Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Learning discriminative representations for deep clustering

Thi-Anh-Loan Trinh · The-Anh Pham

Received: date / Accepted: date

Abstract Clustering is of central importance to many computer vision applications such as image understanding, indexing, searching, and product quantization. Recent advances on deep neural networks have brought promising solutions for clustering analysis. This paper presents a deep autoencoder model to learn discriminative representations for clustering. To this aim, we have utilized the distance-based correlation in the probabilistic space to design novel loss functions. The designed correlation loss is incorporated at the outputs of encoder and decoder for local structure preservation. Noticeably, a novel objective function that captures the probabilistic affinity space is presented to enhance the latent representations targeted to better clustering quality. Moreover, we utilize a clustering-oriented loss based on KL divergence to enhance the latent features. The proposed model has been validated by extensive experiments in comparison with the state-of-the-art methods and accomplished promising clustering accuracy scores on four benchmark datasets: Reuters-10K (83.9%), STL-10 (96.5%), CIFAR-10 (79.7%), and Fashion-MNIST (66.0%).

Keywords Deep Embedding Clustering · Autoencoder · Clustering loss functions

1 Introduction

Clustering analysis plays an essential role to many applications, typically feature quantization, indexing and searching. In pattern recognition, feature matching is a fundamental task and poses many challenges due to the curse of dimensionality problem. To facilitate efficient feature matching, approximate nearest neighbor search has been adopted in combination with a vector quantization method such as PQ [13] or PSVQ [26]. In these applications, designing a good clustering algorithm is the key to accomplish desirable search performance.

In the literature, a large number of works have been presented to the problem of data clustering. The methods can be classified by traditional methods (e.g., K-means [23]) or modern approaches based on deep neural network (DNN) such as DEC [33], ClusterGAN [24], VaDE [14]. Although DNN-based clustering methods achieve remarkable performance, they have limitations in specific contexts. DEC [33], for example, is subjected to the distortion of trainable weights due to the exclusion of decoder component in fine-tuning step. Recent advanced methods (DGC [29], VaGAN-GMM [36]) must design particular neural networks for different data representations (e.g., convolutional layers for visual images, while dense networks for textual data). Some methods [3], [5] exploit only the learned representation spaces without considering clustering-specific objective functions. Moreover, in the terms of clustering accuracy, it is less easy to find a method that performs well across different data sources. For instances, ClusterGAN [24] or DAC [2] achieve interesting results on the MNIST dataset [17] but work poorly on CIFAR-10 [16] and Fashion-MNIST [32].

In this paper, we present a clustering model based on autoencoder framework with effective loss functions. Instead of improving network architecture, we investigate the incorporation of clustering-oriented objectives. To this aim, we exploit pairwise similarities [5] to estimate the correlation loss that helps to preserve local

Thi-Anh-Loan Trinh
Hong Duc University (HDU), Thanh Hoa city, Vietnam.
E-mail: trinhthianhloan@hdu.edu.vn

Corresponding author: The-Anh Pham
Hong Duc University (HDU), Thanh Hoa city, Vietnam.
E-mail: phamtheanh@hdu.edu.vn

structures of data distributions. Furthermore, we derive a new objective function incorporating affinity distances in the probabilistic space to extract features that are conducive to clustering. Moreover, the clustering-oriented loss is also embedded into the model to learn discriminated clustering features. Extensive experiments have been conducted on the four benchmark datasets (i.e., Fashion-MNIST [32], CIFAR-10 [16], STL-10 [4], and Reuters-10K [18]) demonstrating that the presented model achieves promising clustering performance in comparison with the state-fo-the-art methods.

The rest of this paper has been organized as follows. Section 2 presents a deep review of the-state-of-the-art methods for deep clustering. Section 3 describes in details the proposed clustering model along with effective loss functions. Section 4 dedicates to the experimental analysis and discussion. Section 5 concludes the paper and outlines several directions of future work.

2 Related work

In this part, an extensive review of existing methods for deep clustering is presented. The methods can be grouped into two main approaches: autoencoder (AE) and generative adversarial networks (GAN). For each approach, we shall restrict our discussion to the most representative methods.

AE-based methods: A pioneer work on autoencoder for solving the clustering problem can be referred to DEC (Deep Embedding Clustering) [33]. This method is designed as an autoencoder using fully connected neural layers. DEC transforms an input X into the representation layer Z and partitions the embedded points in Z into K clusters whose centers are denoted by $\{\mu_j\}_{j=1}^{K}$. To train the autoencoder, a greedy layer-wise procedure is presented that minimizes the reconstruction loss (L_r) defined as follows:

$$L_r = ||x - g_{W'}(f_W(x))||_2^2 \quad (1)$$

where $x \in X$ is a data point, and W, W' are the trainable weights of the encoder and decoder, respectively.

Once the autoencoder is trained, DEC removes the decoder component. Next, a cluster layer is added to the top of representation layer and a clustering oriented objective is designed to fine-tune the model. To be specific, the loss function, defined as $L_c = KL(P||Q)$, minimizes the KL divergence of two distributions. Here $Q = \{q_{ij}\}$ denotes the set of soft assignments and is computed by using the Student's t-distribution. Furthermore, q_{ij} can be considered as the probability of assigning a point $z_i = f_W(x_i)$ to a center μ_j. The set $P = \{p_{ij}\}$ plays the role of an auxiliary target distribution and is estimated by using a self-learning fashion.

IDEC (Improved Deep Embedded Clustering) [10] improves DEC by putting more focus on local structure preservation. IDEC keeps the decoder part and combines the reconstruction and clustering loss functions to drive the fine-tuning process. As a result, the final objective is defined as: $L = L_r + \gamma L_c$ where γ is a constant parameter balancing the contribution of the two loss functions. DEC-DA (Deep Embedded Clustering with Data Augmentation) [11] improves DEC and IDEC by incorporating more advanced techniques for data augmentation (e.g., random rotation, shifting, and cropping). In addition, DEC-DA explores different backbones for the autoencoder, such as fully connected (FC) model and convolutional neural network (CNN).

