training. The t-SNE [Maaten and Hinton, 2008] visualization on a random subset of MNIST with 1000 samples is shown in Fig. 3. From left to right in the top row, the training process of IDEC, the "shape" of each cluster is almost maintained. On the contrary, the "shape" in the bottom is changed a lot with training proceeding. Further more, when you focus on clusters colored by red and blue (digits 4 and 9), in the first column they are still separable but become distinguishable in the last column. This is a loophole of DEC's objective (clustering loss). Our IDEC doesn't overcome this problem, but does go further than DEC. To validate this, see the figures in the last column: blue and red clusters of IDEC are still somehow separable while in DEC they are totally mixed up. It can be concluded that the autoencoder can preserve the intrinsic structure of data generating distribution and hence help clustering loss to manipulate the embedded feature space *appropriately*.

## 5 Conclusion

This paper proposes Improved Deep Embedded Clustering (IDEC) algorithm, which jointly performs clustering and learns embedded features that are suitable for clustering and preserve local structure of data generating distribution. IDEC manipulates feature space to scatter data by optimizing a KL divergence based clustering loss with a self-training target distribution. And it maintains the local structure by incorporating an autoencoder. Empirical experiments demonstrate that structure preservation is vital to deep clustering algorithm and can favor clustering performance. Future work includes: adding more prior knowledge (e.g. sparsity) in IDEC framework, and modifying the clustering loss to separate close clusters like "4" and "9" in MNIST.

## Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (Project no. 60970034, 61170287, 61232016 and 61672528).

## References

[Bengio *et al.*, 2013] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising autoencoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[Bishop, 2006] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[Chen *et al.*, 2016] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[Chen, 2015] Gang Chen. Deep learning with nonparametric clustering. *arXiv preprint arXiv:1501.03084*, 2015.

[Chollet, 2015] François Chollet. Keras, 2015.

[Glorot *et al.*, 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *Journal of Machine Learning Research*, 15:315–323, 2011.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Goodfellow *et al.*, 2016] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[Hinton *et al.*, 2006] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kuhn, 1955] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lewis *et al.*, 2004] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[MacQueen, 1967] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[Nie *et al.*, 2011] Feiping Nie, Zinan Zeng, Ivor W Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011.

[Nigam and Ghani, 2000] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93. ACM, 2000.

[Peng *et al.*, 2016] Xi Peng, Shijie Xiao, Jiashi Feng, Wei-Yun Yau, and Zhang Yi. Deep subspace clustering with sparsity prior. In *The 25th International Joint Conference on Artificial Intelligence*, 2016.

[Tian *et al.*, 2014] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, pages 1293–1299, 2014.

[Vincent *et al.*, 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010.

[Von Luxburg, 2007] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[Xie *et al.*, 2016] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016.

[Yang *et al.*, 2016] Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5147–5156, 2016.