Neurocomputing 433 (2021) 96–107

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Deep embedding clustering based on contractive autoencoder

Bassoma Diallo [a], Jie Hu [a,*], Tianrui Li [a], Ghufran Ahmad Khan [a], Xinyan Liang [b], Yimiao Zhao [a]

[a] Institute of Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China
[b] Institute of Big Data Science and Industry, Shanxi University, Taiyuan 030006, Shanxi, China

ARTICLE INFO

Article history:
Received 21 April 2020
Revised 5 September 2020
Accepted 19 December 2020
Available online 5 January 2021
Communicated by Steven Hoi

Keywords:
Document clustering
Representation learning
Contractive autoencoder
Deep learning
Deep embedding clustering

ABSTRACT

Clustering large and high-dimensional document data has got a great interest. However, current clustering algorithms lack efficient representation learning. Implementing deep learning techniques in document clustering can strengthen the learning processes. In this work, we simultaneously disentangle the problem of learned representation by preserving important information from the initial data while pushing the original samples and their augmentations together in one hand. Furthermore, we handle the cluster locality preservation issue by pushing neighboring data points together. To that end, we first introduce Contractive Autoencoders. Then we propose a deep embedding clustering framework based on contractive autoencoder (DECCA) to learn document representations. Furthermore, to grasp relevant document or word features, we append the Frobenius norm as penalty term to the conventional autoencoder framework, which helps the autoencoder to perform better. In this way, the contractive autoencoders apprehend the local manifold structure of the input data and compete with the representations learned by existing methods. Finally, we confirm the supremacy of our proposed algorithm over the state-of-the-art results on six real-world images and text datasets.

1. Introduction

Clustering is a challenging but crucial task in fields such as machine learning and pattern recognition [1]. Nevertheless, clustering is still ineffectively and inefficiently deal with large-scale, sophisticated-semantics, high-dimensional data [2]. The dominant disadvantage comes from the two following cases. The first is to apprehend the data representation in a low-dimension, and the latter is to assure the effectiveness of the clustering. Clustering is widely applied on image and text document datasets.

Document clustering (DC) is employed to assemble the topic digital library. DC has performed a meaningful performance in many areas like data mining, information retrieving and filtering, etc [3]. Clustering a document set is equivalent to put these documents into designated packages so that they only wrap up in a similar pack if and only if they link together [4]. That allows us to search documents by looking through a relatively slight collection of boxes (clusters) instead of going into a much bigger dataset of documents directly. Traditional document clustering approaches utilize a vector space model to represent a document. They are generally sensitive to the input affinity matrix since they subdivide the data depend on a fixed data graph. Clustering large text document datasets is a challenging assignment due to the data imbalanced distribution. The accuracy of the current machine, and deep learning clustering algorithms decreases when the dimension of document data increases as they are initially tailored for low dimension data [5]. It is well known that two problem-dependent factors, which are: i) the chosen similarity metric and ii) the data representation, bias the clustering quality [6]. Learning good representations from raw data points by Deep Learning (DL) presents a meaningful role in clustering tasks [7]. Researchers categorized DL algorithms toward two broad stage works. The former applies clustering after learning a representation [7–9]. The latter approach jointly optimizes the feature learning and clustering abilities [10–14]. Critical analysis of existing document clustering algorithms reveals that they either do not well take advantage of deep neural networks or are sensitive to the input data [15,16]. As a consequence, the preservation of the local structure of data-generating distribution during the embedded feature learning phase is disrupted considerably.

DL has been widely employed in the field of big data in the last decade, particularly for temporally and spatially structured data such as images and videos. But it has been less successful in dealing with text data [17]. One crucial step in the fields of text mining, natural language processing, and information retrieval is the

* Corresponding author.
E-mail addresses: sanediallo2003@yahoo.fr (B. Diallo), jiehu@swjtu.edu.cn (J. Hu), trli@swjtu.edu.cn (T. Li), ghufraan.alig@gmail.com (G.A. Khan), liangxinyan48@163.com (X. Liang), Zymiao@my.swjtu.edu.cn (Y. Zhao).

https://doi.org/10.1016/j.neucom.2020.12.094
0925-2312/

document representation [18]. A highly complex latent structured data is still challenging the effectiveness of existing DL clustering models. The current DL methods suffer from an unrealistic assumption of words independence or exchangeability [19]. Disregarding the correlations or order of words in a document misleads DL approaches during the document representation [4]. They do not retain the relevant properties of the text data during the transformation, which results in low accuracy of large-scale text document data. Besides DL techniques, large numbers of representation learning schemes have been recommended in [20]. Among these methods, someone can name the Non-negative Matrix Factorization (NMF) [21–24]. The main shortcoming of these methods is that they could not represent the data as a non-linear latent subspace. To sum up, DL has drawn much attention because its highly non-linear architecture can help to learn dominant feature representations [25]. While these methods preserve the image data information successfully [26], they do not have an explicit preference for maintaining text document local structure [27].

The autoencoder (AE) and its variations have been widely applied in unsupervised learning tasks. They are convenient for learning node representations for graphs. Contractive autoencoder proposes a regularization term that makes a mapping between input space and feature space contracting at training samples actively [28]. It uses contraction ratio and Jacobian matrix constraint in the lost function to suppress the agitation of the input in all directions, which yields the robust extracted features. In this study, inspired by the well-recognized power of the autoencoder in learning intrinsic content representations, we propose to utilize Contractive Autoencoders (CtAE) in the task of clustering. The objective of this study is to introduce the CtAE architecture, which can handle both image and text data. We provide an accurate activation function to learn and model other complicated variety of data, such as text data. In this work, we replace the denoising autoencoder with the CtAE. The reason for this choice is as follows: Denoising autoencoders prepare the reconstruction function (i.e., decoder) to thwart small but finite-sized disorders of the input, while CtAE makes the feature extraction function (i.e., encoder) defy minuscule disturbances of the input. It is generally preferred for data representations to be locally invariant, i.e., remain constant under local disruptions on data points [30]. To that end, we first penalize the reconstruction loss with the Frobenius norm (3) to reinforce the stability of the representation of the data which is retrieved for a training input and then, with the self-augmentation (9) to make the representations closer to the augmentations.

Unlike the Deep Embedded Clustering (DEC) framework [10], we add the Batch Normalization layer to reduce the training time. We jointly normalize the input and the hidden layers by fine-tuning the mean and scaling of the activation using Batch Normalization. This normalization allows the network to use the most advanced learning rate and avoid the vanishing or gradient problems likewise to alleviate over-fitting. Feature extractors like Convolutional Neural Networks (CNN), Autoencoders (AE), Deep Belief Network (DBN) are available. Recently, researchers use neural networks to generate infinite ensemble partitions and to fuse them into a consensus partition to obtain the final clustering (Multilayer Perceptron MLP). We use AE over spectral clustering (SC) not only because its computational complexity is much lower than SC, but it can also be linear to the number of nodes in a sparse graph. Notice that the computational complexity of SC is super quadratic due to the eigenvalue decomposition. AE commonly drives to a local optimum by virtue of the back-propagation algorithm, which is employed in [8].

Thus, we summarize the contributions of our proposed method as follows:

- We recommend a Deep Embedding Clustering algorithm based on Contractive Autoencoder (DECCA) to automatically cluster documents by taking advantage of the CtAE. And the resulting optimization issue can be efficiently determined by mini-batch stochastic gradient descent and back-propagation.
- We propose an alternative approach to avoid uninterested clusters by adding an accurate term in the loss function that penalizes the cluster. Our method extracts only features that reflect variations observed in the training set.
- The model tackles the high-dimensional features issue using a similarity metric, which considers both the direction and the magnitude of document vectors.
- We derive a new loss function, which is the combination of the contractive autoencoder reconstruction objective function and the Jacobian of the hidden layer.

