DEEP CLUSTERING BASED ON A MIXTURE OF AUTOENCODERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper we propose a Deep Autoencoder Mixture Clustering (DAMIC) algorithm. It is based on a mixture of deep autoencoders where each cluster is represented by an autoencoder. A clustering network transforms the data into another space and then selects one of the clusters. Next, the autoencoder associated with this cluster is used to reconstruct the data-point. The clustering algorithm jointly learns the nonlinear data representation and the set of autoencoders. The optimal clustering is found by minimizing the reconstruction loss of the mixture of autoencoder network. Unlike other deep clustering algorithms, no regularization term is needed to avoid data collapsing to a single point. Our experimental evaluations on image and text corpora show significant improvement over state-of-the-art methods.

1 INTRODUCTION

Effective automatic grouping of objects into clusters is one of the fundamental problems in machine learning and data analysis. In many approaches, the first step toward clustering a dataset is extracting a feature vector from each object. This reduces the problem to the aggregation of groups of vectors in a feature space. A commonly used clustering algorithm in this case is the k-means. Clustering high-dimensional datasets is, however, hard because inter-point distances become less informative in high-dimensional spaces. As a result, representation learning has been used to map the input data into a low-dimensional feature space. In recent years, motivated by the success of deep neural network in supervised learning, there are many attempts to apply unsupervised deep learning approaches for clustering. Most methods are focused on clustering over a low-dimensional feature space of an autoencoder or a variational autoencoder. Recent good overviews of deep clustering methods can be found in Aljalbout et al. (2018) and Min et al. (2018).

Using deep neural networks, it is possible to learn nonlinear mappings allowing to transform the data into more clustering-friendly representations. A deep version of k-means is based on learning a data representation and applying k-means in the embedded space. A straightforward implementation of the deep k-means algorithm would lead, however, to a trivial solution where the features are collapsed to a single point in the embedded space and the centroids are collapsed into a single entity. The objective function of most deep clustering algorithms is, therefore, composed of a clustering term computed in the embedded space and a regularization term in the form of reconstruction error to avoid data collapsing. Deep Embedded Clustering (DEC) (Xie et al. (2016)) is first pre-trained using an autoencoder reconstruction loss and then optimizes cluster centroids in the embedded space through a Kullback-Leibeler divergence loss. Deep Clustering network (DCN) (Yang et al. (2017)) is another autoencoder-based method that uses k-means for clustering. Similar to DEC, in the first phase, the network is pre-trained using the autoencoder reconstruction loss. In the second phase, in contrast to DEC, the network is jointly trained using a mathematical combination of the autoencoder reconstruction loss and the k-means clustering loss function. Thus, due to the fact that strict cluster assignments were used during the training (instead of probabilities such as in DEC) the method requires an alternation process between the network training and the cluster updates.

In this paper we propose an algorithm to perform unsupervised clustering within the mixture-of-experts framework (Jacobs et al. (1991)). Each cluster is represented by an autoencoder neural-network and the clustering itself is performed in a low-dimensional embedded space by a softmax classification layer that directs the input data to the most suitable autoencoder. Unlike most deep

clustering algorithms the proposed algorithm is deep in nature and not a deep variant of a classical clustering algorithm. The proposed algorithm does not suffer from the clustering collapsing problem and therefore there is no need for regularization terms that have to be tuned separately for each dataset. Note that parameter tuning in clustering is problematic since it is based, either explicitly or implicitly, on the data labels which are not supposed to be available in the clustering process. Another major difference of the proposed method from previous approaches is the learning method of the embedding latent space where the actual clustering operation is taking place. In most previous methods, the embedded space is controlled by an autoencoder. Thus, in order to gain a good reconstruction, it requires to encode into the embedded space information that can be entirely irrelevant to the clustering process. In contrast, in our algorithm no decoding is applied to the clustering embedded space and the only goal of the embedded space is to find a good organization of the data into separated clusters.

We validate the method on standard real datasets including various document and image corpora. Evidently, visible improvement from the respective state-of-art is observed for all the tested datasets. The contribution of this paper is twofold: (i) a novel deep learning clustering method that unlike deep variants of k-means, does not require a tuned regularization term to avoid clustering collapsing to a single point; and (ii) state-of-the-art performance on standard datasets.