DEPICT [6] presents a new clustering architecture based on convolutional autoencoder that incoporates a denoising path to enforce the robustness. Furthermore, DEPICT investigates several novel ideas for deriving objective functions. First, reconstruction loss is added at every layer of the clean pathway for regularization. Next, cluster soft assignments are created by using a soft-max layer placed at the top of embedding spaces. Finally, clustering loss is derived as the KL divergence between soft assignments and a target distribution.

Recently, the authors in [1] presented a new clustering framework, namely EDESC (Efficient Deep Embedded Subspace Clustering) that is based on deep subspace model. This model consists of an autoencoder to generate embedded space Z for input data. Next, subspace bases D are learned from Z in an iterative refining fashion. Both Z and D are jointly optimized to strengthen the discriminated representation capability. The network does not employ self-expressive mechanism and thus is applicable to large-scale and time-critical clustering applications.

Instead of developing network model, several works focus on improving objective functions. The authors in [27] introduce a new objective function that combines reconstruction loss (L_r) and clustering-weighted MSE loss (L_{cmse}): $L = \beta L_r + L_{cmse}$ where β is a trade-off constant. In [19], the auxiliary targets are boosted from the soft assignments in the manner that easier samples would have higher confidence and vice versa. The authors in [34] suggested combining reconstruction loss and K-means-friendly objective function that is formulated in a discrete fashion. The work in [5] incorporates a deep neural network (DNN) at the top of the representation space of an autoencoder to learn an embedded space for the clustering task. A new loss function is derived by minimizing KL divergence between two distributions computed based on probabilistic similarities of pairwise distances [21].

GAN-based methods: A number of works have been exploited GAN framework for handling clustering task as shown in [3], [24], [20], [36]. In GAN [8], the authors proposed to optimize both neural networks G and D by a min-max objective. To be specific, G is a generator model that aims to generate data space from latent vectors, while D is a discrimator that defines a mapping from data space to a real scalar. D can be considered as binary classifier that indicates how much of certainty for a given data sample being real. To train these networks, GAN defines the objective (L_G) as follows:

$$\begin{aligned} L_G &= \min_G \max_D V(D,G) \\ &= \mathbb{E}_{x\sim P_d}[\log D(x)] + \mathbb{E}_{z\sim P_n}[\log(1 - D(G(z)))] \end{aligned} \tag{2}$$

where P_d and P_n denote the data and noise distribution, respectively.

InfoGAN [3] improves GAN by incorporating mutual information (MI) to the loss function. Specifically, InfoGAN maximizes MI between structured latent variables and the observations. To this aim, the generator G takes as input a vector consisting of two parts: noise (z) and latent variables (c). Here, c is incorporated to characterize the semantic features of underlying data which is not taken into consideration in GAN. The objective of InfoGAN (L_I) is derived as follows:

$$\begin{aligned} L_I &= \min_G \max_D V_I(D,G) \\ &= V(D,G) - \lambda I(c; G(z,c)) \end{aligned} \tag{3}$$

where $I(x;y)$ denotes mutual information between two distribution x, y and λ a trade-off constant.

ClusterGAN [24] designs a novel GAN-based architecture with a clustering-specific objective. Firstly, the input of generator G utilizes both continuous and discrete latent variables: $z = (z_n, z_c)$ where z_c is an one-hot encoded vector in R^K with K the number of clusters and $z_n \sim P_n$. Secondly, in addition to the components G and D, the authors added a new neural network E that projects data back to latent space. Finally, the whole system can be jointly optimized with the following objective:

$$\begin{aligned} L_C = &\min_{G,E} \max_D V(D,G) \\ &+ \beta_n \, \mathbb{E}_z \|z_n - E(G(z_n))\|_2^2 + \beta_c \, \mathbb{E}_z \, H(z_c, E(G(z_c)) \end{aligned} \tag{4}$$

where H denote a cross-entropy loss and β_n, β_c are hyper-parameters.

The authors in [20] introduced a new model, namely IAE-ClusterGAN, that improves ClusterGAN in several aspects. They presented an Inverse AutoEncoder (IAE) that is composed of a generator G for mapping latent space to data and an Encoder C for transforming back data to latent space. Clustering will be manipulated at the output of encoder. Furthermore, the attention mechanism is incorporated into both the networks G and C to learn the discriminated features for clustering task. Moreover, the loss distance contributed to the discrimator D (i.e., L_G in Equation 2) is re-formulated in the hypersphere space by using the method of Sphere-GAN [25] that helps to improve training stability.

The last noticeable model is known as VaGAN-GMM [36] that develops a GAN-based architecture within a Variational AE (VAE) framework [15]. VaGAN-GMM shares a common idea with VaDE (Variational Deep Embedding) [14] in that the latent representations of VAE are characterized by a Gaussian mixture model (GMM) consisting of K distributions with K the number of clusters. On the other hand, VaGAN-GMM adds a discriminator that is derived from WGAN-GP (Wasserstein GAN with Gradient Penalty) [9] to handle effectively the problem of unstable training. The authors also adopted the Student's-t mixture model (SMM) as the prior, resulting in a new model so-called VaGAN-SMM. Both the new models perform well on benchmark datasets and are superior to state-of-the-art methods.

3 The proposed clustering model

In this section, we describe the proposed model for learning a discriminative representation to the task of deep clustering. The main objective is targeted to deriving effective loss functions that help to extract crucial features for clustering. Hence, the network architecture is fixed for all experiments and benchmark datasets.