The groundwork of this study is conceived as follows: Section 2 recovers the related work. The proposed deep embedding model for DC is presented in Section 3. We conduct experimental studies in Section 4. Section 5 contains the present paper conclusion and the future work plan.

2. Related work

This section presents a review of recent literature on deep clustering as well as text clustering. Additionally, we introduce the contractive autoencoder, which is the foundation of our proposed method.

2.1. Deep clustering

Deep clustering is a process of learning feature representations for clustering tasks using deep neural networks. Deep subspace clustering [31], deep spectral clustering [32], as well as deep embedding based on autoencoder and self-expression layer has become famous for clustering method. Clustering methods are always subject to the uncertainty of similarities between samples. To decrease the influence of the uncertainty, Yu and Wang [1] developed deep clustering with Adaptive Siamese Loss and Reconstruction Loss. The first focuses on mapping the samples from the data space to the same representations or the orthogonal representations, and the second provides a priori knowledge to stabilize the clustering process. Huang et al. [31] came up with a deep subspace clustering framework that simultaneously withdraws features via the embedding neural network and executes subspace learning. The structure combines diverse components such as an autoencoder, a self-expression layer, and a supervised competitive feature learning. The proposed model profoundly captures characteristic features then guides the clustering task by using relative entropy to minimize probabilistic cluster assignments. SA-Net [32] overcomes the weaknesses of the spectral analysis procedure and extends it into a deep learning framework with multiple layers. Law et al. [6] proposed a metric learning framework that enhances an embedding function.

Tian et al. [8] proposed a graph-encoder framework based on sparse autoencoder. This model feeds a normalized graph similarity matrix into a deep neural network (DNN) built on a sparse autoencoder. At the pre-training process stage, the algorithm looks for the best non-linear graph representations that can be more similar to the input. Once the desired sparsity properties are obtained, it tacks a mass of sparse autoencoder layers. At the final stage, runs k-means on the sparse encoding output of the last layer to achieve the clustering results.

The DEC algorithm [10] deploys DNN to determine the feature representations and cluster assignment simultaneously. The model

discovers a mapping from the data space to a lower-dimensional feature space and iteratively optimizes only a clustering objective. This algorithm establishes an adequate objective in a self-learning fashion. The clustering loss updates the cluster centers and the network transformation parameters concurrently. The aim is to minimize a KL divergence cost between the soft assignments and the auxiliary distribution. However, it ignores the correlation of word in the documents, which may lead to the corruption of feature space. Further, diverse versions come out in [11,12,33].

Ren et al. [34] proposed a new scenario of Semi-supervised Deep Embedded Clustering (SDEC) to reduce the drawbacks of DEC. Precisely, SDEC learns feature representations that encourage the clustering tasks and performs clustering assignments together by incorporating the pairwise constraints in the feature learning process. Guo et al. proposed the Deep Clustering with Convolutional Autoencoder (DCEC) [11] algorithm, which upgrade the DEC by incorporating convolutional layers to preserve the local structure of data generating distribution. After the initialization step, it uses the denoising autoencoder to train the model. Then, it adds an under-complete autoencoder to the DEC framework [10].The proposed DCEC algorithm exploits the advantages of Convolutional Autoencoder (CAE) and local structure preservation. However, CAE is beneficial only to learn features from image data. Further, they proposed the Improved Deep Embedded Clustering (IDEC) algorithm, which can conserve the data structure [12]. In this model, they manipulate the feature space to discard the data points using a clustering loss. Later, they employ an under-complete autoencoder not only to tighten the manipulation but also to keep the local structure of the generating data distribution. By fusing the clustering loss and autoencoder's reconstruction loss, IDEC can mutually optimize cluster labels assignment and cluster the learned features without destroying the local structure. Afterward, they extend the Deep Embedded Clustering with Data Augmentation (DEC-DA) and instantiated five specific algorithms [33]. In this work, they firstly train an autoencoder with the augmented data to build up the basic feature space. Then, they restrain the embedded features with a clustering loss to further learn clustering-oriented features. Jiang et al. [7] proposed the Variational Deep Embedding (VaDE), a novel unsupervised generative clustering method. VaDE embeds the probabilistic clustering problems into a Variational Auto-Encoder framework. Ren et al. [36] proposed a deep density-based image clustering framework in a two-level to undertake the issues of pre-defining the number of clusters and learning the initial cluster centers. At the first level, they trained a deep CAE to withdraw the low-dimensional feature representations from the high-dimensional image data. Further, they applied t-SNE to reduce the data to the 2-dimensional space favoring density-based clustering algorithms. In the later level, to systematically identify the appropriate number of clusters with inconsistent patterns for the 2-dimensional embedded data, they proposed an innovative clustering approach depend on the density. To that end, several regional groups are achieved to grab the local formats of clusters and then are consolidated through their density relationship to found the eventual clustering result. However, these algorithms only focus on feature extraction by minimizing reconstruction error rather than a specific clustering task, leading to unsatisfactory performance and cannot deal with document data accurately, and they perform better with only image data [43].

2.2. Text clustering

The text clustering technique is a convenient mechanism to deal with a tremendous amount of text documents by grouping these documents into meaningful groups. The document size turns down the textual DC approach efficiency. It is well known that text documents contain sparse, noisy, inappropriate, and redundant features, which affect the effectiveness of the text clustering technique. Abualigaha et al. [41] proposed an innovative feature selection method using the particle swarm optimization algorithm to solve the problem, as mentioned earlier, by generating a new subspace of instructive text features. This method is divided into three stages. In the first stage, the pre-processing steps are applied to represent the text documents in a numerical style. In the second stage, they used the PSO algorithm to remove enigmatic features at the level of the document.

Huang et al. [44] proposed a new co-clustering scheme alongside the adaptive local structure learning. The nominated unified learning structure completes intrinsic structure learning and 3-factor factorization simultaneously. The underlying structure is flexibly determined from the results of tri-factorization, and the factors are reformulated to conserve the refined local structures of the textual data. Furthermore, the framework analyzes together the adaptive local structure both of the data space and the feature space while taking into consideration the match between words and documents. Lu et al. [40] proposed a semi-supervised concept factorization by merging the pairwise constraints and penalty terms into concept factorization (CF). This technique assures that the data points associated with a cluster in the initial space are still uniform in the transformed space. Ailem et al. [42] suggested a novel generative mixture model for co-clustering sparse high dimensional data. They built for document-term matrices, a sparse poisson latent block model, on the Poisson distribution. The proposed method is a meticulous parsimonious statistical model and has been devised from the ground up to deal with data sparsity problems. It simultaneously partitions the set of rows and columns to produce the essential co-clusters (diagonal co-clusters) that capture the most influential associations between the document and term clusters. The algorithm retrieves the legitimate cluster format of challenging as well as unbalanced datasets effectively.

Xu et al. [35] combined CNN and traditional unsupervised dimensionality reduction methods to cluster short-text documents. Park et al. [37] proposed an Advanced Document Clustering (ADC) using contextualized representations. This technology makes use of the advantage of context-aware representation and the sufficient similarity between documents, which reduces the difficulty of network training to the greatest extent. ADC performs clustering by measuring the semantic similarity between documents. The model tackles the high-dimensional features issue using cosine-similarity-based clustering and the mini-batch centroid update rule. Since cosine similarity computes the direction of document vectors and not the magnitude, ADC did not consider the magnitude of the document either.

Therefore, we adopt a similarity metric that takes the magnitude into account [29]. Contrary to ADC, our model takes advantage of CtAE to handle high-dimensional features.

2.3. Contractive autoencoder

The goal of any autoencoder model is to capture the most important features present in the data. Guo et al. proposed a new convolutional autoencoders [11] that does not need tedious layer-wise pre-training. However, the reconstructed image is often blurred, yielding to a lower quality due to compression during which some information is lost.