2 MIXTURE OF AUTOENCODERS

Consider the problem of clustering a set of n points $x_1, \ldots, x_n \in R^d$ into k clusters. The k-means algorithm represents each cluster by a centroid. In our approach, rather than representing a cluster by a centroid, we represent each cluster by an autoencoder that is specialized in reconstructing objects belonging to that cluster. The clustering itself is carried out by directing the input object to the most suitable autoencoder.

We next formally describe the proposed clustering algorithm. The algorithm is based on a (soft) clustering network that produces a distribution over the k clusters:

$$p(c = i|x; \theta_c) = \frac{\exp(w_i h(x) + b_i)}{\sum_{j=1}^{k} \exp(w_j h(x) + b_j)}, \qquad i = 1, \ldots, k \tag{1}$$

such that θ_c is the parameter set of the clustering network, $h(x)$ is a nonlinear representation of a point x computed by the clustering network and $w_1, \ldots, w_k, b_1, \ldots, b_k \in \theta_c$ are the parameters of the softmax output layer. The (hard) cluster assignment of a point x is thus:

$$\hat{c} = \arg\max_{i=1}^{k} p(c = i|x; \theta_c) = \arg\max_{i=1}^{k} (w_i h(x) + b_i). \tag{2}$$

The clustering task is, by definition, unsupervised and therefore we cannot directly train the clustering network. Instead, we use the clustering results to obtain a more accurate reconstruction of the network input. We represent each cluster by an autoencoder that is specializing in reconstructing instances of that cluster. If the dataset is properly clustered, we expect that all the points assigned to the same cluster are similar and hence the task of a cluster-specialized autoencoder should be relatively easy compared to a single autoencoder for the entire data. Hence, we expect that a good clustering should results in a small reconstruction error. Denote the autoencoder associated with cluster i by $f_i(x; \theta_i)$ where θ_i is the parameter-set of the network autoencoder. We can view the reconstructed object $f_i(x; \theta_i) \in R^d$ as a data-driven centroid of the cluster i that is tuned for the input x. The goal of the training procedure is to find a clustering of the data such that the error of the cluster-based reconstruction is minimized.

The clustering is thus computed by minimizing the following loss function:

$$L(\theta_1, \ldots, \theta_k, \theta_c) = -\sum_{t=1}^{n} \log \left(\sum_{i=1}^{k} p(c_t = i|x_t; \theta_c) \exp(-d(x_t, f_i(x_t; \theta_i))) \right) \tag{3}$$

such that $d(x_t, f_i(x_t; \theta_i))$ is the reconstruction error of the i-th autoencoder. In our implementation we set $d(x_t, f_i(x_t; \theta_i)) = \frac{1}{2}\|x_t - f_i(x_t; \theta_i)\|^2$.

In the minimization of (3) we simultaneously perform data clustering in the embedded space $h(x)$ and learn a 'centroid' representation for each cluster in the form of an autoencoder. Unlike most

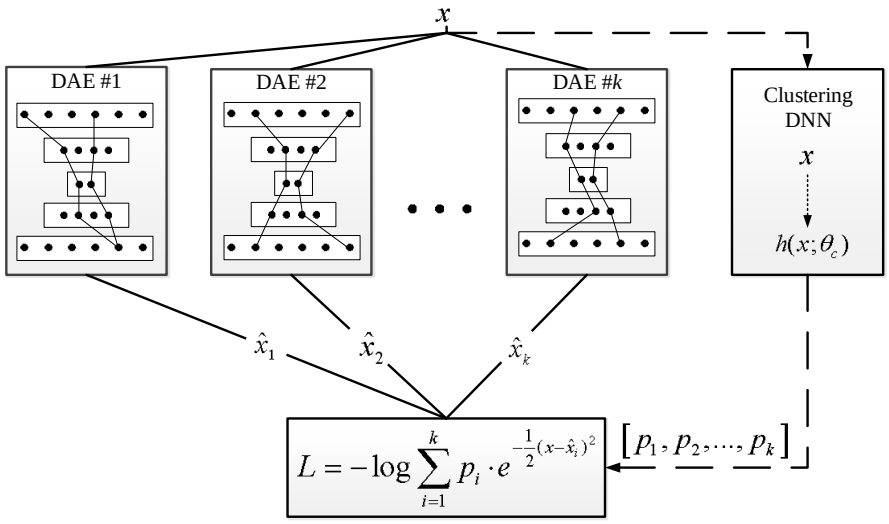


Figure 1: A block diagram of the proposed mixture of deep auto-encoders for clustering.