3.1 Network Architecture

The proposed clustering model is designed based on the autoencoder architecture consisting of two main parts: encoder and decoder. The former transforms an input $x \in X$ in the high dimensional space (R^D) into a latent vector $z \in Z$ by a non-linear mapping: $z = f_W(x)$ where W denotes the weights of the mapping, $Z \in R^d$ is a low dimensional space with $d << D$. The latter reconstructs x' from z by also a non-linear function: $x' = g_{W'}(z)$ so that the reconstruction error between x and x' is minimized. The feature space Z is conceptually considered as the latent space, representation layer, embedded space or hidden layer. In the present work, the main objective is concentrated on designing new loss functions for clustering task rather than optimizing the network model. Hence, we have adopted a simple DNN model composing of dense layers only. Specifi-

cally, the network architecture is chosen from the model in DEC [33] for fair evaluation of performance. Both the encoder and decoder consist of four fully connected (FC) layers as follows ($d = 10$):

- Dimensions of encoder (4 FC layers): $D \rightarrow 500 \rightarrow 500 \rightarrow 2000 \rightarrow d$,
- Dimensions of decoder (4 FC layers): $d \rightarrow 2000 \rightarrow 500 \rightarrow 500 \rightarrow D$.

To drive the training process, we derive new objective functions exploiting the pairwise correlations and affinity distances in the probabilistic space. Then, the optimization pipeline of the network model is conducted in two phases: pre-training and fine-tuning. The pre-training phase aims to enhance the effective representation layer Z by combining the correlation loss (L_{cr}) and the reconstruction loss (L_r) as shown in Figure 1. During the fine-tuning phase, the model is further updated as outlined in Figure 2 by using clustering oriented losses to jointly generate the centers of clusters and the model's weights. In the followings, we present these loss functions in accordance with the optimization of network model.

3.2 Enhanced Representation Space

The autoencoder, as shown in Figure 1, aims to learn a representation space Z by a non-linear mapping subject to the constraint of minimal reconstruction error (L_r). Generally, one can use directly the representation space Z as input features for the clustering task but the performance would not be much satisfactory. To enhance the learning capability of the autoencoder, we propose incorporating a new loss function that is derived based on the pairwise distances [21] among data points in original space and representation space.

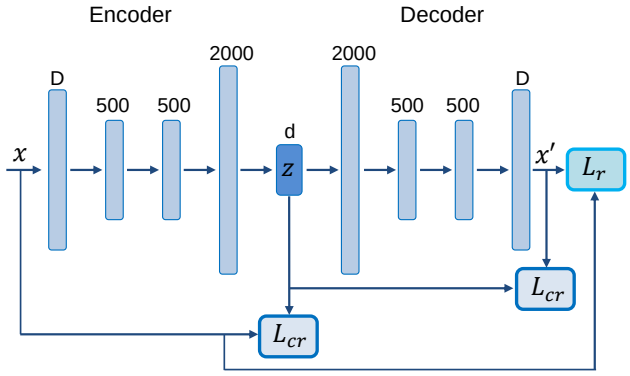

Fig. 1 The architecture of autoencoder model: during the pre-training phase, the correlation loss (L_{cr}) and reconstruction loss (L_r) are incorporated to learn enhanced features.

Denote $x_i, x_j \in X$ two data points in the original space, we define the correlation score $C(x_i, x_j)$ for these two samples by transforming their distance into a probability. To this aim, the Students t-distribution kernel is chosen as follows:

$$C(x_i, x_j) = \frac{(1 + ||x_i - x_j||_2^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k \neq l}(1 + ||x_k - x_l||_2^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{5}$$

where α denotes the degrees of freedom and is set to 1 for all experiments. Intuitively, $C(x_i, x_j)$ can be interpreted as the probability of two samples lying together in the space X. The set of all possible probabilities $C_X = \{C(x_i, x_j)\}$ denotes the correlation distribution in the space X.

When learning the non-linear mapping: $z = f_W(x)$, it is expected that the pairwise correlation $C(z_i, z_j)$ is preserved to capture the local structure of data in the representation space Z, where $z_i = f_W(x_i)$ and $z_j = f_W(x_j)$. More specifically, the autoencoder model is optimized to preserve the pairwise correlations by minimizing the distance between two distributions C_X and C_Z. To this end, we define the correlation loss $L_{cr}(Z, X)$ for the data points in the Z and X spaces based on the KL divergence as follows:

$$\begin{aligned} L_{cr}(Z, X) &= KL(C_Z||C_X) \\ &= \sum_i \sum_j C(z_i, z_j) \log \frac{C(z_i, z_j)}{C(x_i, x_j)}. \end{aligned} \tag{6}$$

Similarly, it is also desired to impose the local structure preservation for the reconstructed data X' from Z by the decoder: $x' = g_{W'}(z)$. As a result, the final loss for pre-training the autoencoder is defined as follows:

$$L_{AE} = \lambda \left[L_{cr}(Z, X) + L_{cr}(Z, X')\right] + L_r(X, X') \tag{7}$$

where L_r is the reconstruction loss defined in Equation 1 and λ is a balanced weight between the loss functions. Training a deep neural network is typically manipulated by the mini-batch Stochastic Gradient Descent (SGD) method where each mini-batch consists of m samples. Hence, the correlation distribution is computed for all pairs of data points within a mini-batch.

3.3 Clustering Oriented Representation

The learned autoencoder that we have presented can produce effective representations for input data. In this fine-tuning stage, the feature space Z can be specifically optimized to handle effectively the problem of deep clustering. This goal can be accomplished by optimizing the network model with clustering oriented loss

functions. We propose exploiting three loss functions to train the clustering model as shown in Figure 2.

Clustering Model: Following the architecture of DEC [33], a cluster layer is attached to the representation layer Z of the autoencoder. The layer has the weights of K centers $\{\mu_j\}_{j=1}^{K}$ in the latent space Z and produces the output of soft assignments ($q_{i,j}$) between a feature vector z_i and a center μ_j by Equation 8:

$$q_{ij} = \frac{(1 + ||z_i - \mu_j||_2^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + ||z_i - \mu_{j'}||_2^2/\alpha)^{-\frac{\alpha+1}{2}}} \tag{8}$$

where α is a parameter and $\alpha = 1$ in all the experiments. For the purppose of presentation, denote $q_i = \{q_{i1}, q_{i2}, \ldots, q_{iK}\}$ the vector of K scalar values, it is straightforward to derive an unit length constraint for q_i from Equation 8 as follows: $\sum_{j=1}^{K} q_{ij} = 1$.

