Deep Clustering based on a Mixture of Autoencoders

Shlomo E. Chazan, Sharon Gannot and Jacob Goldberger

Bar-Ilan University

Ramal-Gan, 5290002, Israel

{Shlomi.Chazan, Sharon.Gannot, Jacob.Goldberger}@biu.ac.il

Abstract

In this paper we propose a Deep Autoencoder Mixture Clustering (DAMIC) algorithm 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 k-means. Clustering high-dimensional datasets is, however, hard since the 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 networks in supervised learning, there have been many attempts to apply unsupervised deep learning approaches to clustering. Most methods are focused on clustering over the low-dimensional feature space of an autoencoder [29][8][13][28], a variational autoencoder [16] [7] or a Generative adversarial Network (GAN) [11][25][6]. Recent good overviews of deep clustering methods can be found in [2] and [22].

Using deep neural networks, nonlinear mappings that can transform the data into more clustering-friendly representations, can be learned. A deep version of k-means is based on learning a nonlinear 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. For this reason, the objective function of most deep clustering algorithms is composed of a clustering term computed in the embedded space and a regularization term in the form of a reconstruction error to avoid data collapse. Deep Embedded Clustering (DEC) [27] is first pre-trained using an autoencoder reconstruction loss and then optimizes cluster centroids in the embedded space through a Kullback-Leibeler divergence loss. The Deep Clustering Network (DCN) [28] 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. However, 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, because strict cluster assignments were used during the training (instead of probabilities such as in DEC) the method requires an alternation process between network training and cluster updates.

In this paper we propose an algorithm to perform deep clustering within the mixture-of-experts framework [15]. 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 deep clustering approach is different from previous algorithms in three main aspects:

- It does not suffer from the clustering collapsing problem, since the trivial solution is not the global optimum

of the clustering learning objective function.

- This implies that in the proposed method, unlike other methods, 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 supposedly unavailable in the clustering process.

- Another major difference between the proposed method and previously proposed approaches is the learning method of the embedded latent space, where the actual clustering takes 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 algorithms no decoding is applied to the clustering in the 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 using standard real datasets including document and image corpora. The results show a visible improvement from previous methods for all the datasets. The contribution of this paper is thus twofold: (i) it presents a novel deep learning clustering method that unlike deep variants of k-means does not require a tuned regularization term to avoid clustering collapse to a single point; and (ii) it demonstrates improved 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:

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

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

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 specialized in reconstructing instances of that cluster. If the dataset is properly clustered, we expect all the points assigned to be same cluster to be similar. Hence, the task of a cluster-specialized autoencoder should be relatively easy compared to using a single autoencoder for the entire data. We thus expect that good clustering should result 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 cluster i that is tuned to 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.

To find the network parameters we jointly train the clustering network and the deep autoencoders. The clustering is thus computed by minimizing the following loss function:

$$L(\theta_1, \ldots, \theta_k, \theta_c) \quad (3)$$

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

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 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 autoencoders for reconstruction. Hence, there is no need to add regularization terms to the loss function (that might influence the clustering accuracy) to prevent data collapse. Specifically, there is no need to add a decoder to the embedded space where the clustering is actually performed to prevent data collapse.

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

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

such that

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

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

In recent years, network pre-training has been largely rendered obsolete for supervised tasks, given availability of large labeled training datasets. However, for hard optimization problems that unsupervised clustering tasks cannot handle (like the one presented in (1)), initialization is still crucial. To initialize the parameters of the network, we first train a single autoencoder and use the layer-wise pre-training method, as described in [3], for training autoencoders. After training the autoencoder, we carry out a k-means clustering on the output of the bottleneck layer to obtain the initial clustering values. The k-means assigns a label to each data point. Note, that in the pre-training procedure a *single* autoencoder is trained on the entire database. We use these labels as 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 parameters have been 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 Autoencoder MIxture Clustering (DAMIC). The architecture of the network trained by the DAMIC algorithm and the final clustering procedure are depicted in the left and right panels of Fig. 1, respectively. 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. In so doing, 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 also be viewed as an instance of the mixture-of-experts (MoE) model introduced in [15] and

[17]. 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 architectural component of a neural network [9, 24]. MoE has been primarily 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 were each autoencoder's expertise is t reconstruct a sample from the associated cluster. Our clustering cost function (3) follows the training strategy proposed in [15], which prefers an error function that encourages expert specialization instead of cooperation.