of previously proposed deep clustering methods, there is no risk of collapsing to a trivial solution where all the data points are transformed to the same vector, even though the clustering is carried out in the embedded space. Collapsing all the data points into a single vector in the embedded space will result in directing all the points to the same autoencoder for reconstruction. As our clustering goal is to minimize the reconstruction error, this situation is, of course, worse than using k different autoencoder for reconstruction. Hence, there is no need for adding regularization terms to the loss function (that might influence the clustering accuracy) to prevent data collapsing. Specifically, there is no need to add a decoder to the embedded space, where the clustering is actually performed, to prevent data collapsing.

The back-propagation equation for the parameter set of the clustering network is:

$$\frac{\partial L}{\partial \theta_c} = -\sum_{t=1}^{n}\sum_{i=1}^{k} w_{ti} \cdot \frac{\partial}{\partial \theta_c} \log p(c_t = i|x_t;\theta_c) \tag{4}$$

such that

$$w_{ti} = \frac{p(c_t = i|x_t;\theta_c)\exp(-d(x_t, f_i(x_t;\theta_i)))}{\sum_{j=1}^{k} p(c_t = j|x_t;\theta_c)\exp(-d(x_t, f_j(x_t;\theta_j)))} \tag{5}$$

is a soft assignment of x_t into the i-th cluster based on the current parameters-set. In other words, the reconstruction error of the autoencoders is used to obtain soft labels that are used for training the clustering network.

In the last few years network pre-training has been largely obsoleted for supervised tasks due to availability of large labeled training datasets. However, for handling hard optimization problems paused by unsupervised clustering tasks, like that in (1), initialization is crucial. To initialize the parameters of the network, we first train a single autoencoder and use the layer-wise pre-training method as in (Bengio et al. (2007)) for training autoencoders. After training the autoencoder, we carry out a k-means clustering on the output of the bottleneck layer to obtain initial clustering values. The k-means assigns a label for each data point. Note, that in the pre-training procedure a *single* autoencoder is trained with all the database. We use these labels as a supervision to pre-train the clustering network (1). The points that were assigned by the k-means algorithm to cluster i are next used to pre-train the i-th autoencoder $f_i(x;\theta_i)$. Once all the network parameter are initialized by this pre-training procedure, the network parameters are jointly trained to minimize the autoencoding reconstruction error defined by the loss function (3). We dub the proposed algorithm Deep Autoen-

codr MIxture Clustering (DAMIC). The architecture of the network that is trained by the DAMIC algorithm is shown in Fig. 1 and the clustering algorithm is summarized in Table 1.

The DAMIC algorithm can be viewed as an extension of the k-means algorithm. Assume we replace each autoencoder in our network by a constant function $f_i(x_t, \theta_i) \equiv \mu_i \in R^d$ and we replace the clustering network by a hard decision based on the reconstruction error, then we obtain exactly the classical k-means algorithm. The DAMIC algorithm replaces the constant centroid with a data driven representation of the input computed by an autoencoder.

The probabilistic modeling used by the DAMIC clustering algorithm can be viewed as an instance of mixture-of-experts (MoE) model introduced by Jacobs et al. (1991) and Jordan & Jacobs (1994). The MoE model is comprised of several expert models and a gate model. Each of the experts provides a decision and the gate is a latent variable that selects the relevant expert based on the input data. In spite of the huge success of deep learning, there are only a few studies that have explicitly utilized and analyzed MoEs as an architecture component of a neural network (Eigen et al. (2014); Shazeer et al. (2017)). MoE has been mostly applied to supervised tasks such as classification and regression. In our clustering algorithm the clustering network is the equivalent of the MoE gating function. The experts here are autoencoders in which each autoencoder expertise is reconstructing a sample from the associated cluster. Our clustering cost function (3) is following the training strategy proposed in Jacobs et al. (1991), which prefers error function that encourages expert specialization instead of cooperation.

Table 1: The Deep Autoencodr MIxture Clustering (DAMIC) algorithm.

Goal: clustering $x_1, \ldots, x_n \in R^d$ into k clusters.