CtAE, sparse autoencoder and denoising autoencoder are different regularization techniques with their advantage and disadvantage. In deep neural networks, a slight disturbance in the initial layers leads to a substantial change in the later layers. To tackle this ill-posed problem, CtAE comes out with the objective of having a robust learned representation, which is less sensitive to a small variation in the data. To that end, it applies a penalty term such as Frobenius norm to the loss function to validate data representa-

tion. It calculates the Jacobian matrix of the hidden layer regarding the input by summing up the square of all elements. Therefore, the generated mapping forcefully contracts the data.

Considering an input x and the latent space h, Eq. (1) estimates the Jacobian of h in regard to x.

$$J_i = h_i(1 - h_i) * W_i \tag{1}$$

where W denotes weight matrix.

The Frobenius norm of the Jacobian at some point measures the contraction of the mapping locally at that point. Intuitively the contraction induced by the proposed penalty term can be measured beyond the immediate training examples by the ratio of the distances between two points in their original (input) space and their distance once mapped in the feature's location. In the limit where the variation in the input space is negligible, this corresponds to the derivative (i.e., Jacobian) of the representation map. The reconstruction error can be write as follows:

$$L = -\sum_{k=1}^{d}[x_k \log x'_k + (1 - x_k)\log(1 - x'_k)] + \lambda * \sum_{i=1}^{d}\sum_{j=1}^{n} J_{ij}^2 \tag{2}$$

In Eq. (2), the first term is the objective function of AE and the second term is the regularization term, λ is the hyper-parameter that controls the strength of the regularization, J represents the Jacobian. Penalizing the reconstruction loss with the Frobenius norm allows the input data to be locally invariant despite the direction changes.

The encoding function maps the input x_i to the hidden representation h_i, and then penalizes it with the sum of squares of all partial derivatives of the extracted features with respect to input dimensions.

$$\|J_F(x)\|_F^2 = \sum_{ij}\left(\frac{\partial h_j(x)}{\partial x_i}\right)^2 \tag{3}$$

From a geometrical perspective, the tenacity of the features can be seen as a contraction of the input space when projected in the feature space, particularly in the neighborhood of the specimens from the original data. In other ways, if the contraction were the same at all distances, it would not be useful, since it would just be a global scaling. Penalizing $\|J_F(x)\|_F^2$ in Eq. (3) encourages the mapping to the feature space to be contractive in the neighborhood of the training data. Such a contraction is happening with the proposed penalty, but much less without it. The CtAE objective function is given in Eq. (4)

$$L_{CtAE} = L_{REC} + \lambda\left\|J_f(x)\right\|_F^2 \tag{4}$$

where L_{REC} is the reconstruction loss as depicted in Section 3.1.1.

3. Proposed method

In this section, we introduce the proposed DECCA model in detail. To that end, we sum up the most frequent notations used in this paper, as shown in Table 1.

3.1. Deep embedded clustering based on contractive autoencoder

Given a gallery of n labeled document $X = \{x_i, c_i\}_{i=1}^n$ from C different classes, the task of our proposed algorithm is to retrieve similar documents from X for the query document by employing a given compatibility measure and to further rank the retrieved documents. Given the problem setup, the proposed framework first vectorizes the documents in terms of the TF-IDF and selects the relevant features.

Table 1
Symbols and their definitions.

Symbol	Description
x_i	the original data point
x'	the reconstructed data point
h	the latent space
L_C	the clustering loss
L_{FEAT}	the unsupervised feature learning loss
L_{REC}	the reconstruction loss
L_{SA}	the self-augmentation loss
L_{CtAE}	the contractive autoencoder loss
a_f	the encoder activation function
a_g	the decoder activation function
Z	the embedded features vectors
z_i	the embedded points
d_{AE}	the distance between the autoencoder input data and its reconstruction
sim	the similarity measurement
ED	the Euclidean Distance
Cos	the cosine similarity
$TS-SS$	the Triangle's Similarity (TS) and Sector's Area Similarity (SS)
μ_j	the cluster center
W	the encoder's weights
W'	the decoder's weights
λ	the hyper-parameter to control the regulation strength
τ	the hyper-parameter to balance L_C and L_{FEAT}

The DECCA structure is quite similar to the ADC model [37]. It falls into two parts: an unsupervised feature learning module where we deploy CtAE and an iterative clustering module where a clustering layer is connected to the embedded layer of CtAE, as depicted in Fig. 1.

The objective function of the proposed DECCA is given as follows:

$$L = \tau L_C + (1 - \tau)L_{FEAT} \tag{5}$$

where L_C is the clustering loss, L_{FEAT} refers to the learning loss, and $\tau \in [0, 1]$ is a constant training.

3.1.1. Unsupervised feature learning

The encoder function f maps an input x_i to the hidden representation $h(x_i)$.

$$h = f(x_i) = a_f(Wx_i + b_h) \tag{6}$$

Eq. (6) computes the projection of x_i into the latent space h. The decoder function g maps hidden representation h back to a reconstruction y.

$$y = g(h) = a_g(W'h + b_y) \tag{7}$$

where a_f and a_g are respectively the encoder's and decoder's activation function. Eq. (7) computes the reconstruction of the input.

An earlier version of CtAE is proposed in [28]. There are two main differences between the former and our proposed version. Instead of classification, we focus on clustering. We use Rectified Linear Unit (ReLU) instead of sigmoid as activation function to speed up the learning performance and reduce the training time as well as the computation cost.

The autoencoder comports two sections: an encoder and a decoder. The former maps the input x_i to a representation z_i in a latent space Z. During the training phase, the decoder strives to reconstitute the input data $x_i$from the embedding features z_i, making sure that the encoding phase has not lost useful information. The reconstruction loss measures the distance $d_{AE}(x_i, g(x_i))$ between the input $x_i$to the autoencoder and the corresponding reconstruction $g(x_i)$. Its formulation uses the mean squared error of the two variables as follows:

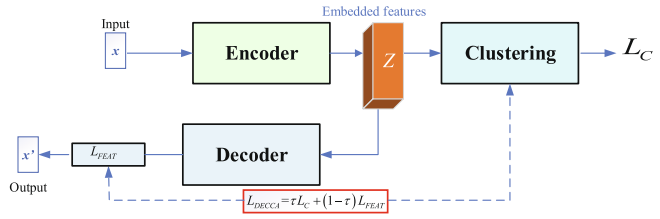


Fig. 1. The framework of the Deep Embedded Clustering based on the Contractive Autoencoder (DECCA) method. The unsupervised feature learning loss L_{FEAT} is composed of the reconstruction L_{REC} with the Jacobian penalty and self-augmentation losses. They ensure that the embedded features preserve the structure of data generating distribution and are insensitive to the input perturbations. The clustering loss L_C consists of *KL* divergence, and the locality preserving losses. It makes the soft cluster assignment probabilities stricter while maintaining the region of the clusters by pushing nearby data points.

$$L_{REC} = \sum_i \|x_i - g(x_i)\|^2 \tag{8}$$

where x_i is the input, $g(x_i)$ is the autoencoder reconstruction.

Self-Augmentation is a loss term that pushes together the representation of the original sample and their augmentations as follows:

$$L_{SA} = -\frac{1}{N}\sum_N sim(f(x_i), f(T(x_i))) \tag{9}$$

where x_i is the original sample, T is the augmentation function, $f(x_i)$ is the representation generated by the model, and *sim* is the similarity measurement as defined in (16).

