Deep Clustering with Convolutional Autoencoders

Xifeng Guo[1], Xinwang Liu[1], En Zhu[1], and Jianping Yin[2]

[1] College of Computer,
National University of Defense Technology, Changsha, 410073, China
guoxifeng13@nudt.edu.cn
[2] State Key Laboratory of High Performance Computing,
National University of Defense Technology, Changsha, 410073, China

Abstract. Deep clustering utilizes deep neural networks to learn feature representation that is suitable for clustering tasks. Though demonstrating promising performance in various applications, we observe that existing deep clustering algorithms either do not well take advantage of convolutional neural networks or do not considerably preserve the local structure of data generating distribution in the learned feature space. To address this issue, we propose a deep convolutional embedded clustering algorithm in this paper. Specifically, we develop a convolutional autoencoders structure to learn embedded features in an end-to-end way. Then, a clustering oriented loss is directly built on embedded features to jointly perform feature refinement and cluster assignment. To avoid feature space being distorted by the clustering loss, we keep the decoder remained which can preserve local structure of data in feature space. In sum, we simultaneously minimize the reconstruction loss of convolutional autoencoders and the clustering loss. The resultant optimization problem can be effectively solved by mini-batch stochastic gradient descent and back-propagation. Experiments on benchmark datasets empirically validate the power of convolutional autoencoders for feature learning and the effectiveness of local structure preservation.

Keywords: Deep Clustering, Convolutional Autoencoders, Convolutional Neural Networks, Unsupervised Learning

1 Introduction

Given a large collection of unlabeled images represented by raw pixels, how to divide them into K groups in terms of inherent latent semantics? The traditional way is first extracting feature vectors according to domain-specific knowledges and then employing clustering algorithm on the extracted features. Thanks to deep learning approaches, some work successfully combines feature learning and clustering into a unified framework which can directly cluster original images with even higher performance. We refer to this new category of clustering algorithms as *Deep Clustering*.

Some researches have been conducted, but what are the critical ingredients for deep clustering still remains unclear. For example, what types of neural networks are proper for feature extraction? How to provide guidance information i.e. to define clustering oriented loss function? Which properties of data should be preserved in feature space? In this paper, we focus on the first and third questions and conclude that Convolutional AutoEncoders (CAE) and locality property are two of key ingredients for deep clustering algorithms.

The most widely used neural networks in deep clustering algorithms are Stacked AutoEncoders (SAE) [12, 15, 17, 11]. The SAE requires layer-wise pretraining before being finetuned in an end-to-end manner. When the layers go deeper, the pretraining procedure can be tedious and time-consuming. Furthermore, SAE is built with fully connected layers, which are ineffective for dealing with images. The work in [7] is the first trial to train CAE directly in an end-to-end manner without pretraining.

In terms of properties of data to preserve in feature space, the primitive work considers sparsity or graph constraints by adding prior knowledges to the objective [14, 12]. They are two-stage algorithms: feature learning and then clustering. Latter, algorithms that jointly accomplish feature learning and clustering come into being [15, 18]. The Deep Embedded Clustering (DEC) [15] 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. However, they ignore the preservation of data properties, which may lead to the corruption of feature space. We improve DEC algorithm by preserving local structure of data generating distribution and by incorporating convolutional layers.

Our key idea is that CAE is beneficial to learning features for images and preserving local structure of data avoids distortion of feature space. The contributions are:

- A Convolutional AutoEncoders (CAE) that can be trained in end-to-end manner is designed for learning features from unlabeled images. The designed CAE is superior to stacked autoencoders by incorporating spacial relationships between pixels in images. We show that convolutional layer, convolutional transpose layer and fully connected layer are sufficient for constructing an effective CAE.
- The local structure preservation is considered during tuning network parameters according to clustering oriented loss function. We demonstrate that preserving local structure helps stabilize the training procedure and avoid the corruption of feature space.
- We propose the Deep Convolutional Embedded Clustering (DCEC) algorithm to automatically cluster images. The DCEC takes advantages of CAE and local structure preservation. And the resulting optimization problem can be efficiently solved by mini-batch stochastic gradient descent and backpropagation.
- Extensive experiments are conducted on benchmark image datasets. The results validate the effectiveness of CAE and local structure preservation.