To initialize the weights of clustering model, the input data X is first projected into the representation space Z. Next, K-means algorithm is applied in this space to create K clusters. The obtained centers are assigned as the weights of cluster layer. As the fine-tuning proceeds, the centers and soft assignments are updated based on the response of clustering loss functions.

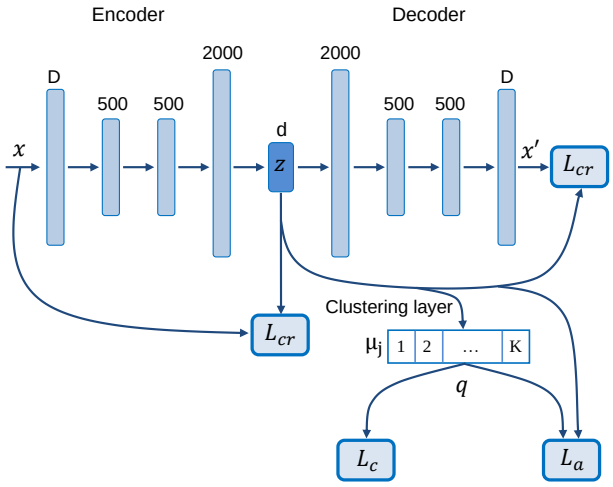

Fig. 2 The architecture of clustering model: a cluster layer is attached to the latent layer Z and the whole model is optimized by using cluster oriented loss functions while embedding regularization loss for local structure preservation.

Clustering Loss: The basic clustering loss function (L_c) presented in [33] is incorporated to drive the training process. As formulated in Equation 9, the function L_c aims to guide the model learning effective representation in such a way that the soft assignment $Q = \{q_{ij}\}$ approaches a target distribution $P = \{p_{ij}\}$:

$$L_c = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{9}$$

where p_{ij} is estimated as follows:

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}} \tag{10}$$

with f_j denote cluster frequencies: $f_j = \sum_i q_{ij}$.

This self-learning strategy starts from easy clusters (i.e., those data points with high assignment scores) to iteratively optimize the network's weights. The learned model then can be expected to handle more effectively the low confidence clusters.

Affinity Loss: We present in this part the affinity loss (L_a) to enhance the representation learning of the model. The new loss is inspired by the observation that if two embedded points z_i and z_t are lying close to each other in the representation space, the corresponding soft assignments (q_i and q_t) are expected to be nearly identical. To this aim, we employ the Students t-distribution kernel to estimate the probability distributions $C_Z = \{C(z_i, z_t)\}$ and $C_Q = \{C(q_i, q_t)\}$ as conducted in the correlation loss. KL divergence is then utilized to measure the distance between two distributions where C_Q plays the role of target labels and is computed in a similar manner to $P = \{p_{ij}\}$ in Equation 10. Formally, the affinity loss is defined as follows:

$$L_a = KL(C_Q||C_Z) = \sum_i \sum_t C(q_i, q_t) \log \frac{C(q_i, q_t)}{C(z_i, z_t)}. \tag{11}$$

Model Regularization: When training the model with clustering oriented losses, one may arrive in a circumstance that the learned weights of encoder and decoder could be distorted. As a result, this could weaken the effectiveness of the feature representation space Z and cause a drop in clustering quality. To alleviate this issue, one may employ the reconstruction loss as shown in [10], [5], [27]. Instead, as indicated in Figure 2, we propose incorporating the correlation loss L_{cr} at two levels: the latent layer Z and decoder's output layer. To be specific, the regularization loss is formulated as follows:

$$L_{re} = L_{cr}(Z, X) + L_{cr}(Z, X') \tag{12}$$

The integration of this loss helps to preserve the local structure of data and hence offering effective clustering performance as will be shown in our experiments.

Putting all materials aforementioned together, the final loss (L_{DC}) for optimizing the deep clustering model can be derived as follows:

$$L_{DC} = L_c + \eta L_a + \beta L_{re} \tag{13}$$

where η, β denote the hyper-parameters to balance the contribution of loss functions.

3.4 Optimization and Training

The model is optimized in an end-to-end fashion by two phases: pre-training and fine-tuning. In the first phase, the model is trained by using the loss function L_{AE} in Equation 7. After that, the input data are projected into the representation layer, followed by K-means algorithm to produce K clusters. The model's weights and K centers are further optimized in the fine-tuning phase by using the objective L_{DC} in Equation 13. Algorithm 1 presents the optimization and training process of our clustering model. Note that the auxiliary target distribution p_{ij} is updated after every h iterations [10].

Training Settings: The model is trained and evaluated on a moderate GPU configuration: GeForce RTX 2080 Ti, 11GB memory. The training hyper-parameters are set based on the suggestions reported in previous work [10] and our observation for studied datasets. The fine-tuning phase is conducted by the SGD optimization method with momentum of 0.9 and learning rate at 0.01. The batch size is set to 64, $h \in \{50, 80\}$, and $T = 5000$.

Algorithm 1 Clustering Model Training

Input: Input data X, number of clusters K, number of iteration T, batch size m, and the number of iterations (h) for updating target distribution.
Output: The clustering model and K cluster centers.

Pre-train the autoencoder model $(f_W, g_{W'})$ by the loss function in Equation 7.
Project input data X into representation space: $Z \leftarrow f_W(X)$.
Apply K-means on Z to yield K cluster centers: $\{\mu_j\}_{j=1}^{K}$.
Initialize the weights of clustering layer by $\{\mu_j\}$.
for $t \in \{1, 2, \ldots, T\}$ **do**
  if $t\%h == 0$ **then**
    Compute $Z \leftarrow f_W(X)$.
    Compute q_{ij} by using (8).
    Update p_{ij} by using (10).
  end if
  Select a random batch S consisting of m samples.
  Update μ on S by using: $L_c + \eta L_a$.
  Update W' on S using L_{AE} in (7).
  Update W on S by using L_{DC} in (13).
end for
return $(f_W, g_{W'}, \mu)$