After introducing the autoencoder reconstruction and the self-augmentation, the objective for the feature learning phase is as follows:

$$L_{FEAT} = L_{CtAE} + L_{SA} \tag{10}$$

where L_{FEAT} is the non-clustering loss, that is

$$L_{FEAT} = \sum_i \|x_i - f(x_i)\|^2 - \frac{1}{N}\sum_N sim(f(x_i), f(T(x_i))) + \lambda \|J_F(x_i)\|_F^2 \tag{11}$$

3.1.2. Clustering layer

The clustering layer maps each embedded point z_i of input data x_i into a label. Then the clustering loss *Lc* is defined as the sum of *KL* divergence and the locality preserving loss L_{LP}as follows.

$$L_{KL} = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{12}$$

where q_{ij} and p_{ij} are respectively the similarity between embedded point z_i and cluster center μ_j, and the target distribution are formulated as follows:

$$q_{ij} = \frac{\left(1 + \left\|z_i - \mu_j\right\|^2\right)^{-1}}{\sum_j \left(1 + \left\|z_i - \mu_j\right\|^2\right)^{-1}} \tag{13}$$

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j \left(q_{ij}^2 / \sum_i q_{ij}\right)} \tag{14}$$

Note that q_{ij} is measured by the student *t*-distribution. L_{KL} is minimized for the aforementioned q_{ij} and p_{ij} via neural network training.

$$L_{LP} = \sum_i \sum_{j \in N_k(i)} sim(x_i, x_j) \left\|z_i - z_j\right\|^2 \tag{15}$$

where $N_k(i)$ is the set of k nearest neighbors of the data point x_i, and $sim(x_i, x_j)$ is a similarity measurement between the points x_i and x_j. The well-known similarity metrics for DC are Cosine and Euclidean Distance (ED). Since both have drawbacks to be used alone, we employ the following similarity metric [29] to take advantage of the state-of-the-art document similarity metrics.

$$sim(x_i, x_j) = ED(x_i, x_j) \times Cos(x_i, x_j) + TS - SS(x_i, x_j) \tag{16}$$

where *ED* is the Euclidean Distance, *Cos* the cosine similarity and *TS-SS* the triangle's similarity and the sector's similarity between two documents *i* and *j*. We derive the objective function of the clustering step from the cluster assignment, and the locality preserving constraints are: $L_C = L_{KL} + L_{LP}$, where

$$L_C = \gamma \left(\begin{array}{l} \sum_i \sum_{j \in N_k(i)} sim(x_i, x_j) \left\|z_i - z_j\right\|^2 \\ + \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \end{array} \right) \tag{17}$$

Here $\gamma > 0$ is a coefficient that controls the degree of distorting the embedded area.

3.2. The DECCA optimization

We perform the training in two phases. In the first phase, we train the autoencoder using a standard reconstruction loss from Eq. (8) with Frobenius norm of the Jacobian matrix from Eq. (3) as the penalty term. Once trained, we use the learned parameter values of the resulting feature-extractors (weight and bias of the encoder) as initialization of a clustering layer. In the second phase, the network is pre-trained with the above configuration and fine-tuned using the cluster assignment hardening and the locality preserving losses respectively from Eqs. (12) and (15) while keeping the autoencoder reconstruction loss. We decide to keep the autoencoder reconstruction loss in this phase to avoid a risk of collapsing clusters contrary to DEC.

We follow the same optimization process of IDEC [12]. Given a mini batch with m samples and the learning rate λ, the cluster centre μ_j, the encoder's weights W and the decoder's weights W' are respectively updated by Eqs. (18)–(20)

$$\mu_j = \mu_j - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{dL_C}{d\mu_j} \tag{18}$$

$$W = W - \frac{\lambda}{m}\sum_{i=1}^{m} \left(\frac{dL_{FEAT}}{dW} + \tau \frac{dL_C}{dW}\right) \tag{19}$$

$$W' = W' - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{dL_{CtAE}}{dW'} \tag{20}$$

toward back-propagation optimization strategy. We compute gradients of error (loss) with respect to weights and then accordingly optimize weights using Gradient Descend to reduce the errors. Algorithm 1 describes the overall procedure of DECCA.

Algorithm 1. Deep Embedded Clustering Based on Contractive Autoencoder (DECCA)

Input: Input data: X; Number of clusters: K; Target distribution update interval: T; Stopping threshold: δ; Maximum iterations: $MaxIter$

Output: Contractive Autoencoder's weights W and W', Cluster centers μ, and Cluster labels C.

begin

Initialize μ, W and W' according to

Stage I: Train a CtAE and clustering with its features

for $x \in X$ **do**

Add the regularization term from Equation (1),

Computes the reconstruction error using Equation (2),

Computes the Jacobian of h with respect to x_i using Equation (3)

Computes the projection of the x_i input into the latent space h using Equation (6)

Computes the reconstruction of the input x using Equation (7)

Stage II: Jointly learn the features and cluster centers

while $iter \leq MaxIter$ **do**

Compute all embedded points:

Update P using Equation (13), Equation (14) and save the last label assignment: $s_{old} = s$

Compute new label assignments s using $s_i = \arg\max_j q_{ij}$

if $\frac{\sum(s_{old} \neq s)}{n} < \delta$ **then**

Stop training.

Choose a batch of samples $S \in X$

Update μ, W and W', using Equation (18), Equation (19) and Equation (20) on S.

4. Experiments

In this section, we evaluate the proposed DECCA for document and image representation using three datasets in each case. Afterward, we compare the model to the traditional *k*-means and seven state-of-the-art DL clustering algorithms.

4.1. Datasets

We followed the data preprocessing as depicted in [38]. The original data is randomly split into training and testing sets with different size to avoid the over-fitting problem. The statistic of the datasets is given in Table 2.

- MNIST: This dataset consists of 28 by 28 gray-scale images of 70000 handwritten digits, which is randomly split into training and testing sets of size 60*K* and 10*K*, respectively.
- FASHION-MNIST: It is a database of fashion articles. The dataset contains 60000 28 by 28 gray scale images of 10 fashion categories and can be used as a drop-in replacement for MNIST.
- USPS: This dataset is a handwritten digit from the USPS postal service, it contains 9298 samples of 16 by 16 images.
- REUTERS: Reuters contains around 810000 English news stories labeled with a category tree. The corpus is a collection of 11228 text documents partitioned into 46 different groups. Following the DEC model [10], we used four root categories: corporate/industrial, government/social, markets, and economics as labels and excluded all documents with multiple labels. Noticed that some SC methods (e.g., LDMGI) could not scale to full Reuters dataset, we will use instead REUTERS-10K.
  REUTERS-10K: Restricted by computational resources, we randomly sampled a subset of 10000 examples and computed TF-IDF features on the 2000 most frequent words from the REUTERS dataset. The sampled dataset is referred to as REUTERS-10K.
- 20NEWSGROUPS: This dataset is a collection of about 20000 newsgroup documents. The corpus is a collection of 18846 text documents. These documents are partitioned into 20 different groups according to their topics. We represented every document as a vector of TF-IDF scores of each word and built the cosine similarity graph based on the TF-IDF scores. We use the 2000 highest TF-IDF score as the features.

4.2. Evaluation metrics

The clustering performance of all algorithms is measured by the unsupervised clustering indexes.

- Accuracy (ACC) [5]:

$$ACC = \max_{m \in M} \frac{1}{N} \sum_{i=1}^{N} 1\{g_i = m(c_i)\} \tag{21}$$

where N is the total number of samples, g_i is the ground-truth label, and M is the set of all possible one-to-one mappings, c_i is the cluster assignment produced by the mixture assignment network, i.e., $c_i = \arg\max_k p_k^{(i)}$

- Normalized Mutual Information (NMI)[25]: Given a piece of mutual information $I(\Omega,)$ which measures the information gain to the true partition after knowing the clustering result, $H(\cdot)$ the entropy, the NMI is calculated with the following formula:

$$NMI(c, g) = \frac{I(c, g)}{\sqrt{H(c)H(g)}} \tag{22}$$

where c and g respectively denote clustering label and ground-truth label of any given sample.