2 Convolutional AutoEncoders

A conventional autoencoder is generally composed of two layers, corresponding to encoder $f_W(\cdot)$ and decoder $g_U(\cdot)$ respectively. It aims to find a code for each input sample by minimizing the mean squared errors (MSE) between its input and output over all samples, i.e.

$$\min_{W,U} \frac{1}{n} \sum_{i=1}^{n} \|g_U(f_W(x_i)) - x_i\|_2^2 \tag{1}$$

For fully connected autoencoder,

$$\begin{aligned} f_W(x) &= \sigma(Wx) \equiv h \\ g_U(h) &= \quad \sigma(Uh) \end{aligned} \tag{2}$$

where x and h are vectors, and σ is activation function like ReLU, sigmoid. Note that the bias is omitted for convenient description. After training, the embedded code h serves as the new representation of input sample. Then h can be fed into another autoencoder to form Stacked AutoEncoders (SAE). To exploit the spacial structure of images, convolutional autoencoder is defined as

$$\begin{aligned} f_W(x) &= \sigma(x * W) \equiv h \\ g_U(h) &= \quad \sigma(h * U) \end{aligned} \tag{3}$$

where x and h are matrices or tensors, and "$*$" is convolution operator. The Stacked Convolutional AutoEncoders (SCAE) [9] can be constructed in a similar way as SAE.

We propose a new Convolutional AutoEncoders (CAE) that does not need tedious layer-wise pretraining, as shown in Fig. 1. First, some convolutional layers are stacked on the input images to extract hierarchical features. Then flatten all units in the last convolutional layer to form a vector, followed by a fully connected layer with only 10 units which is called embedded layer. The input 2D image is thus transformed into 10 dimensional feature space. To train it in the unsupervised manner, we use a fully connected layer and some convolutional transpose layers to transform embedded feature back to original image. The parameters of encoder $h = F_\omega(x)$ and decoder $x' = G_{\omega'}(h)$ are updated by minimizing the reconstruction error:

$$L_r = \frac{1}{n} \sum_{i=1}^{n} \|G_{\omega'}(F_\omega(x_i)) - x_i\|_2^2 \tag{4}$$

where n is the number of images in dataset, $x_i \in \mathbb{R}^2$ is the ith image.

The key factor of the proposed CAE is the aggressive constraint on the dimension of embedded layer. If the embedded layer is large enough, the network may be able to copy its input to output, leading to learning useless features. The intuitive way of avoiding identity mapping is to control the dimension of latent

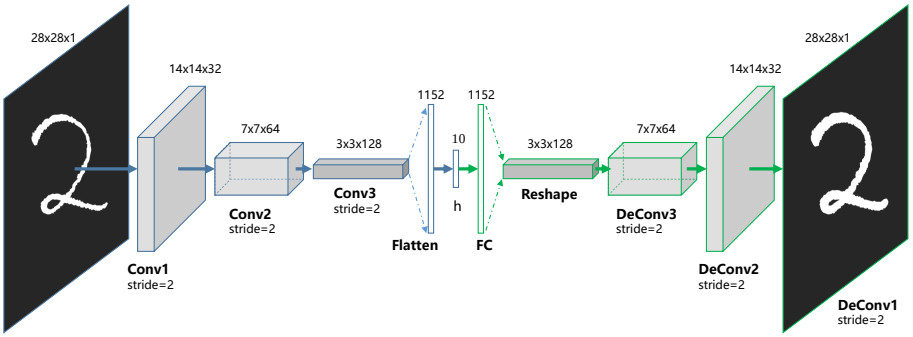

Fig. 1. The structure of proposed Convolutional AutoEncoders (CAE) for MNIST. In the middle there is a fully connected autoencoder whose embedded layer is composed of only 10 neurons. The rest are convolutional layers and convolutional transpose layers (some work refers to as Deconvolutional layer). The network can be trained directly in an end-to-end manner.