Network components:

- A nonlinear representation: $x \to h(x; \theta_c)$
- A linear classifier: $p(c = i|x; \theta_c) = \frac{\exp(w_i h(x) + b_i)}{\sum_{j=1}^{k} \exp(w_j h(x) + b_j)}$
- A set of autoencoders (one for each cluster): $f_i(x_t; \theta_i), \qquad i = 1, \ldots, k$

Pre-training:

- Train a single autoencoder for the entire dataset.
- Apply k-means algorithm in the embedded space.
- Use the k-means clustering to initialize the network parameters.

Clustering is obtained by minimizing the reconstruction error:

$$L(\theta_1, \ldots, \theta_k, \theta_c) = -\sum_{t=1}^{n} \log \big(\sum_{i=1}^{k} p(c_t = i|x_t; \theta_c) \exp(-d(x_t, f_i(x_t; \theta_i))) \big)$$

The final (hard) clustering is:

$$\hat{c}_t = \arg\max_{i=1}^{k} p(c_t = i|x_t; \theta_c), \qquad t = 1, \ldots, n.$$

3 EXPERIMENTS

In this section we evaluate the clustering results of our approach. We carried out experiments on different datasets and compared the proposed method to state-of-the-art standard and k-means related deep clustering algorithms.

3.1 DATASETS

The datasets used in the experiments are standard clustering benchmark collections. We considered both image and text datasets to demonstrate the general applicability of our approach. Image datasets consist of MNIST (70,000 images, 28×28 pixels, 10 classes) which contains hand-written digit images. We reshaped the images to one dimensional vectors and normalized the pixel intensity levels (between 0 and 1). The text collections we considered are the 20 Newsgroups dataset (hereafter dubbed, 20NEWS) and the RCV1-v2 dataset (hereafter dubbed, RCV1) (Lewis et al. (2004)). For 20NEWS, the entire dataset comprising 18,846 documents labeled into 20 different news-groups was used. For the RCV1, similar to (Yang et al. (2017)) we used a subset of the database containing 365,968 documents, each of which pertains to only one of 20 topics. Because of the text datasets sparsity, and as proposed in (Xie et al. (2016)) and (Yang et al. (2017)), we selected the 2000 words with the highest tf-idf values to represent each document.

3.2 EVALUATION MEASURES

The clustering performance of the evaluated methods is evaluated with respect to the following three standard measures: normalized mutual information (NMI) (Cai et al. (2011)), clustering accuracy (ACC) (Cai et al. (2011)), and adjusted Rand index (ARI) (Yeung & W. L. Ruzzo (2001)). NMI is an information-theoretic measure based on the mutual information of the ground-truth classes and the obtained clusters, normalized using the entropy of each. ACC measures the proportion of data points for which the obtained clusters can be correctly mapped to ground-truth classes, where the matching is based on the Hungarian algorithm (Kuhn (1955)). Finally ARI is a variant the Rand index that is adjusted for the chance grouping of elements. Note that NMI and ACC lie in the range of 0 to 1 with one being the perfect clustering result and zero the worst. ARI is a value between minus one to (plus) one, with 1 being the best clustering performance and -1 the opposite.

3.3 BASELINE METHODS

The proposed DAMIC algorithm is compared with the following methods:

K-means (KM): The classic k-means (Lloyd (1982)).

Deep Autoencoder followed by k-means (DAE+KM): This algorithm is carried out in two steps. First, a DAE is applied. Next, KM is applied to the embedded layer of the DAE. This algorithm is also used as an initialization step for the proposed algorithm.

Deep Clustering Network (DCN): The algorithm performs joint reconstruction and k-means clustering at the same time. The loss comprises penalties on both the reconstruction and the clustering losses (Yang et al. (2017)).

Deep Embedding Clustering (DEC): The algorithm performs joint embedding and clustering in the embedded space. The loss function contains only a clustering loss term (Xie et al. (2016)).

3.4 NETWORK IMPLEMENTATION

The proposed method was implemented with the deep learning toolbox Tensorflow (Abadi et al. (2016)). All datasets were normalized between 0 and 1. All neurons in the proposed architecture except the output layer were using rectified linear unit (ReLU) as the transfer function. The output layer in all DAEs was the sigmoid function, while the clustering network output layer was a softmax layer. Batch normalization (Ioffe & Szegedy (2015)) was utilized to all layers, and the ADAM optimizer (Kingma & Ba (2014)) was used for both the pre-training as well as the training phase. In the pre-training phase, the DAE networks were trained with the binary cross-entropy loss function. We set the number of ephocs for the training phases to be 50 ephocs. Yet, *early stopping* was used to prevent mis-convergence of the loss. The mini-batch size is 256.