4.3. Compared algorithms

The number of clusters K is set to the number of classes for each dataset. When the algorithm code is publicly available, we run it and put the result; otherwise, we quote it from the original paper. We perform 20 random restarts when initializing all clustering models and pick the result with the average objective value. To validate the effectiveness of our proposed method, we compare it with the following state-of-the-art methods in terms of the multi-view and deep auto-encoders based clustering.

- Deep Embedded Clustering[1] (DEC) [10] is specially designed to learn a representation for clustering without ground truth cluster membership labels. The algorithm iteratively optimizes a *KL* divergence-based clustering objective with a self-training target distribution to clusters a set of data points in a jointly optimized feature space.
- Improved Deep Embedded Clustering[2] (IDEC) [12] algorithm applies an under-complete autoencoder to deal with the data structure preservation by manipulating the feature space to scat-

[1] https://github.com/piiswrong/dec.
[2] https://github.com/dawnranger/IDEC-pytorch.

Table 2
Statistic of the datasets.

Datasets	#Points	Train_set	Test_set	#Clusters	#Dimensions	Type
MNIST	70000	60000	10000	10	784	Image
Fashion	70000	60000	10000	10	784	Image
USPS	9298	7291	2007	10	256	Image
Reuters_10k	10000	8000	2000	4	2000	Text
Reuters	11228	8982	2246	4	2000	Text
20NEWSGROUPS	18846	11314	7532	20	2000	Text

ter data points under the clustering loss guidance. It combines the clustering and the autoencoder's reconstruction losses to jointly optimize cluster labels assignment and learn features that are relevant for clustering.

- Deep Clustering Network[3] (DCN) [39] joints dimensionality reduction (DR) and *k*-means clustering approach in which DR is accomplished via learning a deep neural network.
- Variational Deep Embedding[4] (VaDE) [7] embeds the probabilistic clustering problems into a variational auto-encoder framework. The algorithm first, models the generative data procedure by a GMM model and a neural network. Then it maximizes the evidence lower bound of the log-likelihood of data using the SGVB estimator and the reparameterization trick.
- Mixture of Autoencoders[5] (MIXAE) [5] is a collection of autoencoders where each autoencoder learns the underlying manifold of a group of similar objects and then links together several latent vectors from the autoencoders which serve as input and derive the distribution over clusters.
- Advanced document clustering using contextualized representations (ADC) [37] extracts features from pre-trained language models and initialize cluster centroids to spread out uniformly. In the clustering module of ADC, the semantic similarity is measured using the cosine similarity and centroids update while assigning centroids to a mini-batch input.
- Semi-supervised deep embedded clustering[6] (SDEC) [34] learns feature representations that favor the clustering tasks and performs clustering assignments simultaneously by incorporating the pairwise constraints in the feature learning process.
- Maximizing Self-Augmented Training[7] (IMSAT) [30] maximize information-theoretic dependency between inputs and their mapped outputs while regularizing the mapping function.
- Clustering via Cross-Metrics Verification [45] employs a fully connected neural network to jointly learn a collection of hierarchical representation and cluster assignments in an end-to-end manner.
- Deep clustering via a Gaussian-mixture variational autoencoder with Graph embedding[8] (DGG) [46] respectively applies Gaussian mixture model (GMM) as the prior in VAE to facilitate clustering and graph embedding to handle data with a complex spread.

4.4. Parameters setups

The following are methods to assign and set the values of τ:

- Scenario 1: we train the network using the non-clustering loss (11) only by setting τ to 0.
- Scenario 2: Subsequently, τ is set to 1, in this case, the non-clustering network branches (e.g., autoencoder's decoder) are removed and the clustering loss (17) is used to train (fine-tune) the obtained network. If the constraint forced by the reconstruction loss is lost after training the system long enough for clustering, which may lead to worse results, we move to the next assumption.
- Variable scenario and fine-tuning: τ is varied during the training. We start with the highest value for τ and gradually decrease by ten it in every phase of the training. This case will consider the disadvantages of both scenario 1 and 2.

The optimizer SGD with learning rate $\lambda = 0.1$ and momentum $\beta = 0.9$ is used for 20NEWSGROUPS and $\lambda = 0.01$ for the other datasets.

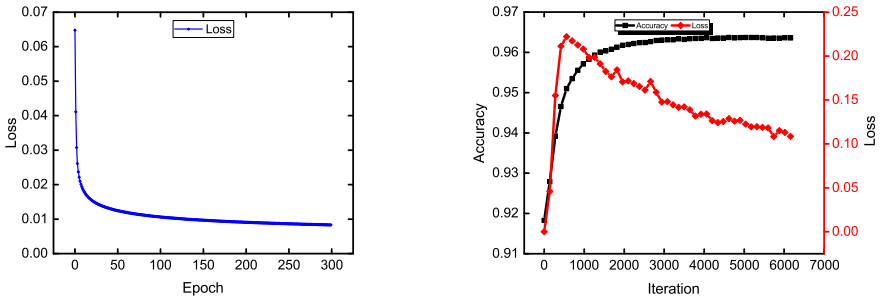
4.5. Results and discussion

We evaluate the accuracy of our proposed method for every dataset from Figs. 2–7. For each figure, on the left side, we report the training loss function values over the epoch. On the right side, we gauge the clustering accuracy and the corresponding loss cost. As shown in Fig. 2a) and Fig. 3(a) the loss decreases rapidly and converges to a stable value of 0.1 for both MNIST and FMNIST datasets. The model converges to the value of 0.02 (Fig. 4(a)) for USPS, 0.55 (Fig. 5(a) for REUTERS, 0.31 (Fig. 6(a)) for REUTERS10K and 0.0032 (Fig. 7(a)) for 20NEWSGROUPS. Moreover, the accuracy of the clustering as shown in Figs. 2(b), 3(b), 4(b) and 5(b) agree with the loss function curves. In Figs. 6(b) and 7(b) the accuracy first declines because the model begins to focus on the noise in the training dataset and extracts features based on it, consequently leads to the over-fitting problem. Indeed, that helps the model to improve its performance on the training set; however, it hurts its ability to generalize, so the accuracy of the validation set decreases. In Fig. 6(b) the accuracy rises because the model starts performing the data augmentation, which increases the training set size to deal with this over-fitting problem. Tables 3 and 4 respectively report the results of all comparing algorithms on image and text datasets.

From Table 3, we can find that all the DL models perform well on image datasets. What is more, DECCA outperforms traditional deep clustering algorithms with a large margin. That indicates the fascinating potentials of contractive autoencoder in the unsupervised clustering field. However, in Table 4, it appears that the state-of-the-art DL models yield poor accuracy on text data. Compared to other algorithms, our model gets higher accuracy on all text datasets. Undoubtedly, DECCA is efficient to learn discriminative features and the samples are gradually well separated with training procedure. Additionally, ADC is more effective in the case of documents in the natural language form rather than in the case of documents with listed tokens, which is not inherent in the real world. Contrary to ADC, DECCA performs well in both cases.

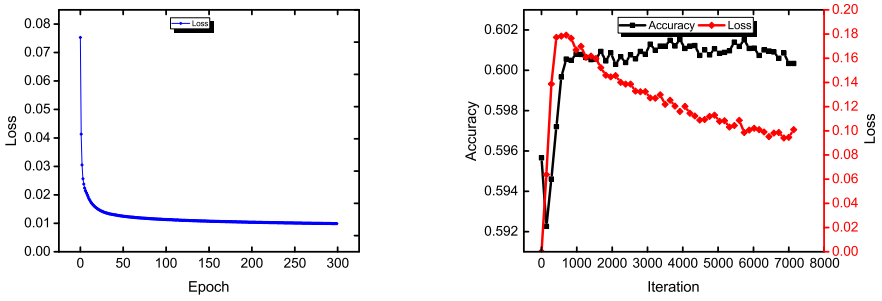
The performance gap between our model and previous DL algorithms reflects the effect of balancing the non-clustering and the clustering losses while controlling the regularization length λ.