code h lower than input data x. Learning such under-complete representations forces the autoencoder to capture the most salient features of the data. Thus we force the dimension of embedded space to equal to the number of clusters of dataset. In this way, the network can be trained directly in an end-to-end manner even without any regularizations like Dropout [13] or Batch Normalization [4]. The learned compact representations are proved effective for clustering task.

Another factor is that we utilize convolutional layer with stride instead of convolutional layer followed by pooling layer in the encoder, and convolutional transpose layer with stride in the decoder. Because the convolutional (transpose) layers with stride allow the network to learn spacial subsampling (upsampling) from data, leading to higher capability of transformation.

Note that we do not aim at the state-of-the-art clustering performance, so we do not adopt fancy layers or techniques like BatchNormalization layer, LeakyReLu activation or layer-wise pretraining.We only show the CAE is superior to fully connected SAE in image clustering task.

3 Deep Convolutional Embedded Clustering

As introduced in Sect. 2, the CAE is a more powerful network for dealing with images compared with fully connected SAE. So we extend Deep Embedded Clustering (DEC) [15] by replacing SAE with CAE. Then we argue that the embedded feature space in DEC may be distorted by only using clustering oriented loss. To this end, the reconstruction loss of autoencoders is added to the objective and optimized along with clustering loss simultaneously. The autoencoders will preserve the local structure of data generating distribution, avoiding the corruption of feature space. The resulting algorithm is termed as Deep Convolutional Embedded Clustering (DCEC). In the following sections, we first give the struc-

ture of DCEC, then introduce the clustering loss and local structure preservation mechanism in detail. At last, the optimization procedure is provided.

3.1 Structure of Deep Convolutional Embedded Clustering

The DCEC structure is composed of CAE (see Fig. 1) and a clustering layer which is connected to the embedded layer of CAE, as depicted in Fig. 2. The clustering layer maps each embedded point z_i of input image x_i into a soft label. Then the clustering loss L_c is defined as Kullback-Leibler divergence (KL divergence) between the distribution of soft labels and the predefined target distribution. CAE is used to learn embedded features and the clustering loss guides the embedded features to be prone to forming clusters.

The objective of DCEC is

$$L = L_r + \gamma L_c \tag{5}$$

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 $L_r \equiv 0$, (5) reduces to the objective of DEC [15].

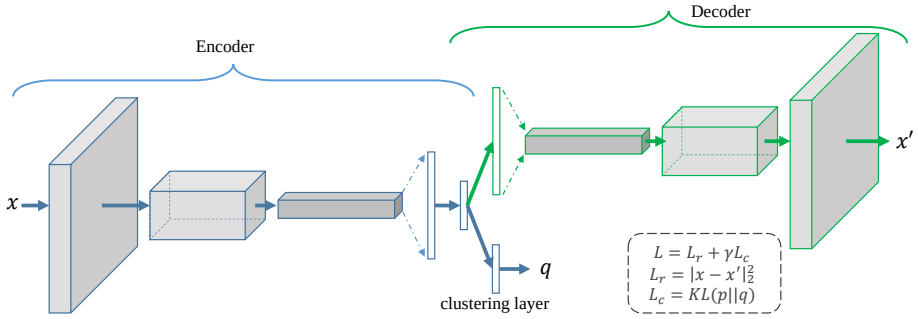

Fig. 2. The structure of deep convolutional embedded clustering (DCEC). It is composed of a convolutional autoencoders and a clustering layer connected to embedded layer of autoencoders.