4 Experiments

4.1 Datasets and Evaluation Protocols

In the literature, most of deep clustering methods are validated by using classification benchmark datasets including: Reuters [18], Fashion-MNIST (or FMNIST) [32], CIFAR-10 [16], and STL-10 [4]. For objective evaluation, we also selected these datasests for all of our experiments. However, due to memory limitation of our GPU machine, it is not possible to handle the whole CIFAR-10 dataset composing of 60,000 images. Hence, we randomly sampled 30,000 images for evaluation. For Reuters dataset, as it is a large dataset to process, other works [10], [30] proposed to randomly select a subset of 10,000 samples, resulting in a Reuters-10K dataset. Specifically, we have selected the Reuters-10K dataset that have been used for comparative evaluation in the work EDESC [1] at this host[1]. Furthermore, as our model consists of dense layers only, some datasets are needed to be pre-processed for appropriate input representation. Following the work in [1], the ResNet50 [12] model is applied to several datasets (CIFAR-10, STL-10) to extract 2048-dimensional (2048D) feature vectors. Detailed information of the datasets are presented in Table 1.

Table 1 The datasets used in our experiments

Dataset	# Samples	# Classes	# Feature Size
Reuters-10K	10,000	4	2000
FMNIST	70,000	10	784
CIFAR-10	30,000	10	2048
STL-10	13,000	10	2048

For evaluation metrics, we have selected two scores: Clustering Accuracy (ACC) and Normalized Mutual Information (NMI) as they have been widely employed in the literature for evaluating performance of clustering methods. Mathematically, ACC is computed by:

$$ACC = \max_{m} \frac{\sum_{i=1}^{n} 1\{l_i = m(c_i)\}}{n}, \tag{14}$$

where n is the number of data samples, l_i and c_i denote the ground-truth label and predicted cluster, and m denotes the one-to-one mapping between cluster assignment and true label. Additionally, NMI is defined as follows:

$$NMI(l, c) = \frac{I(l, c)}{\frac{1}{2}[H(l) + H(c)]} \tag{15}$$

where $I(l, c)$ is the mutual information between cluster assignment and true label, and H denotes the entropy. These two metrics have the values in the range of [0, 1] with the meaning that higher score demonstrates better clustering performance.

[1] https://github.com/JinyuCai95/EDESC-pytorch

Table 2 Clustering performance ACC and NMI (in bracket) on four datasets. The best results are marked in bold.

Model/Dataset	Reuters-10K	Fasion-MNIST	STL-10	CIFAR-10
EDESC	82.5 (61.1)	63.1 (**67.0**)	74.5 (68.7)	62.7 (46.4)
ClusterGAN	67.76 (40.6)	63.0 (64.0)	-	33.5 (28.2)
DEPICT	63.1 (36.96)	61.0 (60.1)	-	-
IAE-ClusterGAN	-	66.2 (63.5)	-	40.2 (33.6)
VaGAN-GMM	80.1 (53.6)	63.8 (63.3)	-	28.7 (15.8)
DAC	-	61.5 (63.2)	47.0 (36.6)	52.2 (39.6)
DGC	76.39 (59.8)	**68.2** (64.19)	-	-
JULE	62.6 (40.5)	56.3 (60.8)	27.7 (18.2)	27.2 (19.2)
Our CDC (mean)	82.6 (60.7)	64.9 (64.5)	96.3 (91.8)	79.2 (68.5)
Our CDC (median)	**83.9** (**61.9**)	66.0 (64.6)	**96.5** (**92.0**)	**79.7** (**68.6**)

Finally, the most representative methods for clustering have been chosen to conduct comparative evaluation. The selected methods include the followings: DEPICT [6], ClusterGAN [24], IAE-ClusterGAN [20], EDESC [1], VaGAN-GMM [36], DAC [2], DGC [30] and JULE [35]. For objective evaluation, the clustering performance of these methods have been reported from previous works. Specifically, the results on the STL-10 and CIFAR-10 datasets are reused from [20] and [1], while most of other metrics are reproduced from [30]. Several methods are not applicable for specific datasets and hence the corresponding scores are marked by a minus symbol. The results of our model (namely CDC[2]) for each dataset are computed as the mean and median performance over 10 random running trials. The source code of our model is developed from this address[3] and is publicly available at this host[4].

4.2 Comparative Results

Table 2 shows the clustering performance (ACC and NMI) of all the studied methods on four datasets. As can be shown, the proposed clustering model performs impressively on all the datasets. Interestingly, our CDC model gives outstanding results in the terms of both ACC and NMI scores on the two datasets: STL-10 and CIFAR-10. Taking the STL-10 dataset for example, we obtain an accuracy ACC=96.5% and NMI = 92.0% when compared to the second place winner (i.e., EDESC) having ACC=74.5% and NMI = 68.7%. The same impressive performance of our method can be also consistently observed on the Reuters-10K dataset with ACC (NMI) = 83.9% (61.9%) and on the CIFAR-10 dataset with ACC (NMI) = 79.7% (68.8%). For the other dataset (Fasion-MNIST), the proposed system is quite competitive to the best methods (i.e., EDESC and DGC) with our performance: ACC (NMI) = 66.0% (64.6%), while still outperforming many other clustering models. It is worth noting that the input features of Fasion-MNIST for all the clustering models are raw pixel data. We do not apply any pre-trained DNN model to extract features as done with STL-10 or CIFAR-10. These results demonstrate the important role of our objective functions for different types of datasets.

[2] Correlation losses for Deep Clustering
[3] https://github.com/XifengGuo/DEC-keras
[4] https://github.com/fict-labs/Deep-Clustering-CDC

4.3 Visualization Results

In this part, we visually investigate the clustering quality of our model for all the datasets. To this aim, the visualization method t-SNE [22] is employed for each dataset by using three levels of features: the input features, the embedded features of pre-trained autoencoder, and the representation features after fine-tuning the clustering model. The results are presented in Figure 3. As can be observed, the final clustering structures (the right column) of four datasets are clearly revealed after the fine-tuning phase. For the STL-10 dataset (the first row), the input data are 2048D feature vectors, as shown in Figure 3(a), extracted from the ResNet50 model. Although the clusters are already visible to some extent, the class boundaries are not very separated. After transforming the input data into the embedded layer by using the pre-trained autoencoder, as depicted in Figure 3(b), the decision boundaries have been further improved. Impressively, after applying the fine-tuning phase, the learned representations are very discriminative. As shown in Figure 3(c), the cluster boundaries are separated with a large margin degree. This corresponds to an impressive accuracy of the proposed clustering model (ACC = 96.5%).