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

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

Network components:

- A network that computes a soft clustering of the data point:

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

- A set of autoencoders (one for each cluster):

$$x \to \hat{x}_i = f_i(x; \theta_i), \quad i = 1, \ldots, k$$

Pre-training:

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

Training: 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:

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

We note that after the training process is finished, there is another way to extract the clustering from the trained network. Given a data point x_t, we can ignore the clustering DNN and assign each point to the cluster whose reconstruc-
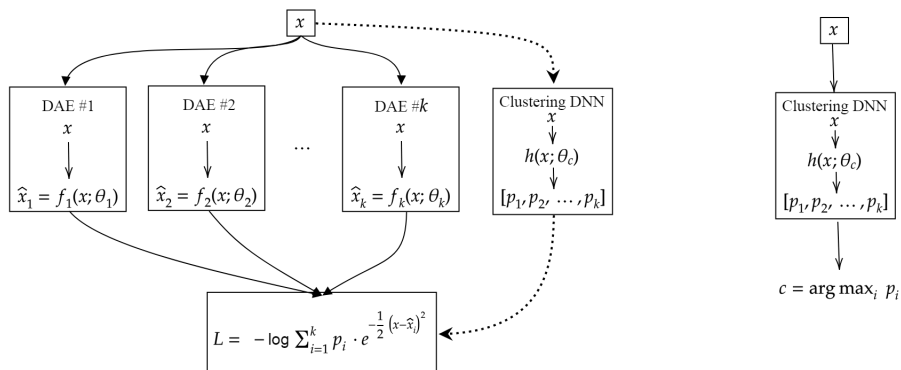
Figure 1: A block diagram of the proposed mixture of deep auto-encoders for clustering. The training procedure is described on the left side, and the final clustering on the right side.

tion error is minimal:

$$c_t = \arg \min_{i=1}^{k} d(x_t, f_i(x_t; \theta_i))). \qquad (6)$$

We found that the performance of this clustering decision is very close to the clustering strategy we proposed (2). Moreover, in almost all cases the hard classification decision of the clustering network (1) coincides with the cluster whose reconstruction error is minimal, i.e.,

$$\arg \max_{i=1}^{k} p(c_t = i | x_t; \theta_c) = \arg \min_{i=1}^{k} d(x_t, f_i(x_t; \theta_i))).$$

We can thus consider a variant of our clustering algorithm that completely avoids the clustering network. Instead, the training goal is to directly minimize the reconstruction error of the most suitable autoencoder using the following cost function:

$$L(\theta_1, \ldots, \theta_k) = \sum_{t=1}^{n} \min_{i=1}^{k} d(x_t, f_i(x_t; \theta_i))). \qquad (7)$$

This cost function is very similar to the cost function of the k-means algorithm. The only difference is that the constant centroid is replaced here by the autoencoder bottleneck where the given point is the input. However, there are two drawbacks of using this alternative and simpler cost function. First, in our algorithm, in addition to the data clustering, we also obtain a nonlinear data embedding $x \rightarrow h(x)$ that can be used to visualize the clustering in a clustering friendly space. The second issue is that we found empirically that without the clustering network even if we use the pre-processing procedure we described above, we are more vulnerable to clustering collapsing issues, in the sense that at the end of the training procedure some of the autoencoders are not used by any data point. This provides another motivation for the proposed architecture that is based on an explicit clustering network.

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 the state-of-the-art standard and k-means related deep clustering algorithms.

3.1. Datasets

We used both synthetic dataset as well as real datasets. The synthetic dataset will be described in Sec. 4.1. The real 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.

The image datasets consisted of MNIST (70,000 images, 28×28 pixels, 10 classes) which contain hand-written digit images. We reshaped the images to one dimensional vectors and normalized the pixel intensity levels (between 0 and 1). The Fashion dataset [26], which is consisting of 70,000 examples with similar dimensions as of the MNIST dataset, was also used. This dataset is divided into 10 fashion classes.

The text collections we considered are the 20 Newsgroups dataset (hereafter 20NEWS) and the RCV1-v2 dataset (hereafter RCV1) [20]. For 20NEWS, the entire dataset comprising 18,846 documents labeled into 20 different news-groups was used. For the RCV1, similar to [28] we used a subset of the database containing 365,968 documents, each of which pertains to only one of 20 topics. Because of text dataset sparsity, and as proposed in [27] and [28], we selected the 2000 words with the highest tf-idf values to represent each document.