3.2 Clustering Layer and Clustering Loss

The clustering layer and loss are directly borrowed from DEC [15]. We briefly review their definitions for completeness of DCEC structure.

The clustering layer maintains cluster centers $\{\mu_j\}_1^K$ as trainable weights and maps each embedded point z_i into soft label q_i by Student's t-distribution [8]:

$$q_{ij} = \frac{(1 + \|z_i - \mu_j\|^2)^{-1}}{\sum_j (1 + \|z_i - \mu_j\|^2)^{-1}} \tag{6}$$

where q_{ij} is the jth entry of q_i, representing the probability of z_i belonging to cluster j.

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 P is the target distribution, defined as

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j \left(q_{ij}^2 / \sum_i q_{ij}\right)} \tag{8}$$

3.3 Reconstruction Loss for Local Structure Preservation

DEC [15] abandons the decoder and finetunes the 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 layer.

As shown in [12] and [3], autoencoders can preserve local structure of data generating distribution. Under this condition, manipulating embedded space slightly using clustering loss L_c will not cause corruption. So the coefficient γ is better to be less than 1, which will be empirically fixed to 0.1 for all experiments.

3.4 Optimization

We first pretrain the parameters of CAE by setting $\gamma = 0$ to get meaningful target distribution. After pretraining, the cluster centers are initialized by performing k-means on embedded features of all images. Then set $\gamma = 0.1$ and update CAE's weights, cluster centers and target distribution P as follows.

Update autoencoders' weights and cluster centers. As $\frac{\partial L_c}{\partial z_i}$ and $\frac{\partial L_c}{\partial \mu_j}$ are easily derived according to [15], then the weights and centers can be updated by using backpropagation and mini-batch SGD straightforwardly.

Update target distribution. The target distribution P serves as ground truth soft label but also depends on predicted soft label. Therefore, to avoid instability, P should not be updated at each iteration using only a batch of data. In practice, we update target distribution using all embedded points every T iterations. See (6) and (8) for the update rules.

The training process terminates if the change of label assignments between two consecutive updates for target distribution is less than a threshold δ.

4 Experiment

4.1 DataSets

The proposed DCEC method is evaluated on three image datasets: **MNIST-full**: The MNIST dataset [6] consists of total 70000 handwritten digits of 28x28

pixels. **MNIST-test**: The test set of MNIST, containing 10000 images. **USPS**: The USPS dataset contains 9298 gray-scale handwritten digit images with size of 16x16 pixels.

4.2 Experiment Setup

Comparing methods. We demonstrate the effectiveness of our DCEC algorithm mainly by comparing with **DEC** [15]. The two-stage deep clustering algorithm is denoted as **SAE+k-means** (or **CAE+k-means**), i.e. performing k-means on embedded features of pretrained SAE (or CAE). **IDEC** [16] denotes the algorithm that adds reconstruction loss L_r to DEC's objective. Note that the difference between the result of **IDEC** [16] reported in Table 1 and that in their paper [16] is due to the different pretraining strategy (see next paragraph for details). **DEC-conv** is the structure that directly replaces SAE in DEC with CAE but without L_r. **DCEC** is the proposed structure, which adds both L_r and convolutional layers to DEC. For the sake of completeness, two traditional and classic clustering algorithms, **k-means** and Spectral Embedded Clustering (**SEC**) [10], are also included in comparison.

Parameters setting. For SAE+k-means, DEC [15] and IDEC [16], 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. Except for input, output and embedding layers, all internal layers are activated by ReLU nonlinearity function [2]. The SAE is pretrained end-to-end for 400 epochs using SGD with learning rate 0.01 and momentum 0.9.