It is worth noting that for simplicity and to show the robustness of the proposed method, the architectures of the proposed DAMIC in all the following experiments are with a similar shape, i.e. for each of the DAEs we use 5-layers DNN with the following input size: 1024, 256, k, 256, 1024, ReLUs, respectively, and for the clustering network we used 512, 512, k, ReLUs, respectively, where k is

Table 2: Objective measures of the MNIST database.

Method	DAMIC	DCN	DAE+KM	DEC	KM
NMI	**0.85**	0.81	0.74	0.80	0.50
ARI	**0.77**	0.75	0.67	0.75	0.37
ACC	0.77	0.83	0.80	**0.84**	0.53

Table 3: Objective measures of the 20NEWS database.

Method	DAMIC	DCN	DAE+KM	KM
NMI	**0.56**	0.48	0.47	0.41
ARI	**0.42**	0.34	0.28	0.15
ACC	**0.57**	0.44	0.42	0.30

the number of clusters. We emphasize that there was no need for hyperparameter tuning for the experiments on the different datasets.

3.5 RESULTS

3.5.1 MNIST

As mentioned before, the MNIST database has 70000 hand written gray-scale digit images. Each image size is 28×28 pixels. Note that we work on the raw data of the dataset (without pre-processing). For simplicity, the architecture of each one of the DAE is identical. Specifically, for the MNIST dataset we used 5-layers network with number of neurons 1024, 256, 10, 256, 1024, respectively. The output layer of each DAE was set to be the sigmoid function. For the clustering network we used simpler network with 3-layers 512, 512, 10 neurons, respectively. The output transfer layer of the clustering network is the softmax function. Table 2 presents the results of the NMI, the ACC and the ARI of the proposed DAMIC method and several standard baselines. It is clear that the DAMIC outperforms the other methods in the NMI and ARI measures. The DEC method get the highest ACC result.

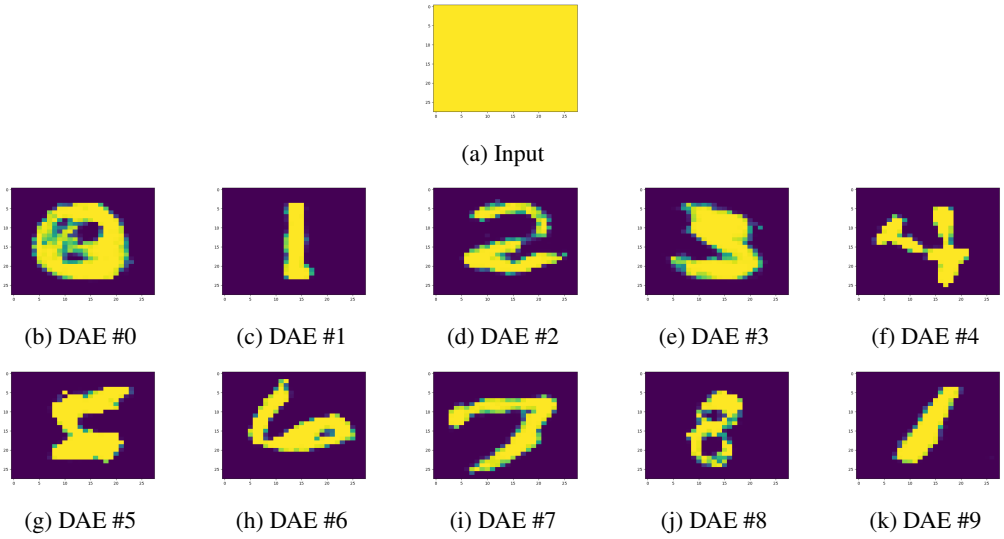
DAE expertise To demonstrate the expertise of each one of the DAE we conducted the following experiment. After the clustering algorithm converged on the MNIST dataset, we synthetically created a new image in which all the pixels are set to be '1' (Fig. 2a). The image reconstruction of all the 10 DAEs is shown in Fig. 2. It is evident that each DAE assumes a different pattern of the input. Specifically, each DAE is responsible for a different digit. The clustering task is unsupervised and we sorted the autoencoders in Fig. 2a) by their corresponding digits from '0' to '9' just for visualization.