3.2. Evaluation measures

The clustering performance of the methods was evaluated with respect to the following three standard measures:

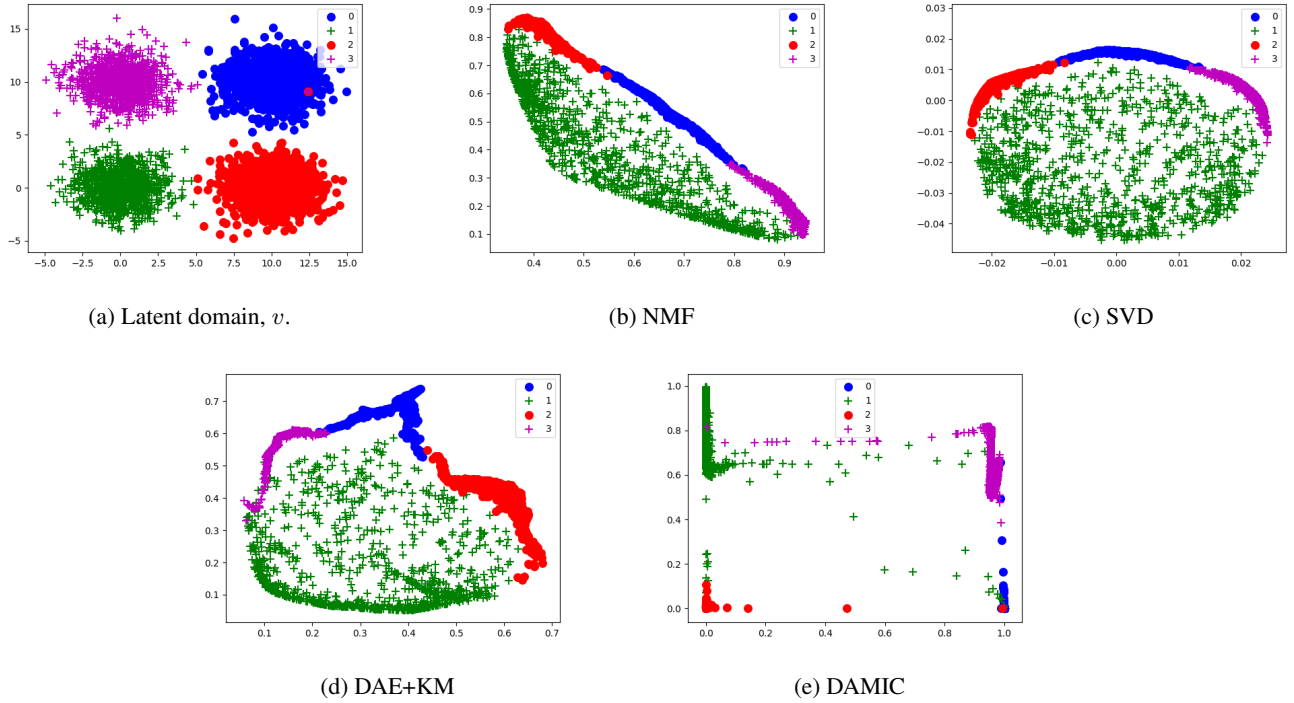(a) Latent domain, v.   (b) NMF   (c) SVD

(d) DAE+KM   (e) DAMIC

Figure 2: Synthetic dataset with 4 clusters. Each true cluster label has a different color. The observable data is generated from the Gaussian distributed clusters in the first figure, through (8). The 2D representations of the observed data are shown by the NMF, SVD, DAE+KM and the proposed DAMIC methods.

normalized mutual information (NMI) [4], adjusted Rand index (ARI) [30], and clustering accuracy (ACC) [4]. 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 [19]. Finally, ARI is a variant of 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 where one is the perfect clustering result and zero the worst. ARI is a value between minus one to (plus) one, where one is the best clustering performance and minus one the worst.

3.3. Baseline methods

The proposed DAMIC algorithm was compared to the following methods:

K-means (KM): The classic k-means [21].

Spectral Clustering (SC): The classic SC algorithm [23].

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 [28].

Deep Embedding Clustering (DEC): The algorithm performs joint embedding and clustering in the embedded space. The loss function only contains a clustering loss term [27].

