Applied Intelligence
https://doi.org/10.1007/s10489-021-02569-y

Deep convolutional self-paced clustering

Rui Chen[1,2] · Yongqiang Tang[2] · Lei Tian[2,3] · Caixia Zhang[1] · Wensheng Zhang[2,3]

Accepted: 25 May 2021

Abstract
Clustering is a crucial but challenging task in data mining and machine learning. Recently, deep clustering, which derives inspiration primarily from deep learning approaches, has achieved state-of-the-art performance in various applications and attracted considerable attention. Nevertheless, most of these approaches fail to effectively learn informative cluster-oriented features for data with spatial correlation structure, e.g., images. To tackle this problem, in this paper, we develop a deep convolutional self-paced clustering (DCSPC) method. Specifically, in the pretraining stage, we propose to utilize a convolutional autoencoder to extract a high-quality data representation that contains the spatial correlation information. Then, in the finetuning stage, a clustering loss is directly imposed on the learned features to jointly perform feature refinement and cluster assignment. We retain the decoder to avoid the feature space being distorted by the clustering loss. To stabilize the training process of the whole network, we further introduce a self-paced learning mechanism and select the most confident samples in each iteration. Through comprehensive experiments on seven popular image datasets, we demonstrate that the proposed algorithm can consistently outperform state-of-the-art rivals.

Keywords Deep clustering · Convolutional autoencoder · Local structure preservation · Self-paced learning

1 Introduction

Clustering aims to partition data into different groups, where samples in the same group are more similar to each other than to those in other groups. Over the past several decades, numerous clustering methods have been exploited in diverse real-world applications, e.g., image classification [1–4] and data visualization. Conventional clustering methods [5–7] have high speed and are suitable for a wide range of problems. Despite their success in data clustering, these methods usually depend on predefined similarity measurements, which are subject to the original data space and tend to be ineffective when the input dimensionality is relatively high.

To remedy the above issues, many works attempt to borrow the advantages of dimensionality reduction techniques to map raw data into a new low-dimensional feature space [8–12]. For instance, principal component analysis (PCA) [8] aims to learn a linear projection and maximizes the data variance in the projected low-dimensional space. Nevertheless, such a simple linear transformation makes PCA unable to depict the extremely complex latent structure underlying the data. Recently, Peng et al. [11] have proposed to utilize the tensor similarity to capture complementary information that a pairwise similarity fails to provide, i.e., the structural information, achieving excellent clustering performance. In contrast to the above traditional methods, deep neural networks (DNNs) have shown much greater power to extract high-level features by nonlinear embedding. Among existing DNNs, the autoencoder is a favorable method that first embeds the data into a latent feature space and then attempts to reconstruct the input data based on this space. The past few years have witnessed many related achievements [27, 28]. Peng et al. [27] integrate prior sparsity information into the middle layer of an autoencoder to simultaneously adapt the local and global subspace structure. Ji et al. [28]

✉ Yongqiang Tang
yongqiang.tang@ia.ac.cn

✉ Caixia Zhang
zh_caixia@163.com

[1] Department of Automation, Foshan University, Foshan, China

[2] Research Center of Precision Sensing and Control, Institute of Automation, Chinese Academy of Sciences, Beijing, China

[3] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China

Published online: 29 July 2021

incorporate a differentiable self-expressive layer into an autoencoder to learn pairwise relationships between data points. Although the improved clustering performance is achieved by the autoencoder, it generally treats the processes of feature learning and pseudolabel assignment as two separate steps, rendering these two processes incapable to benefit from each other.

More recently, several works [32–34, 41] have attempted to integrate autoencoder and data clustering into a unified framework, such that the representations of data points as well as their corresponding cluster assignments can be optimized jointly to achieve superior performance. For example, Xie et al. [32] utilize an autoencoder to dig out the cluster-oriented feature representations for input data, which achieves a substantial improvement compared with traditional clustering approaches. Guo et al. [33] further propose to merge the reconstruction loss of an autoencoder into DEC's [32] objective, achieving appreciable improvement. Yang et al. [34] combine autoencoder-based dimensionality reduction and K-means [5] clustering into a joint learning framework to improve the performance of both, which requires an alternative optimization strategy to discretely update cluster centers, cluster assignments and network parameters. Huang et al. [41] propose to jointly extract distinctive features via autoencoder and perform subspace learning to guide the clustering task.