[3] https://github.com/boyangumn/DCN.
[4] https://github.com/slim1017/VaDE.
[5] https://github.com/icannos/mixture-autoencoder.
[6] https://github.com/yongzx/SDEC-Keras.
[7] https://github.com/weihua916/imsat.
[8] https://github.com/ngoc-nguyen-0/DGG.



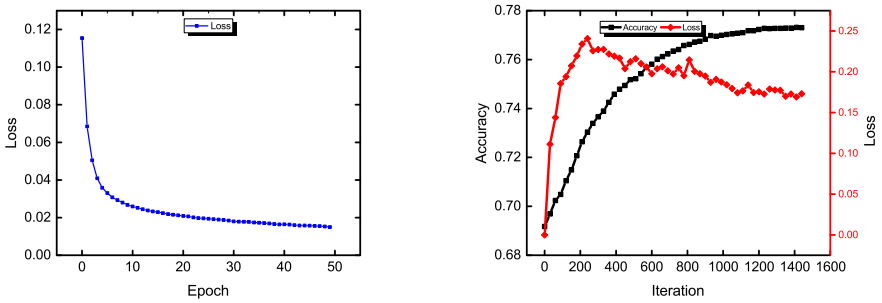
(a) Training (b) Clustering

Fig. 2. Accuracy and loss of the proposed method on MNIST.



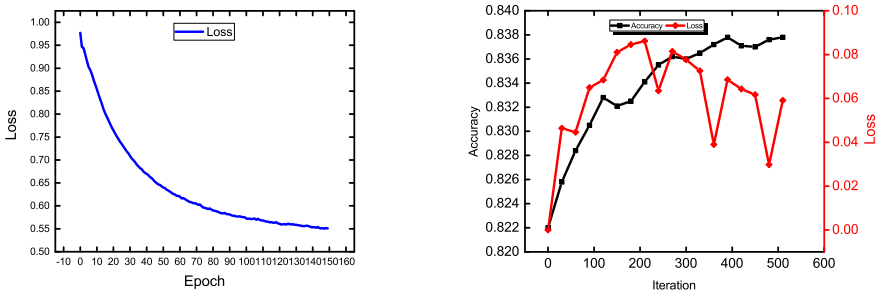
(a) Training (b) Clustering

Fig. 3. Accuracy and loss of the proposed method on FMNIST.



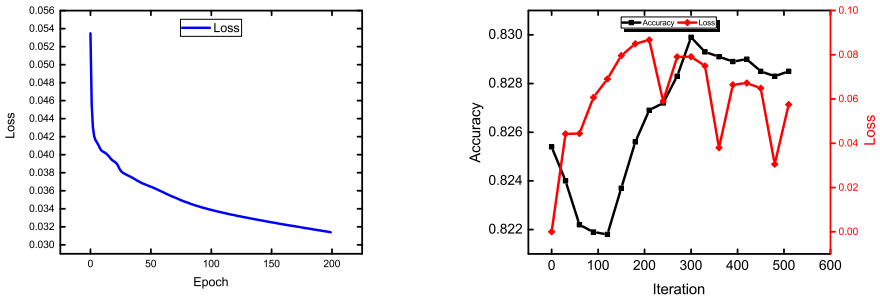
(a) Training (b) Clustering

Fig. 4. Accuracy and loss of the proposed method on USPS.



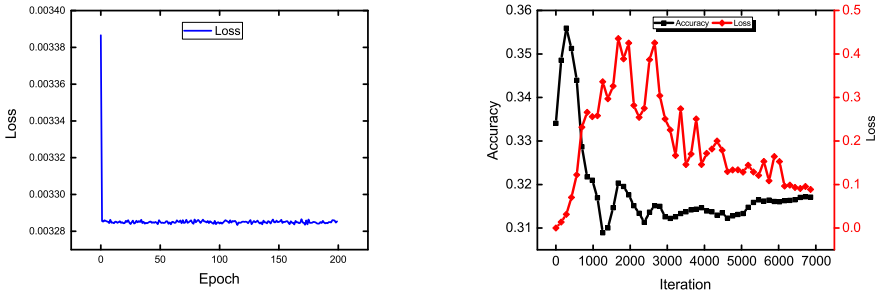
(a) Training (b) Clustering

Fig. 5. Accuracy and loss of the proposed method on REUTERS.



(a) Training (b) Clustering

Fig. 6. Accuracy and loss of the proposed method on REUTERS10K.



(a) Training (b) Clustering

Fig. 7. Accuracy and loss of the proposed method on 20NEWSGROUPS.

Table 3
Clustering performance on image datasets by ACC and NMI.

Methods	MNIST		USPS		Fashion_mnist	
	ACC	NMI	ACC	NMI	ACC	NMI
k-means -MacQueen (1967)	0.5348	0.5000	0.6685	0.6269	0.4758	0.5124
DEC Xie et al. (2016)	0.8430	0.8340	0.7620	0.7670	0.5180	0.5460
IDEC Guo et al. (2017)	0.8806	0.8672	0.7605	0.7846	0.529	0.5570
VaDE Jiang et al. (2017)	0.9446	0.8760	0.5660	0.5120	0.5780	0.6300
DCN Yang et al. (2017)	0.8300	0.8100	0.6880	0.6830	0.5010	0.5580
IMSAT (RPT) Hu et al. (2017)	0.8960	–	–	–	–	–
IMSAT (VAT) Hu et al. (2017)	**0.9840**	–	–	–	–	
MIXAE Zhang et al. (2017)	0.8560	–	–	–	–	–
SDEC Ren et al. (2019)	0.8611	0.8289	0.7639	0.7768	0.5280	0.5931
HOE(CNN) Peng et al. (2019)	0.9325	0.8639	–	–	–	–
DGG Yang et al. (2019)	0.9758					
DECCA	0.9637	**0.9074**	**0.7731**	**0.8053**	**0.6099**	**0.6698**

Table 4
Clustering performance on text datasets by ACC and NMI.

Methods	REUTERS-10 K		REUTERS		20NEWSGROUPS	
	ACC	NMI	ACC	NMI	ACC	NMI
k-means -MacQueen (1967)	0.5493	0.3538	0.5152	0.3090	0.2390	0.2323
DEC Xie et al. (2016)	0.7217	0.4976	0.7563	0.4812‘	0.3080	–
IDEC Guo et al. (2017)	0.7564	0.4981	0.5306	0.2094	0.3241	0.3097
VaDE Jiang et al. (2017)	0.7983	–	0.7938	–	–	–
IMSAT (RPT) Hu et al. (2017)	0.7190	–	–	–	0.2440	–
IMSAT (VAT) Hu et al. (2017)	0.7100	–	–	–	0.3110	–
MIXAE Zhang et al. (2017)	–	–	0.7940	–	–	–
ADC Park et al. (2019)	0.6440	0.3813	0.6711	0.3969	–	–
SDEC Ren et al. (2019)	0.6792	0.5086	0.5054	0.3350	0.3104	0.3155
HOT Peng et al. (2019)	–	–	0.7098	0.3601	–	–
DGG Yang et al. (2019)			0.8230			
DECCA	**0.8299**	**0.6082**	**0.8378**	**0.6343**	**0.3559**	**0.3423**

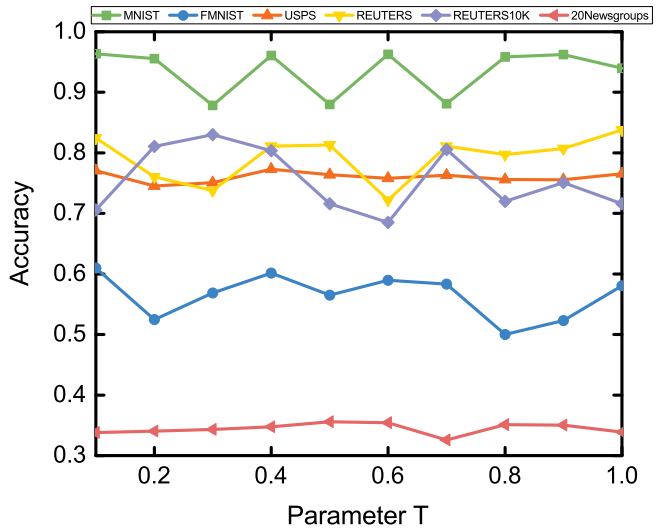


Fig. 8. The effect of learning rate and clustering coefficient in (5) on clustering performance for the datasets.