3.4. Network implementation

The proposed method was implemented with the deep learning toolbox Tensorflow [1]. All datasets were normalized between 0 and 1. All neurons in the proposed architecture except the output layer used exponential linear unit (elu) as the transfer function. The output layer in all DAEs was the sigmoid function, and the clustering network output layer was a softmax layer. Batch normalization [14] was utilized on all layers, and the ADAM optimizer [18] 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 epochs for the training phases to be 50. However, *early stopping* was used to prevent mis-convergence of the loss. The mini-batch size was 256.

Note that for simplicity and to show the robustness of the proposed method, the architectures of the proposed DAMIC in all the following experiments had a similar shape; i.e., for each of the DAEs we used a 5-layer DNN with the following input size: 1024, 256, k, 256, 1024, elu neurons, respectively, and for the clustering network we used 512, 512, k, elu neurons, respectively, where k is the number of clusters. There was no need for hyperparameter tuning for the experiments on the different datasets.

4. Results

4.1. Synthetic dataset

To illustrate the capabilities of the DAMIC algorithm we generated synthetic data as in [28]. The 2D latent domain contained 4000 samples from four Gaussian distributed clusters as shown in Fig. 2a. The observed signal is

$$x_t = (\sigma(\mathbf{W} \cdot v_t))^2 \quad t = 1, \cdots, n \quad (8)$$

where σ is the sigmoid function, $\mathbf{W} \in \mathbb{R}^{100 \times 2}$ 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 consisted of a 4-layer encoder with 100, 50, 10, 2 neurons respectively. The decoder was a mirrored version of the forward network. Fig. 2b, 2c and 2d depict the 2D representations of (8) by NMF, the SVD and the DAE+KM methods, respectively. It is clear that it is not sufficiently separated.

The proposed DAMIC algorithm was then applied. The architecture of each autoencoder consisted of 5-layers of 1024, 256, 4, 256, 1024 neurons as described in the previous section. The clustering network was also similar, with 512, 512, 2 neurons, respectively. Fig. 2e 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 2 summarizes the results of the k-means, the DAE+KM, the SC 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 2: Objective measures for the synthetic database.

Method	DAMIC	DAE+KM	SC	KM
NMI	**0.94**	0.83	0.82	0.80
ARI	**0.96**	0.84	0.83	0.81

Table 3: Objective measures of the MNIST database.

Method	DAMIC	DCN	DAE+KM	DEC	KM
NMI	**0.87**	0.81	0.74	0.80	0.50
ARI	**0.81**	0.75	0.67	0.75	0.37
ACC	**0.89**	0.83	0.80	0.84	0.53

Table 4: Objective measures of the Fashion database.

Method	DAMIC	DCN	DAE+KM	DEC	KM
NMI	**0.65**	0.55	0.60	0.54	0.51
ARI	**0.49**	0.42	0.45	0.40	0.37
ACC	**0.60**	0.50	0.57	0.51	0.47

Table 5: Objective measures of the 20NEWS database.

Method	DAMIC	DCN	DAE+KM	SC	KM
NMI	**0.57**	0.48	0.47	0.40	0.41
ARI	**0.42**	0.34	0.28	0.17	0.15
ACC	**0.56**	0.44	0.42	0.34	0.30

Table 6: 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	**0.41**	0.33	0.33	0.01	0.29
ACC	**0.47**	**0.47**	0.46	0.14	0.47