The same reports can be also observed for the other datasets with specific remarks. Taking the CIFAR-10 dataset for instance, the final cluster structures shown in Figure 3(f) are greatly improved when comparing to that of Figure 3(d). The clusters are well separated from

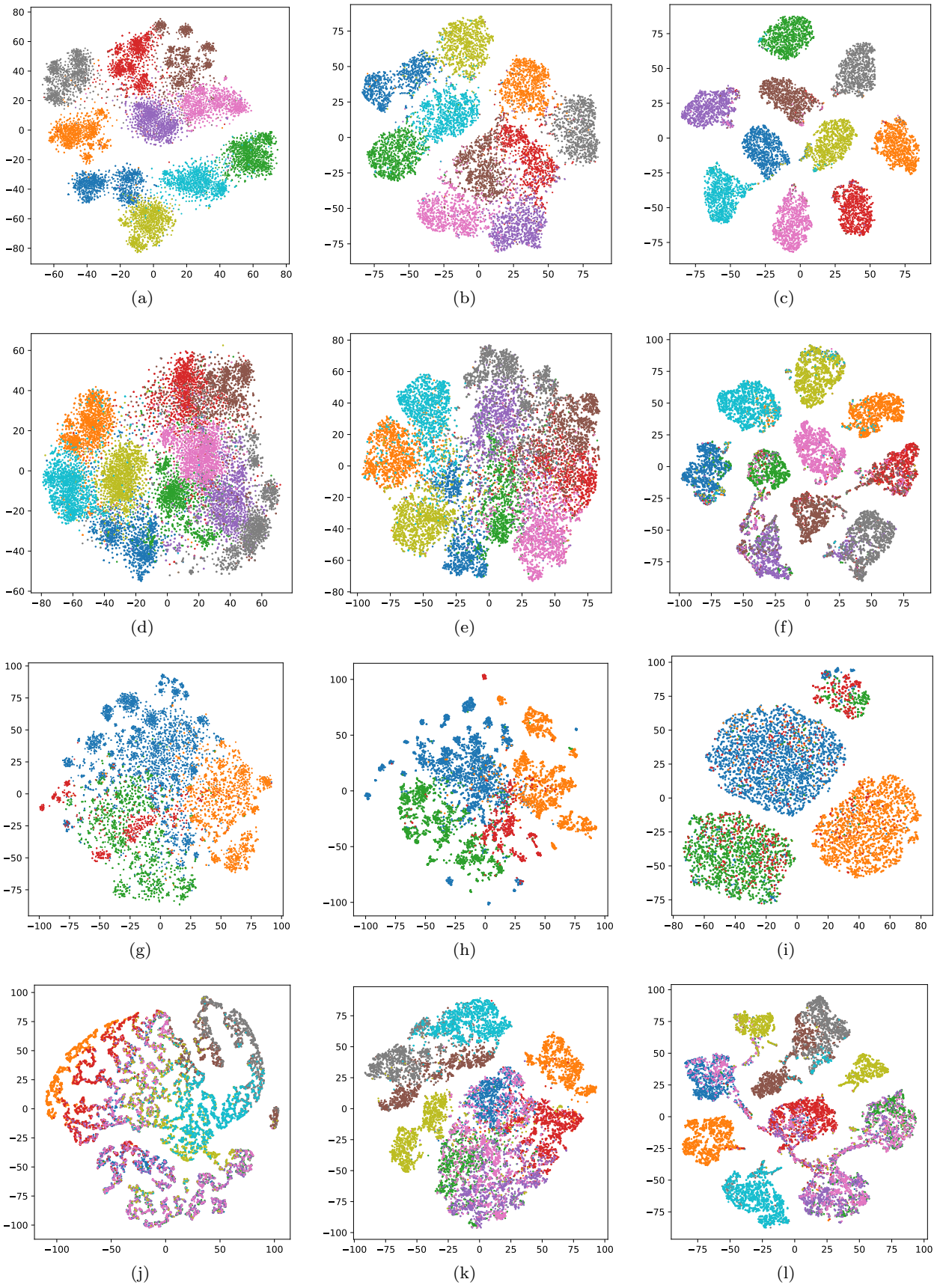

Fig. 3 Visualization of learned features using t-SNE: STL-10 (first row), CIFAR-10 (second row), Reuters-10K (third row), and Fashion-MNIST (bottom row). From left to right: input features, autoencoder's output, and clustering model's output.

each other, yielding an accuracy of about 79.2%. Differing from the above datasets, there is an imbalanced class problem for the textual Reuters-10K dataset as shown in Figure 3(g)-(i): the smallest cluster has 895 data points while the largest has 4,022 examples. As a consequence, this problem causes a drop in clustering performance. Nonetheless, our model still achieves an interesting accuracy of 83.9%. The last dataset (Fasion-MNIST) is a really challenge for all methods as demonstrated in Figure 3(j)-(l). Similar to the best methods (e.g., DGC and EDESC), the proposed method successfully detects several clusters but not all. Figure 3(l) also reveals that several clusters do not have a spherical shape. This suggests that using an appropriate clustering algorithm (e.g., DBSCAN [7]) on the learned representations would produce a better clustering accuracy.

4.4 Ablation Study

In this part, different ablation experiments are conducted to study the behavior of the proposed clustering model. To study the improvement in clustering accuracy, Figure 4 shows the accuracy curves of four datasets during the fine-tuning phase. As we can see, the accuracy scores are stably improved as the training proceeds. For several datasets, the proposed model gains large degrees of accuracy (e.g., approximately 9% for STL-10 and 4% for CIFAR-10 or Fasion-MNIST). Another noticeable point could be mentioned from Figure 4 that the model converges quickly after 2,000 iterations on four datasets.