Despite these excellent efforts, the existing deep clustering methods still present three drawbacks: First, the most widely used autoencoder in deep clustering algorithms is the stacked autoencoder (SAE) [25], which requires layer-wise training before being tuned. However, this procedure may be time-consuming and redundant as the layers go deeper. Furthermore, SAE is built with fully connected layers, which is ineffective for dealing with data with spatial correlation structures such as images. Second, one more issue deserving our attention is that most current works typically ignore the preservation of data properties, which can result in the corruption of feature space and hinder the clustering performance accordingly. Third, previous approaches fail to take the effect of marginal samples into account during network training. Although near the cluster boundaries, these marginal examples may not provide convincing guidance since clustering is performed in an unsupervised manner. As a result, as illustrated in [37], unreliable samples could confuse or even mislead the training process of the DNN and thus seriously degrade the clustering performance.

Aiming to tackle the above three issues simultaneously, in this paper, we propose a Deep Convolutional Self-Paced Clustering (DCSPC) method. Specifically, our approach embodies two stages: pretraining and finetuning. In the pretraining stage, we train a convolutional autoencoder (CAE) [26] by minimizing the reconstruction loss in an end-to-end manner to learn high-quality features. By employing CAE, our approach can transform data from a relatively high-dimensional and sparse space to a low-dimensional and compact space. Then, in the finetuning stage, different from some previous works [31, 32, 37] that only retain the encoder, we tune the whole autoencoder (or specifically CAE) by using both clustering loss and reconstruction loss, such that data properties can be preserved to avoid the corruption of feature space. In addition, to eliminate the influence of marginal samples, we introduce a self-paced learning mechanism to gradually select training samples from easy to hard. To summarize, the main contributions of this work are highlighted as follows:

- We propose a novel deep clustering method called DCSPC. Our DCSPC employs a convolutional autoencoder to capture the spatial relationship and avoids the corruption of feature space by tuning both clustering loss and reconstruction loss, such that high-quality cluster-oriented features can be learned.
- In the finetuning stage, we further introduce a self-paced learning mechanism into our DCSPC. The samples are involved in the optimization process from easy to hard, such that adverse effects of marginal samples can be mitigated effectively.
- Extensive experiments on seven widely used benchmark image datasets demonstrate the superiority of our DCSPC against the state-of-the-art deep clustering methods. An elaborate ablation study confirms the effectiveness of our proposals.

The remainder of this paper is organized as follows. In Section 2, we present a brief review of the related work. Section 3 describes the details of the developed DCSPC algorithm. Comprehensive experimental results are reported and analyzed in Section 4. Finally, Section 5 concludes this paper.

2 Related work

This section reviews some of the previous work closely related to this paper. We first briefly review the classical K-means algorithm. Next, related studies of deep unsupervised clustering are reviewed. Finally, we introduce the self-paced learning paradigm.

2.1 *K*-means algorithm

Given a set of data $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathbb{R}^{D\times n}$, the goal of clustering is to divide the n samples into K groups, forcing samples within the identical groups to be more similar to each other than to those in other groups. Among existing works, prototype-based clustering [5, 15]

has attracted much attention. In this paper, we focus on the commonly used prototype-based clustering method, i.e., K-means algorithm, whose optimization problem can be stated as follows:

$$\begin{aligned} &\min_{\mathbf{M},\mathbf{s}} \frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - \mathbf{M}\mathbf{s}_i\|_2^2, \\ &s.t.\ \mathbf{s}_i \in \{0,1\}^K,\ \mathbf{1}^{\mathrm{T}}\mathbf{s}_i = 1, \end{aligned} \tag{1}$$

where $\mathbf{s}_i \in \mathbb{R}^K$ is the assignment vector of data point $\mathbf{x}_i$, which has only one nonzero element, and the jth column of $\mathbf{M} \in \mathbb{R}^{D\times K}$, i.e., m_j, denotes the centroid of the jth cluster. When the data points are evenly scattered around their corresponding centroids in the feature space, the K-means algorithm works very effectively. However, K-means is usually not suitable for high-dimensional data because of the inefficient similarity measurement caused by the "curse of dimensionality". Thus, in practice, we should use dimensionality reduction methods, such as PCA [8], MDS [9] and NMF [10], to project the original data onto low-dimensional space and then employ the K-means algorithm to cluster the low-dimensional data, which will usually bring about better results. In addition to the above linear dimensionality reduction approaches, nonlinear algorithms such as tSNE [17], LLE [18], and DNN-based methods [19–21] are widely used for preprocessing before the K-means algorithm. We refer interested readers to [22–24] for a comprehensive understanding. In many real-world applications, data may be derived from different views, and thus, a number of multiview clustering methods have been proposed. For example, Zhang et al. [13] first map the multiview samples to a shared view space, then convert the samples to a discriminative space, and finally conduct K-means to cluster the converted samples. Wang et al. [14] propose a general graph-based multiview clustering framework, which generates a unified graph matrix by extracting feature matrices and fusing graph matrices from multiple views for direct clustering. Considering that there may be cases in which a specific class is not in the training data, Hayashi et al. [16] propose a cluster-based zero-shot learning method to divide the data into invisible and visible classes.