4.2. MNIST

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

DAE expertise To test 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 were set to be '1' (Fig. 3a). The image reconstruction of all the 10 DAEs is shown in Fig. 3. It is evident that each DAE assumes a different pattern of input. Specifically, each DAE is responsible for a different digit. The clustering task
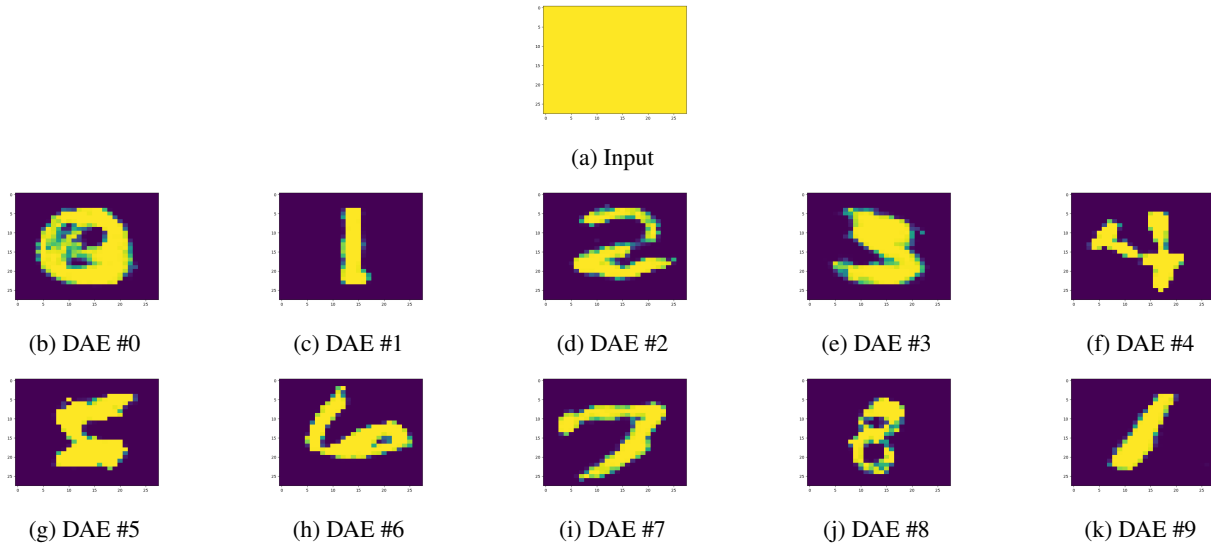
(a) Input

(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: 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
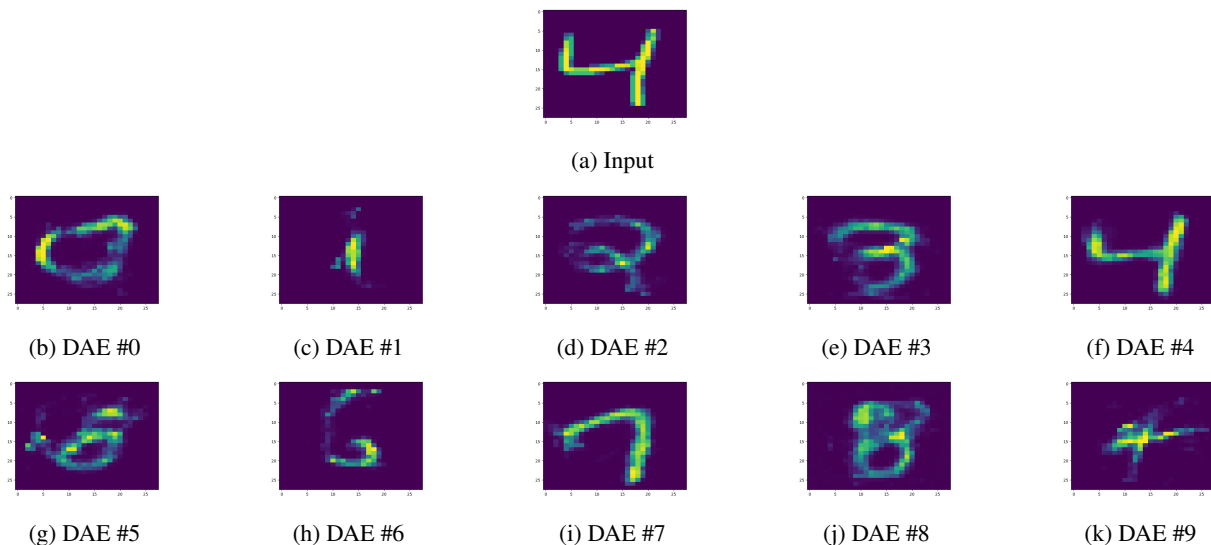

(j) DAE #8

(k) DAE #9

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

was unsupervised and we sorted the autoencoders in Fig. 3a) by their corresponding digits from '0' to '9' merely for purpose of 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. 4a). The outputs of the different DAEs are depicted in Fig. 4. Since each DAE specializes in a different digit, it was expected that the respective DAE would have the lowest reconstruction error. This was also reflected in 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 expertise.

4.3. Fashion

The Fashion dataset shares the same dimensions and structure as the MNIST dataset. The only difference is the content of the images, which are now one of ten fashion items. Therefore, the same processing described in the former section was carried out on this dataset.

Table 4 describes the results on the Fashion dataset. It is clear that the proposed DAMIC algorithm outperforms the compared algorithms.