For CAE+k-means, DEC-conv and DECE, the encoder network structure is $\text{conv}^5_{32} \rightarrow \text{conv}^5_{64} \rightarrow \text{conv}^3_{128} \rightarrow \text{FC}_{10}$ where conv^k_n denotes a convolutional layer with n filters, kernel size of $k \times k$ and stride length 2 as default. The decoder is a mirror of encoder. The CAE is pretrained end-to-end for 200 epochs using Adam [5] with default parameters. The convergence threshold is set to $\delta = 0.1\%$. And the update intervals $T = 140$. Our implementation is based on Python and Keras [1] and the code is available at `https://github.com/XifengGuo/DCEC`.

Evaluation Metric. All clustering methods are evaluated by clustering accuracy (ACC) and Normalized Mutual Information (NMI) which are widely used in unsupervised learning scenario.

4.3 Results

The clustering results are shown in Table 1. Our DCEC algorithm outperforms all opponents in terms of ACC and NMI on all datasets.

Advantage of CAE. SAE+k-means, DEC and IDEC share the same pretrained SAE network structure and weights. As a counterpart, CAE+k-means, DEC-conv and DCEC use the same CAE structure and weights. By comparing each pair of equivalents (like IDEC and DCEC), we see that methods using CAE outperform their counterparts that use SAE by a large margin. Notice that, at

Table 1. Comparison of clustering performance in terms of accuracy (%). The results of DEC† [15] is obtained by using publicly available code. DEC shares the pretrained weights with SAE+k-means and IDEC [16] for fair comparison.

Methods	MNIST-full		MNIST-test		USPS	
	ACC	NMI	ACC	NMI	ACC	NMI
k-means	54.24	48.52	54.63	50.18	66.82	62.66
SEC [10]	80.37	–	–	–	–	–
DEC† [15]	86.55	83.72	82.36	79.58	73.68	75.29
SAE+k-means	78.17	71.46	66.81	59.59	61.65	57.27
DEC	84.08	81.28	69.94	67.69	69.28	70.18
IDEC [16]	84.21	83.81	71.45	69.40	72.10	73.23
CAE+k-means	84.90	79.27	79.00	72.55	74.15	73.30
DEC-conv	88.63	87.59	84.83	82.62	77.90	81.08
DCEC	**88.97**	**88.49**	**85.29**	**83.61**	**79.00**	**82.57**

pretraining stage, CAE is trained for 200 epochs while SAE for 400 epochs. And at clustering stage, methods with CAE converge much faster than SAE counterparts. This demonstrates that CAE is superior to SAE in image clustering task.

Local structure preservation. We can see the effect of adding reconstruction loss by comparing DEC and DEC-reco (or DEC-conv and DCEC). The clustering performance of IDEC are higher than that of DEC. And the same is true for DEC-conv and DCEC. We assume that this superiority is due to the fact that autoencoders can preserve local structure of data by minimizing the reconstruction loss. We validate this property by visualizing the embedded features. The t-SNE [8] visualization on a random subset of MNIST-full with 1000 samples is shown in Fig. 3. For DCEC, the "shape" of each cluster is almost maintained compared with pretrained CAE. Furthermore, when you focus on clusters colored by red and blue (digits 4 and 9), in DCEC they are still somehow separable but totallydistinguishable in DEC-conv. 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*. By comparing DEC and IDEC, the same conclusion was conducted in [16].

5 Related Work

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, [14, 12] use autoencoder to learn low dimensional features of original graph or data samples, and then runs conventional clustering algorithm like k-means and non-parametric maximum-margin clustering on learned representations.

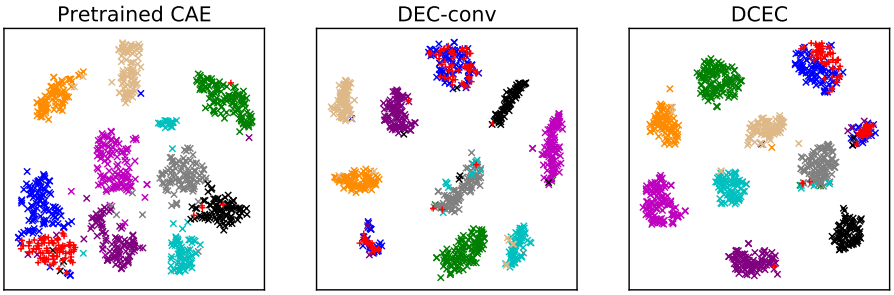

Fig. 3. Visualization of clustering results on subset of MNIST-full. Different colors mark different clusters. The data structure in DCEC is preserved better than DEC-conv. Note points with red and blue colors, they are totally mixed together in DEC-conv while still somehow separable in our DCEC.