2.2 Deep unsupervised clustering

Deep unsupervised clustering methods can be roughly divided into two categories. One category includes approaches that usually treat feature learning or clustering independently, i.e., project the raw data into a low-dimensional feature space first, and then conduct conventional clustering algorithms to group feature points. Unfortunately, this kind of separated form can place restrictions on the clustering performance because of the neglect of some potential relationships between feature learning and clustering.

Another category refers to methods that use the joint optimization criterion, which perform both feature learning and clustering simultaneously and have shown great superiority beyond the separated methods. Recently, several approaches have been proposed to integrate feature learning and clustering into a unified framework. Joint unsupervised learning (JULE) [29] proposes to guide the agglomerative clustering and representation learning concurrently on the basis of a unified weighted triplet loss, but it involves relatively high computational complexity. Chang et al. [30] raise a hypothesis of the binary relationship between pairwise images and develop a deep adaptive clustering (DAC) model to reestablish the clustering task as a binary pairwise-classification problem, showing excellent results on six image datasets. Drawing lessons from hard-weighting self-paced learning, adaptive self-paced clustering (ASPC) [37] prioritizes high-confidence samples during the clustering network training to eliminate the negative effect of marginal samples and stabilize the training process. Ren et al. [40] propose a deep density-based clustering (DDC) technique, which can adaptively estimate the number of data clusters with arbitrary shapes. Deep embedded clustering with data augmentation (DEC-DA) [36] incorporates the data augmentation trick into the original deep embedded clustering framework and achieves excellent clustering performance on four grayscale image datasets. Semisupervised deep embedded clustering (SDEC) [39] overcomes the drawback in that DEC [32] fails to guide the training process by taking advantage of prior knowledge.

2.3 Self-paced learning

Similar to the core idea of curriculum learning [43], the goal of self-paced learning is to learn a model by gradually involving samples for training from easy to hard. The obvious difference between these approaches is that the former requires a predetermined prior of easy and hard samples, whereas the latter can automatically select the order from the data themselves. Given one training set $\mathbf{X} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ and training model f_θ with θ being the model parameters, the general objective of self-paced learning can be stated as follows:

$$\begin{aligned} &\min_{\theta,\mathbf{v}} \frac{1}{n}\sum_{i=1}^{n}\mathbf{v}_i L(f_\theta(\mathbf{x}_i), \mathbf{y}_i) + h(\lambda, \mathbf{v}_i), \\ &s.t.\ \mathbf{v}_i \in [0,1], \end{aligned} \tag{2}$$

where $L(\cdot)$ denotes the loss function of a specific problem, $h(\lambda, \mathbf{v}_i)$ represents a self-paced regularizer that is independent of $L(\cdot)$ and can be defined in various forms,

$\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]^{\mathrm{T}}$ symbolizes the weight variable reflecting the complexity of samples, and λ is a parameter, called learning pace, for controlling the "model age" which gradually increases in order to explore more samples. When $h(\lambda, \mathbf{v}_i) = -\lambda \mathbf{v}_i$ with $\mathbf{v}_i$ equals 0 or 1, self-paced learning degenerates into the hard-weighting form, i.e.,

$$\min_{\theta, \mathbf{v}} \frac{1}{n} \sum_{i=1}^{n} \mathbf{v}_i L(f_\theta(\mathbf{x}_i), \mathbf{y}_i) - \lambda \mathbf{v}_i, \quad s.t.\ \mathbf{v}_i \in \{0, 1\}. \tag{3}$$

With θ fixed, the optimal $\mathbf{v}_i$ of problem (3) is determined by:

$$\mathbf{v}_i = \begin{cases} 1, & \text{if } L(f_\theta(\mathbf{x}_i), \mathbf{y}_i) \leq \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Additionally, for updating θ with fixed $\mathbf{v}$, problem (3) degenerates into a weighted loss minimization problem, which can be readily settled via stochastic gradient descent (SGD) and backpropagation (BP).