Fig. 8 shows the performance of our proposed method on six datasets by varying τ from 0.1 to 1.

As can be observed from Table 5, the performance of DECCA varies on the dataset with the values of τ. For instance, when $\tau = 0.1$, the model gets the best achievement on MNIST and FMNIST datasets. Except for the Reuters data, the model performs better when $0.1 \leqslant \tau \leqslant 0.5$. Compared to IDEC which fixes the degree of distorting embedded space τ to 0.1, our model automatically balances the non-clustering and the clustering losses to cluster the dataset while controlling the regulation length λ . This procedure reduces considerably the convergence time. In comparison with DEC and IDEC techniques, our method has the advantage of CtAE. The model defies the minuscule disturbances the input which leads to a fast convergence as depicted in Fig. 9(b). Applying a Frobenius term as penalty to the reconstruction loss function allows DECCA to be 3000 iterations faster than DEC as it can be observed in Fig. 9(a).

Note that $\tau = 1$ means we directly use the clustering layer, and the non-clustering network branches (autoencoder's decoder) are removed. That leads to a worse accuracy for every dataset. We can conclude that the decoder plays an essential role in preserving the local structure of data contrast to the assumption in DEC [10]. Toward such conditions combining these loss functions can manipulate the embedded space to get better clustering accuracy.

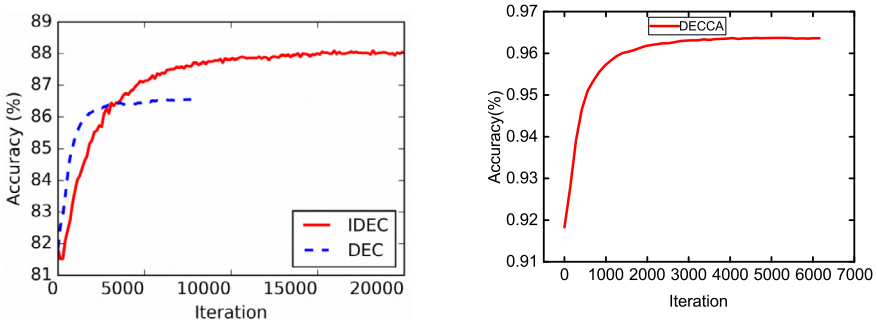
5. Conclusion

In this work, we proposed a novel contractive deep embedding clustering architecture to learn a discriminative embedding space for clustering document data. The CtAE based representations can capture the local inter-actions of the document in a better fashion. The DECCA model can learn a document vector embedding considering the variation in the data. This framework helps us in preserving the neighborhood information of the local structure and stocking their non-linearity. Overall, the results on six real-world datasets demonstrated that CtAE surpasses results obtained by regularizing autoencoder using weight decay or by denoising. Future research should consider the potential effects of the recurrent neural network on document clustering more carefully. Besides, combining it with CtAE may improve the clustering accuracy.

Table 5
Comparison of the threshold parameter on different datasets.

τ	MNIST	FMNIST	USPS	REUTERS	REUTERS10K	20NEWSGROUPS
0.1	**0.9637**	**0.6099**	0.7706	0.8247	0.7057	0.3381
0.2	0.9556	0.5244	0.7454	0.7603	0.8108	0.3406
0.3	0.8785	0.5686	0.7508	0.7379	**0.8299**	0.3433
0.4	0.9610	0.6015	**0.7732**	0.8111	0.8034	0.3477
0.5	0.8799	0.5650	0.7637	0.8132	0.7158	**0.3560**
0.6	0.9627	0.5895	0.7581	0.7220	0.6849	0.3544
0.7	0.8813	0.5833	0.7632	0.8109	0.8055	0.3257
0.8	0.9583	0.5001	0.7560	0.7971	0.7200	0.3511
0.9	0.9622	0.5230	0.7555	0.8070	0.7507	0.3505
1	0.9396	0.5803	0.7653	**0.8378**	0.7158	0.3388

Bold values are the best values for every dataset.



(a) DEC-IDEC (b) DECCA

Fig. 9. Convergence comparison between DECCA, DEC and IDEC on MNIST.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by the National Science Foundation of China (Nos. 61603313, 61573292, 61772435, 61976182, 61876157).