The other category of algorithms try to explicitly define a clustering loss, simulating classification error in supervised deep learning. [18] proposes a recurrent framework, which integrates feature learning and clustering into a single model with a unified weighted triplet loss and optimizes it end-to-end. DEC [15] learns a mapping from the observed space to a low-dimensional latent space with SAE, which can obtain feature representations and cluster assignments simultaneously. DBC [7] improves DEC by replacing SAE with CAE. And IDEC [16] adds reconstruction loss of autoencoders to the objective of DEC [15].

The proposed DCEC falls into the second category. It excels [18] by simplicity without recurrent and outperforms DEC in terms of clustering accuracy and feature's representativeness. DBC [7] studied the CAE but still neglected the local structure preservation problem. While IDEC [16] did not incorporate convolutional layers. Our DCEC takes care of both convolutional networks and local structure preservation.

6 Conclusion

This paper proposes a Deep Convolutional Embedded Clustering (DCEC) algorithm to take advantage of both convolutional neural networks and local structure preservation mechanism. DCEC is a framework that jointly learns deep representations of images and performs clustering. It learns good features with local structure preserved by using Convolutional AutoEncoders (CAE) and manipulates feature space by incorporating a clustering oriented loss. The experiment empirically demonstrates the effectiveness of DCEC on image clustering task and validates that both convolutional networks and local structure preservation mechanism are vital to deep clustering for images. The future work include conducting more experiments on high dimensional image datasets and exploring more advanced convolutional networks.

Acknowledgments

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

References

1. Chollet, F., et al.: Keras. `https://github.com/fchollet/keras` (2015)
2. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. Journal of Machine Learning Research 15, 315–323 (2011)
3. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT Press (2016)
4. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML). pp. 448–456 (2015)
5. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
7. Li, F., Qiao, H., Zhang, B., Xi, X.: Discriminatively boosted image clustering with fully convolutional auto-encoders. arXiv preprint arXiv:1703.07980 (2017)
8. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research 9(Nov), 2579–2605 (2008)
9. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional auto-encoders for hierarchical feature extraction. International Conference on Artificial Neural Networks pp. 52–59 (2011)
10. Nie, F., Zeng, Z., Tsang, I.W., Xu, D., Zhang, C.: Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. IEEE Transactions on Neural Networks 22(11), 1796–1808 (2011)
11. Peng, X., Feng, J., Lu, J., Yau, W.Y., Yi, Z.: Cascade subspace clustering. In: AAAI Conference on Artificial Intelligence (AAAI). pp. 2478–2484 (2017)
12. Peng, X., Xiao, S., Feng, J., Yau, W.Y., Yi, Z.: Deep subspace clustering with sparsity prior. In: International Joint Conference on Artificial Intelligence (IJCAI) (2016)
13. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
14. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: AAAI Conference on Artificial Intelligence (AAAI). pp. 1293–1299 (2014)
15. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: International Conference on Machine Learning (ICML) (2016)
16. Xifeng Guo, Long Gao, X.L.J.Y.: Improved deep embedded clustering with local structure preservation. In: International Joint Conference on Artificial Intelligence (IJCAI-17). pp. 1753–1759 (2017)
17. Yang, B., Fu, X., Sidiropoulos, N.D., Hong, M.: Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In: International Conference on Machine Learning (ICML). pp. 3861–3870 (2017)
18. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5147–5156 (2016)