To date, self-paced learning has been applied in a variety of tasks and models. Kumar et al. [44] first demonstrate that a self-paced learning algorithm outperforms the state-of-the-art methods for learning a latent structural SVM. In [45], the self-paced learning paradigm is successfully employed for time series clustering. Jiang et al. [46] propose a self-paced curriculum learning (SPCL) framework, which is capable of jointly considering prior knowledge and the learning progress. To simultaneously enhance the robustness and the effectiveness for supervised learning, Pi et al. [47] first put forward the self-paced boost learning (SPBL) framework, which can reveal and utilize the association of boosting and self-paced learning. Noticing that standard self-paced learning may suffer from the class imbalance issue, Ren et al. [48] carefully design two novel soft-weighting schemes to remedy this issue by assigning weights and selecting instances locally for each class. More recently, SPUDRFs [49] solves the basic issue of ranking and selecting in self-paced learning with respect to fairness and can be handily combined with various deep discriminative models. In SAMVC [50], a soft-weighting form of self-paced learning is introduced into the multiview clustering model to reduce the adverse impacts from outliers and noises, and an auto-weighted strategy is developed to judge the importance of different views. Meng et al. [51] manage to provide some explanations of the self-paced learning paradigm to pursue a theoretical understanding. Overall, these literature publications confirm that self-paced learning is beneficial to avoid entrapment in undesirable local minima and to generally improve the performance of their models.

3 Deep convolutional self-paced clustering

In this section, we first introduce the basic deep clustering model, which takes into account the local structure preservation (LSP). Next, we equip both a convolutional neural network (CNN) and self-paced learning (SPL) into our basic model to learn high-quality cluster-oriented

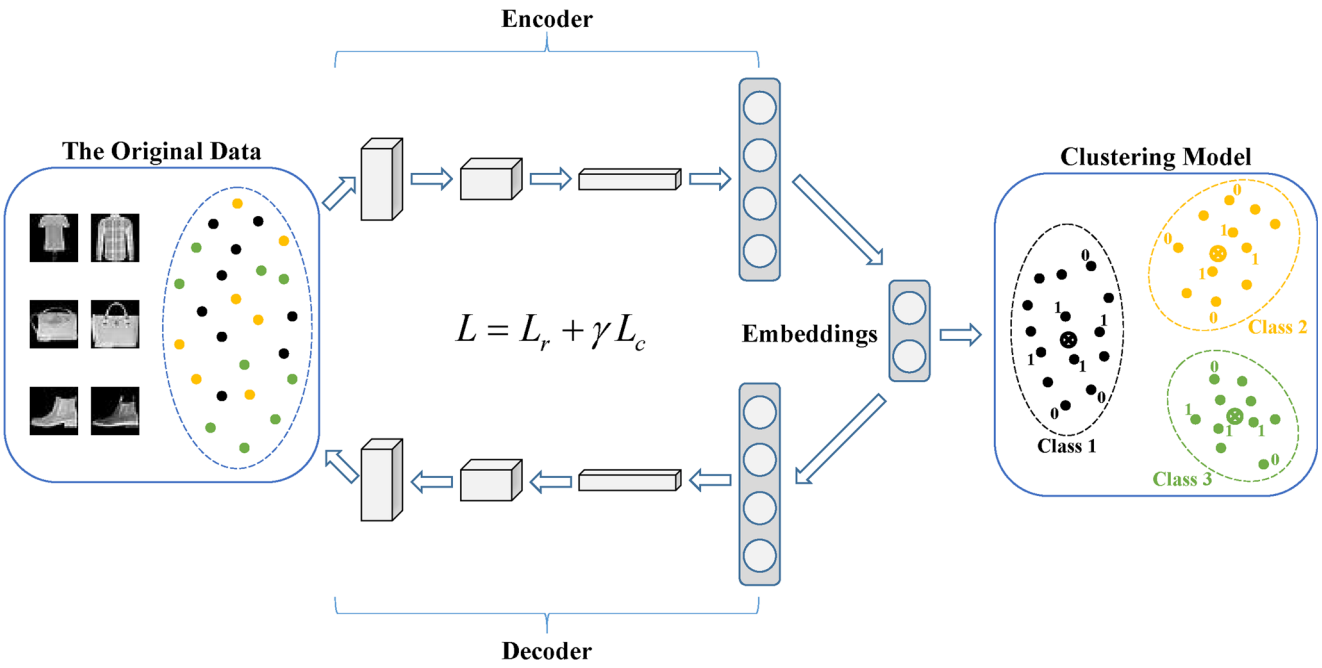


Fig. 1 Framework of the proposed DCSPC algorithm

features containing spatial relationship information and to eliminate the negative effect of marginal samples. In the end, we elaborate the optimization procedure for our intact model. The framework of the proposed DCSPC algorithm is illustrated in Fig. 1.