We note that the best reported clustering results on the MNIST data achieved by the VaDE algorithm [16] that

Table 7: Ablation study on the MNIST database.

Method	DAMIC	Pre-training only	Joint-training only	KM
NMI	**0.87**	0.74	0.71	0.50
ARI	**0.81**	0.67	0.53	0.37
ACC	**0.89**	0.80	0.60	0.53

applies variational autoencoder modeling assuming mixture of Gaussians distribution of the latent random variable. We compared our method to state-of-the-art deep k-means-based algorithms using the same network architecture and parameter initialization and showed improvement in the performance. VaDE algorithm belongs to a different family of algorithms with different network architecture and parameter initialization strategies. Hence, a direct performance comparison is difficult since it is heavily dependent on the implementations. It is worth noting that in the Fashion dataset, our results even outperforms the VaDE and the DEC with data augmentation (DEC-DA) algorithms results [10] in this dataset.

4.4. 20NEWS

The 20Newsgroup corpus consists of 18,846 documents from 20 news groups. As in [28] we also used the tf-idf representation of the documents and picked the 2,000 most frequently used words as the features. The architecture used in each one of the DAEs for this experiment also consisted of a 5-layer DNN with 1024, 256, 20, 256, 1024 neurons, respectively. The clustering network here consisted of 512, 512, 20 neurons.

Table 5 shows the results of the NMI, ARI and ACC measures. It is clear that the proposed clustering method outperformed the competing baseline algorithms.

4.5. RCV1

The dataset used in this experiment is a subset of the RCV-1-v2 with 365, 968 documents, each containing one of 20 topics. As in [28] the 2,000 most frequently used words (in the tf-idf form) are used as the features for each documents. In contrast to the previous databases, in the RCV1 dataset, the size of each class is not equal. Therefore, KM-based approaches might not be sufficient in this case. In our architecture we used 1024, 256, 20, 256, 1024 elu neurons in all DAEs, respectively, and in the clustering network we used 512, 512, 20 elu neurons.

Table 6 presents the 3 objective measurements for the RCV1 experiment. The proposed method outperformed the competing methods in NMI and ARI measures and had the same ACC score as of the DCN.

4.6. Ablation study

The DAMIC algorithm comprises of two steps, the initialization step, which is based on a deep autoencoder fol-lowed by a k-means clustering (DAE+KM), and a joint training of the gate and the clusters' autoencoders. We next explore the necessity of each part in the algorithm. For that, we compared the DAMIC with two variants of the proposed algorithm. The first one is based only on the initialization phase (without the joint the training), and the second is based only on the joint training phase with random initialization.

Table 7 describes the results of our ablation study. The first conclusion is that the joint training with random initialization improves the k-means results. This result confirm the first contribution of this paper that even with random initialization, the algorithm does not suffer from the clustering collapsing problem. It is also evident that the pre-training phase is important for parameter initialization but it is not enough. The proposed algorithm which employs both, initialization step and joint training outperforms each of the variants separately.

5. 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 strong performance of the proposed algorithm over the other methods.

References

[1] 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, pages 265–283, 2016.

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

[3] 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.

[4] 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.

[5] 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.

[6] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Adv. Neural Inf. Process. Syst.*, 2016.

[7] N. Dilokthanakul1, P. Mediano, M. Garnelo, M. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoder. *arXiv preprint arXiv:1611.02648*, 2016.

[8] K. G. Dizaji, A. Herandi, C. Deng, W. Cai, and H. Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5747–5756. IEEE, 2017.

[9] 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.

[10] X. Guo, E. Zhu, X. Liu, and J. Yin. Deep embedded clustering with data augmentation. In *Asian Conference on Machine Learning*, pages 550–565, 2018.

[11] W. Harchaoui, P. A. Mattei, and C. Bouveyron. Deep adversarial gaussian mixture autoencoder for clustering. In *ICLR*, 2017.

[12] 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.

[13] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, and M. Sugiyama. Learning discrete representations via information maximizing self augmented training. *arXiv preprint arXiv:1702.08720*, 2017.

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

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

[16] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

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

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

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

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

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

[22] 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*, pages 1–1, 07 2018.

[23] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[24] 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.

[25] J. T. Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv: 1511.06390*, 2015.

[26] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

[28] 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.

[29] J. Yang, D. Parikh, and D. 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.

[30] 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.