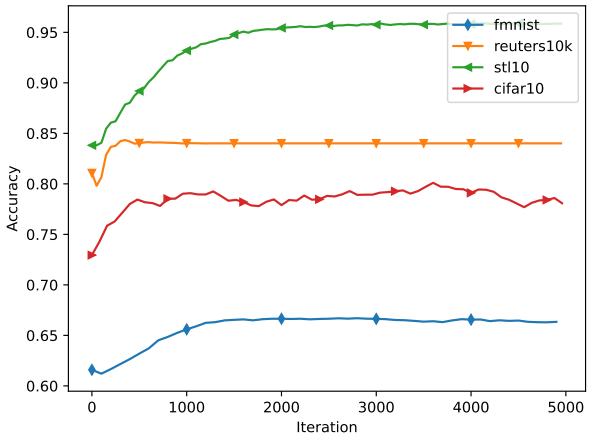

Fig. 4 Accuracy scores on four datasets during the fine-tuning phase of our model.

Figure 5 shows the stability of clustering performance over 10 random running trials. One can observe that this behavior is highly attributed to a specific data distribution. Particularly, our clustering model is shown to be very robust to the STL-10 dataset with mostly constant clustering accuracy of around 96% over all running trials. For the other datasets, the clustering accuracy variability is varying in the range of $[4\%, 12\%]$ but the model generally produces pertinent accuracy scores over different initializations. To have more insight for a few cases of dropped performance, it was found out that the initialization for pre-training the autoencoder model is an important factor. Taking the Reuters-10K dataset for example, the lower accuracy score of clustering model (ACC $= 72.7\%$) at the running trial ID of 3 is associated to the dropped performance of pre-trained autoencoder model (ACC $= 71.1\%$). This remark suggests that there is a room for optimization of model initialization in the pre-training phase.

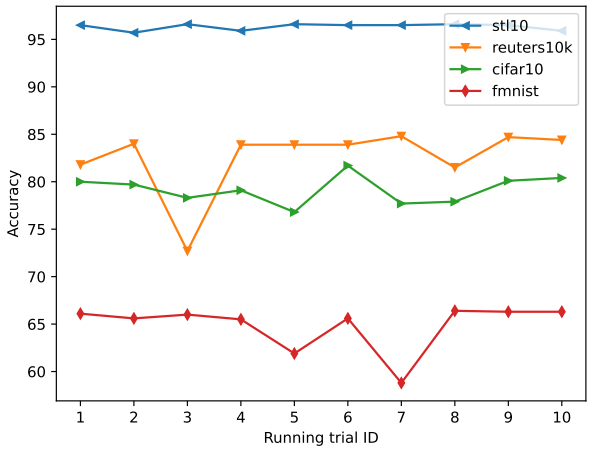

Fig. 5 Accuracy scores on four datasets of our model for random running trials.

One of important ablation studies in deep clustering is concerned with the impact of objective functions. To this aim, we present in Table 3 the clustering performance in the terms of average ACC and NMI with respect to different instances of loss functions. For the pre-training phase, it can be observed that the autoencoder (AE) model, that is trained by using our loss (AE* = AE$+L_{AE}$), generally outperforms the model trained with the standard reconstruction loss (AE$+L_r$) on four datasets. For example, the improvement gaps in clustering accuracy of AE* are around 10.9% and 4.3% on the STL-10 and Reuters-10K datasets, respectively.

As for the fine-tuning phase, we investigate the impact of using the basic clustering loss (AE$^*+L_c$), affinity loss (AE$^*+L_c+L_a$), and all the losses (AE$^*+L_{DC}$). As expected, the clustering results are consistently improved when incorporating more loss functions. The full model (AE$^*+L_{DC}$) enhances the improvements in

Table 3 Impact of loss functions in term of ACC and NMI (in bracket) of our model.

Dataset/Loss	AE + L_r	AE +L_{AE}	AE* +L_c	AE* +$L_c + L_a$	AE* +L_{DC}
Reuters-10K	75.2 (51.2)	79.5 (54.9)	77.8 (58.5)	81.5 (60.7)	82.6 (60.7)
Fashion-MNIST	56.4 (59.0)	60.1 (58.1)	61.1 (64.0)	63.1 (64.3)	64.9 (64.5)
STL-10	75.4 (68.5)	86.3 (78.0)	95.0 (90.2)	95.7 (91.3)	96.3 (91.8)
CIFAR-10	75.5 (62.4)	75.6 (62.7)	78.4 (67.1)	79.1 (68.5)	79.2 (68.5)

both metrics of ACC and NMI with the performance increasing up to 10% when comparing to the accuracy of pre-trained model (AE*) on the STL-10 dataset. Note that for Reuters-10K dataset, the model employing only the clustering loss (AE* +L_c) gives negative effect for clustering accuracy. The source of this dropped performance could be explained by the distortion of local data structures. Interestingly, the models, integrating our new loss functions: AE* +$L_c + L_a$ and AE* +L_{DC}, produce significant improvements with accuracy scores of 81.5% and 82.6%, respectively.

For other hyper-parameters, the balanced weights η, β, and λ are tuned for each dataset by using a grid search in the list of $\{10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}, 0.5, 0.75, 1.0\}$. Interestingly, it was found out that their values are fairly converged to the following configuration: $\eta = 0.75, \beta = 0.001$ for all the studied datasets, $\lambda = 10^{-6}$ for CIFAR-10, and $\lambda = 0.001$ for the others. As conclusion, the proposed loss functions, weighted by their corresponding contributions, help the model explore more latent representations and capture features that are conductive and discriminative to clustering task.