Best reconstruction wins To further understand the behavior of the gate we carried out a different test. An image of the digit '4' was fed to the network (Fig. 3a). The outputs of the different DAEs are depicted in Fig. 3. Since each DAE specializes in a different digit, it is expected that the respective DAE will obtain the lowest reconstruction error. This is also reflected by decision of the clustering network $p(c = 4|x; \theta_c) = 0.99$. Note, that the other DAEs reshaped the reconstruction to be close to their digit specialty.

3.5.2 20NEWS

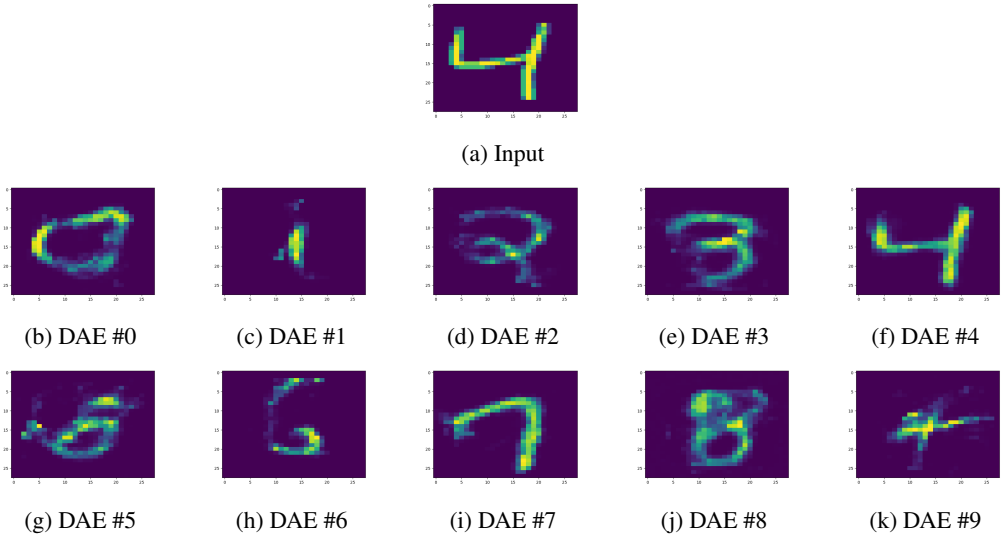
The 20Newsgroup corpus consists of 18,846 documents from 20 news groups. As in Yang et al. (2017) we also use the tf-idf representation of the documents and pick the 2,000 most frequently used words as the features. The architecture used in each one of the DAEs for this experiment also consists of 5-layers with number of neurons 1024, 256, 20, 256, 1024, respectively. The clustering network here consists of 512, 512, 20 neurons.

Table 3 shows the results of the NMI, ARI and ACC measures. It is evident that the proposed clustering method outperforms the compared baselines.



(a) Input (b) DAE #0 (c) DAE #1 (d) DAE #2 (e) DAE #3 (f) DAE #4 (g) DAE #5 (h) DAE #6 (i) DAE #7 (j) DAE #8 (k) DAE #9

Figure 2: The outputs of the different DAEs with a vector of all-ones input.



(a) Input (b) DAE #0 (c) DAE #1 (d) DAE #2 (e) DAE #3 (f) DAE #4 (g) DAE #5 (h) DAE #6 (i) DAE #7 (j) DAE #8 (k) DAE #9

Figure 3: An example of the outputs of the different DAEs with the digit '4' as the input.

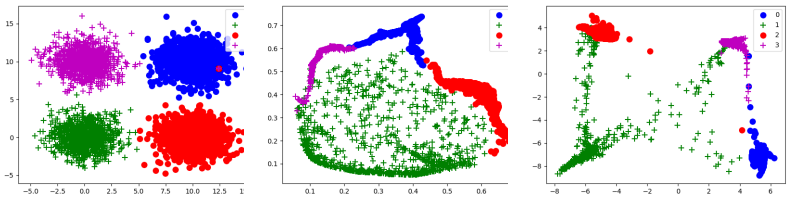
3.5.3 RCV1

The dataset was used in this experiment is a subset of the RCV-1-v2 with 365, 968 documents each containing one of 20 topics. As in Yang et al. (2017) the 2,000 most frequently used words (in the tf-idf form) are used as the features of each documents. In contrast with the previous databases, in the RCV1 dataset, the size of each class in not equal. Therefore, KM-based approaches might be not sufficient in this case. In our architecture we used 1024, 256, 20, 256, 1024 ReLU neurons in all DAEs, respectively, and in the clustering network we used 512, 512, 20 ReLU neurons.