References

[1] L. Yu, W. Wang, DCSR: Deep clustering under similarity and reconstruction constraints, Neurocomputing 411 (2020) 216–228.
[2] C. Xu, Y. Dai, R. Lin, S. Wang, Deep clustering by maximizing mutual information in variational auto-encoder, Knowledge-Based Systems 205 (2020) 1–8.
[3] W. Yan, B. Zhang, S. Ma, Z. Yang, A novel regularized concept factorization for document clustering, Knowledge-Based Systems 135 (2017) 147–158.
[4] J. Yi, Y. Zhang, X. Zhao, J. Wan, A novel text clustering approach using deep-learning vocabulary network, Mathematical Problems in Engineering (2017) 1–13.
[5] D. Zhang, Y. Sun, B. Eriksson, and L. Balzano, "Deep unsupervised clustering using mixture of autoencoders," arXiv: Learning, 2017..
[6] M.T. Law, R. Urtasun, R.S. Zemel, Deep spectral clustering learning, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1985–1994.
[7] Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: an unsupervised and generative approach to clustering, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 1965–1972.
[8] F. Tian, B. Gao, Q. Cui, E. Chen, T.-Y. Liu, Learning deep representations for graph clustering, in: Proceedings of the 28th AAAI Conference on Artificial Intelligence, 2014, pp. 1293–1299.
[9] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, Z. Yi, Deep subspace clustering with sparsity prior, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2016, pp. 1925–1931.
[10] J. Xie, R. Girshick, A. Farhadi, Unsupervised deep embedding for clustering analysis, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 478–487.
[11] X. Guo, X. Liu, E. Zhu, J. Yin, Deep clustering with convolutional autoencoders, in: Proceedings of the International Conference on Neural Information Processing, 2017, pp. 373–382.
[12] X. Guo, L. Gao, X. Liu, J. Yin, Improved deep embedded clustering with local structure preservation, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, 2017, pp. 1753–1759.
[13] F. Li, H. Qiao, B. Zhang, Discriminatively boosted image clustering with fully convolutional auto-encoders, Pattern Recognition 83 (2018) 161–173.
[14] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5147–5156.
[15] M. Kulkarni, S.S. Karande, S. Lodha, Unsupervised word clustering using deep features, in: Proceedings of the 12th IAPR Workshop on Document Analysis Systems, 2016, pp. 263–268.
[16] S. Subramani, V. Sridhar, K. Shetty, A novel approach of neural topic modelling for document clustering, in: Proceedings of the IEEE Symposium Series on Computational Intelligence, 2018, pp. 2169–2173.
[17] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, J. Long, A survey of clustering with deep learning: From the perspective of network architecture, IEEE Access 6 (2018) 39501–39514..
[18] S. Wang, J. Cai, Q. Lin, W. Guo, An overview of unsupervised deep feature representation for text categorization, IEEE Transactions on Computational Social Systems 6 (2019) 504–517.
[19] W. Zhang, Y. Li, S. Wang, Learning document representation via topic-enhanced LSTM model, Knowledge-Based Systems 174 (2019) 194–204.
[20] P. Dahal, Learning embedding space for clustering from deep representations, in: Proceedings of the IEEE International Conference on Big Data, 2018, pp. 3747–3755.
[21] C. Xu, Z. Guan, W. Zhao, Y. Niu, Q. Wang, Z. Wang, Deep multi-view concept learning, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2018, pp. 2898–2904.
[22] L. Zong, X. Zhang, L. Zhao, H. Yu, Q. Zhao, Multi-view clustering via multi-manifold regularized non-negative matrix factorization, Neural Networks 88 (2017) 74–89.
[23] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, B.W. Schuller, A deep matrix factorization method for learning attribute representations, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (3) (2016) 417–429.
[24] S. Huang, Z. Kang, Z. Xu, Auto-weighted multi-view clustering via deep matrix decomposition, Pattern Recognition 97 (2020) (Article 107015)..
[25] P. Huang, Y. Huang, W. Wang, L. Wang, Deep embedding network for clustering, in: Proceedings of the 22nd International Conference on Pattern Recognition, 2014, pp. 1532–1537.
[26] K. Ghasedi Dizaji, A. Herandi, C. Deng, W. Cai, H. Huang, Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5736–5745.
[27] Z. Mo, J. Ma, DocNet: A document embedding approach based on neural networks, in: Proceedings of the 24th International Conference on Automation and Computing, 2018, pp. 1–5.
[28] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 833–840.
[29] B. Diallo, J. Hu, T. Li, G. Khan, C. Ji, Concept-enhanced multi-view clustering of document data, in: Proceedings of the 14th International Conference on Intelligent Systems and Knowledge Engineering, 2019, pp. 1258–1264.
[30] W. Hu, T. Miyato, S. Tokui, E. Matsumoto, M. Sugiyama, Learning discrete representations via information maximizing self-augmented training, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1558–1567.
[31] Q. Huang, Y. Zhang, H. Peng, T. Dan, W. Weng, H. Cai, Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning, Neurocomputing 404 (2020) 340–350.
[32] J. Wang, J. Jiang, Sa-Net: A deep spectral analysis network for image clustering, Neurocomputing 383 (2020) 10–23.
[33] X. Guo, E. Zhu, X. Liu, J. Yin, Deep embedded clustering with data augmentation, in: Proceedings of the Asian Conference on Machine Learning, 2018, pp. 550–565.
[34] Y. Ren, K. Hu, X. Dai, L. Pan, S.C. Hoi, Z. Xu, Semi-supervised deep embedded clustering, Neurocomputing 325 (2019) 121–130.
[35] J. Xu, B. Xu, P. Wang, S. Zheng, G. Tian, J. Zhao, Self-taught convolutional neural networks for short text clustering, Neural Networks 88 (2017) 22–31.
[36] Y. Ren, N. Wang, M. Li, Z. Xu, Deep density-based image clustering, Knowledge-Based Systems (2020) 1–10.
[37] J. Park, C. Park, J. Kim, M. Cho, S. Park, ADC: Advanced document clustering using contextualized representations, Expert Systems with Applications 137 (2019) 157–166.
[38] X. Wang, D. Peng, P. Hu, Y. Sang, Adversarial correlated autoencoder for unsupervised multi-view representation learning, Knowledge Based Systems 168 (2019) 109–120.
[39] B. Yang, X. Fu, N.D. Sidiropoulos, M. Hong, Towards k-means-friendly spaces: simultaneous deep learning and clustering, in: Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 3861–3870.
[40] M. Lu, X.-J. Zhao, L. Zhang, F.-Z. Li, Semi-supervised concept factorization for document clustering, Information Sciences 331 (2016) 86–98.
[41] L.M. Abualigah, A.T. Khader, E.S. Hanandeh, A new feature selection method to improve the document clustering using particle swarm optimization algorithm, Journal of Computational Science 25 (2018) 456–466.
[42] M. Ailem, F. Role, M. Nadif, Sparse poisson latent block model for document clustering, IEEE Transactions on Knowledge and Data Engineering 29 (7) (2017) 1563–1576.
[43] M. Caron, P. Bojanowski, A. Joulin, M. Douze, Deep clustering for unsupervised learning of visual features, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 132–149.
[44] S. Huang, Z. Xu, J. Lv, Adaptive local structure learning for document co-clustering, Knowledge-Based Systems 148 (2018) 74–84.
[45] X. Peng, H. Zhu, J. Feng, C. Shen, H. Zhang, J.T. Zhou, Deep clustering with sample-assignment invariance prior, IEEE Transactions on Neural Networks and Learning Systems (2019) 1–12.
[46] L. Yang, N.-M. Cheung, J. Li, J. Fang, Deep clustering by gaussian mixture variational autoencoders with graph embedding, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6440–6449.

Bassoma Diallo received his M.S. degrees respectively from University of Sciences Techniques and Technology of Bamako in the field of applied physics in 2009 and from Wuhan University of Technology in the field of applied technology of computer in 2017. He is an associate professor of Institute of Teacher Training of KATI (MALI). Currently, he is a Ph.D. candidate at the Institute of Artificial Intelligence, Southwest Jiaotong University (CHINA). His research interests include the areas of machine learning, data mining, big data, clustering analysis and deep learning.

Jie Hu received her B.Ed. degree from Southwest Normal University in 2001 and M.S. degree from East China Normal University in 2007. She received her Ph.D. degree from Southwest Jiaotong University in 2016. In 2018–2019, she was a visiting scholar at the Georgia State University. She is currently an associate professor of School of Information Science and Technology, Southwest Jiaotong University. Her research interests include the areas of data mining, big data, clustering analysis and clustering ensemble. She is the recipient of 2017 ACM Chengdu Doctoral Dissertation Award.

Tianrui Li received the B.S., M.S., and Ph.D. degrees from the Southwest Jiaotong University, Chengdu, China, in 1992, 1995, and 2002, respectively. He was a postdoctoral researcher with SCKCEN, Belgium, from 2005 to 2006, and a visiting professor with Hasselt University, Belgium, in 2008, the University of Technology, Sydney, Australia, in 2009, and the University of Regina, Canada, 2014. He is currently a professor and the director of the Key Laboratory of Cloud Computing and Intelligent Techniques, Southwest Jiaotong University. He has authored or coauthored more than 300 research papers in refereed journals and conferences. His research interests include big data, cloud computing, data mining, granular computing and rough sets. He is a fellow of IRSS and a senior member of ACM and IEEE.

Ghufran Ahmad Khan received the M.S. degree in computer science from Aligarh Muslim University, Aligarh, India. He is currently pursuing the Ph.D. degree from the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China. He has huge Academic and Technical Experience in India. He has authored some research articles. His research areas include machine learning, data mining, rough set theory, and deep learning.

Xinyan Liang was born in 1989. He received the BSc degree at the School of Computer and Information technology from Shanxi University, China, in 2014. Currently, he is a PhD candidate at the Institute of Big Data Science and Industry, Shanxi University. His main research interests include multi-modal machine learning, multi-view learning and granular computing.

Yimiao Zhao is a graduate student in the School of Information Science and Technology, Southwest Jiaotong University of China. She received her bachelor degree in the School of Information Science and Technology from Southwest Jiaotong University, China, in 2018. Her research interests include the areas of machine learning, data mining and clustering analysis.