5 Conclusion

In this work, an advanced deep autoencoder model is presented to learn clustering-specific features. The main target is to design distinctive loss functions to handle effectively the task of data clustering. Specifically, we derive a correlation loss by using probabilistic similarities of pairwise distances. This loss is incorporated at several layers of the model to preserve local structures of underlying data. In addition, a novel loss function based on affinity distances in the probabilistic space is presented to learn important clustering features. Moreover, the model employs the basis KL distance to estimate the loss divergence between a predicted clustering distribution and an auxiliary target space. This self-learning based fashion plays the primary role of supervision to model training. Comprehensive experiments on four benchmark datasets have been conducted to justify the clustering performance of the proposed model. The experimental results showed that our clustering model is highly competitive with the state-of-the-art methods and produces outstanding clustering performance.

Nonetheless the obtained results are encouraging, there are several interesting aspects to be more investigated. Firstly, it is helpful to give deeper focus on optimization of initial estimate of the autoencoder in the pre-training phase. Secondly, one can employ a semi-supervised learning fashion, that utilizes a small part of true labels, to accomplish better clustering effect. Finally, the incorporation of more advanced networks (e.g., attention mechanism [28], [31]) would help to improve clustering accuracy.

Declarations:

– Data Availability Statement: Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.
– Conflict of Interest: The authors declare that they have no conflict of interest.
– Funding: No funding is provided for this work.

References

1. Cai, J., Fan, J., Guo, W., Wang, S., Zhang, Y., Zhang, Z.: Efficient deep embedded subspace clustering. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 21–30 (2022)
2. Chang, J., Wang, L., Meng, G., Xiang, S., Pan, C.: Deep adaptive image clustering. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5880–5888 (2017)
3. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: 30th Conference on Neural Information Processing Systems (NIPS 2016), pp. 1–9 (2016)
4. Coates, A., Ng, A.Y., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: In International Conference on Artificial Intelligence and Statistics, pp. 215–223 (2011)
5. Dahal, P.: Learning embedding space for clustering from deep representations. In: 2018 IEEE International Conference on Big Data (Big Data), pp. 3747–3755 (2018)
6. Dizaji, K.G., Herandi, A., Deng, C., Cai, W., Huang, H.: Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5747–5756 (2017)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial

databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD'96, pp. 226–231. AAAI Press (1996)
8. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengiob, Y.: Generative adversarial networks. In: Advances in Neural Information Processing Systems 27 (NIPS 2014), pp. 2672–2680 (2014)
9. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein gans. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 5769–5779 (2017)
10. Guo, X., Gao, L., Liu, X., Yin, J.: Improved deep embedded clustering with local structure preservation. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), pp. 1753–1759 (2017)
11. Guo, X., Zhu, E., Liu, X., Yin, J.: Deep embedded clustering with data augmentation. In: The 10th Asian Conference on Machine Learning (ACML), vol. 95, pp. 550–565 (2018)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
13. Jegou, H., Douze, M., Schmid, C.: Product Quantization for Nearest Neighbor Search. IEEE Trans. Pattern Anal. Mach. Intell. **33**(1), 117–128 (2011)
14. Jiang, Z., Zheng, Y., Tan, H., Tang, B., Zhou, H.: Variational deep embedding: an unsupervised and generative approach to clustering. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence, pp. 1965–1972 (2017)
15. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. CoRR **abs/1312.6114** (2013). URL https://api.semanticscholar.org/CorpusID:216078090
16. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report (2009). URL https://www.cs.toronto.edu/~kriz/cifar.html
17. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998). DOI 10.1109/5.726791
18. Lewis, D.D., Yang, Y., Rose, T.G., Li, F.: Rcv1: A new benchmark collection for text categorization research. The Journal of Machine Learning Research **5**, 361–397 (2004)
19. Li, F., Qiao, H., Zhang, B.: Discriminatively boosted image clustering with fully convolutional auto-encoders. Pattern Recognition **83**, 161–173 (2018)
20. Ling, C., Cao, G., Cao, W., Wang, H., Ren, H.: Iae-clustergan: A new inverse autoencoder for generative adversarial attention clustering network. Neurocomputing **465**, 406–416 (2021)
21. van der Maaten, L.: Learning a parametric embedding by preserving local structure. In: Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics, *Proceedings of Machine Learning Research*, vol. 5, pp. 384–391 (2009)
22. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
23. Macqueen, J.B.: Some methods for classification and analysis of multivariate observations. In: In 5-th Berkeley Symposium on Mathematical Statistics and Probability, pp. 281–297 (1967)
24. Mukherjee, S., Asnani, H., Lin, E., Kannan, S.: Clustergan: latent space clustering in generative adversarial networks. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, pp. 4610–4617 (2019)
25. Park, S.W., Kwon, J.: Spheregan: Sphere generative adversarial network based on geometric moment matching and its applications. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(3), 1566–1580 (2020)
26. Pham, T.A., Le, D.N., Nguyen, T.L.P.: Product subvector quantization for feature indexing. Journal of Computer Science and Cybernetics **35**(1), 161–173 (2019)
27. Si, L., Ruisi, L.: Dac–deep autoencoder-based clustering: A general deep learning framework of representation learning. In: Intelligent Systems and Applications, pp. 205–216 (2022)
28. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6450–6458 (2017)
29. Wang, W., Bao, J., Guo, S.: Neural generative model for clustering by separating particularity and commonality. Information Sciences **589**, 813–826 (2022)
30. Wang, W., Bao, J., Guo, S.: Neural generative model for clustering by separating particularity and commonality. Information Sciences **589**, 813–826 (2022)
31. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 1–17 (2018)
32. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv e-prints arXiv:1708.07747 (2017). DOI 10.48550/arXiv.1708.07747
33. Xie, J., Girshick, R., Farhadi, A.: Unsupervised deep embedding for clustering analysis. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, pp. 478–487 (2016)
34. Yang, B., Fu, X., Sidiropoulos, N., Hong, M.: Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In: Proceedings of the 34th International Conference on Machine Learning, pp. 3861–3870 (2017)
35. Yang, J., Parikh, D., Batra, D.: Joint unsupervised learning of deep representations and image clusters. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5147–5156 (2016)
36. Yang, L., Fan, W., Bouguila, N.: Clustering analysis via deep generative models with mixture models. IEEE Transactions on Neural Networks and Learning Systems **33**(1), 340–350 (2022)