In Table 4 we present the 3 objective measurements for the RCV1 experiment. The proposed method outperforms the competing methods in NMI and ARI measures but obtains lower result in the ACC measure.

Table 4: Objective measures of the RCV1 database.

Method	DAMIC	DCN	DAE+KM	DEC	KM
NMI	**0.62**	0.61	0.59	0.08	0.58
ARI	**0.38**	0.33	0.33	0.01	0.29
ACC	0.43	**0.47**	0.46	0.14	0.47



(a) Input (b) DAE+KM (c) DAMIC

Figure 4: Synthetic dataset with 4 clusters.

3.5.4 SYNTHETIC DATASET

To further illustrate the capabilities of the DAMIC algorithm we generated a synthetic data as in Yang et al. (2017). The 2D latent domain contains 4000 samples from four Gaussian distributed clusters as shown in Fig. 4a. The observed signal is $x_t = (\sigma(\mathbf{W} \cdot v_t))^2 \quad t = 1, \cdots, n$, where σ is the sigmoid function, $\mathbf{W} \in R^{100\times 2}$ and v_t is the t-th point in the latent domain.

We first applied the DAE+KM algorithm for initialization. The architecture of the DAE consists of 4-layers-encoder with 100, 50, 10, 2 neurons respectively. The decoder is a mirrored version of the forward network. Fig. 4b depicts the 2D embedded space of the DAE. It is not sufficiently separated in the embedded space. The proposed DAMIC algorithm was then applied. The architecture of each autoencoder consists of 5-layers 1024, 256, 4, 256, 1024 neurons as in the previous experiments. The clustering network is also similar with 512, 512, 2 neurons, respectively. Fig. 4c depicts the 2D embedded space of the clustering network $h(x_t)$. It is easy to see that the embedded space is much more separable.

Table 5 summarizes the results of the k-means, the DAE+KM and the DAMIC algorithms on the synthetic generated data. It is easy to verify that the DAMIC algorithm outperforms the two competing algorithms in both NMI and ARI measures.

Table 5: Objective measures of the synthetic database.

Method	DAMIC	DAE+KM	KM
NMI	**0.90**	0.83	0.80
ARI	**0.92**	0.84	0.81

4 CONCLUSION

In this study we presented a clustering technique which leverages the strength of deep neural network. Our technique has two major properties: first, unlike most previous methods, the clusters are represented by an autoencoder network instead of a single centroid vector in the embedded space. This enables a much richer representation of each cluster. Second, The algorithm does not cause a data collapsing problem. Hence, there is no need for regularization terms that have to be tuned for each dataset separately. Experiments on a variety of real datasets showed the improved performance of the proposed algorithm.

REFERENCES

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

E. Aljalbout, V. Golkov, Y. Siddiqui, and D. Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

D. Cai, X. He, and J. Han. Locally consistent concept factorization for document clustering. *IEEE Transactions on Knowledge and Data Engineering*, 23(6):902–913, 2011.

M. Caron, P. Bojanowski, A. Joulin, and M. Douze. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision (ECCV)*, 2018.

D. Eigen, M. Ranzato, and I. Sutskever. Learning factored representations in a deep mixture of experts. In *International Conference on Learning Representations (ICLR), Workshop*, 2014.

C. C. Hsu and C. W. Lin. CNN-based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Transactions on Multimedia*, 20(2):421–429, 2018.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

R. Jacobs, S. Nowlan M. Jordan, and G. Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1): 79–87, 1991.

M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2): 181–214, 1994.

D. Kingma and J. Ba. ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, pp. 1–1, 07 2018.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

J. Xie, R. Girshick, and A. Farhad. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning (ICML)*, 2016.

B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong. Towards K-means-friendly spaces: Simultaneous deep learning and clustering. In *International Conference on Machine Learning (ICML)*, 2017.

K. Y. Yeung and W. L W. L. Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 19(9):763–774, 2001.