3.1 Basic deep clustering model

Suppose we have n samples, denoted by $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^D$. In our proposal, we take the SAE as the basic model, in which each sample $\mathbf{x}_i$ is first transformed to a d-dimensional low feature space by the encoder network $f_\theta(\cdot)$,= and then is reconstructed by the decoder network $g_{\boldsymbol{\theta}'}(\cdot)$ using the corresponding d-dimensional feature representation $\mathbf{z}_i$. For simplicity, we presume that the number of clusters K is provided as a priori knowledge. With the center of the jth cluster as $m_j \in \mathbb{R}^d$, the cluster matrix can be expressed as $\mathbf{M} = [m_1, m_2, \ldots, m_K] \in \mathbb{R}^{d\times K}$. The cluster assignment for sample $\mathbf{z}_i$ is denoted by $\mathbf{s}_i \in \{0,1\}^K$, and then, $\mathbf{y}_i = \mathbf{M}\mathbf{s}_i$ represents the cluster centroid to which the sample $\mathbf{z}_i$ belongs. Inspired by DCN [34], we simultaneously consider the reconstruction loss and clustering loss in our basic model. To this end, our objective function can be formulated as:

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

where

$$L_r = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{x}_i - g_{\theta'}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))\|_2^2 \tag{6}$$

and

$$L_c = \frac{1}{n}\sum_{i=1}^{n} \|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i\|_2^2 \tag{7}$$

are corresponding to the reconstruction error and the clustering error, respectively. γ is a parameter that trades off between L_r and L_c to protect feature properties.

However, different from DCN [34], which alternatively updates cluster centers $\mathbf{M}$ and cluster assignments during its finetuning stage, in our work, we merely conduct K-means to initialize $\mathbf{M}$ based on the latent features $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\} = f_{\boldsymbol{\theta}}(\mathbf{X}) \in \mathbb{R}^{d\times n}$ and then fix it to prevent all samples from being assigned into one group in the feature space. In fact, once $\mathbf{M}$ is fixed, decision boundaries are also determined. In such a case, it will be impossible to assign all samples together by minimizing (7) with fixed θ. For simplicity, we refer to the aforementioned basic deep clustering model as BDCM.

In our BDCM, the quality of the target label $\mathbf{y}_i$ is influenced by two factors: the initialization and the finetuning process. That is, to obtain a high-quality target label, we are supposed to carefully focus on the above two factors. Unfortunately, the features learned by utilizing SAE may not be sufficient to support a good initialization. Moreover, as noted earlier, margin samples could seriously mislead the finetuning process. Therefore, there is a strong motivation to find an effective method of feature learning for good initialization and to eliminate the negative effect of marginal samples in the finetuning process. To achieve these two goals, we equip a CNN and SPL in our BDCM.

3.2 Equipping the CNN and SPL

As is known, compared with SAE, CAE is a more powerful network to deal with structured image data as CAE can successfully capture the spatial relationship of data points. Therefore, we can obtain more robust cluster-oriented features with spatial relationship information by replacing SAE with CAE, i.e., equipping a CNN in our BDCM model, which is a great help to creating a better initialization. Moreover, to remove the effect of marginal samples as well as guarantee the stability of the minimization of (5), we equip SPL into our basic model to gradually select the most confident samples for training. Then, the optimization problem of our enhanced model, named Deep Convolutional Self-Paced Clustering (DCSPC), can be formulated as:

$$\begin{aligned}&\min_{\theta,\theta',\mathbf{v},\mathbf{s}} \\ &\frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_i(\|\mathbf{x}_i - g_{\theta'}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))\|_2^2 + \gamma\|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i\|_2^2) - \lambda\mathbf{v}_i, \\ &s.t.\ \mathbf{v}_i \in \{0,1\},\ \mathbf{s}_i \in \{0,1\}^K,\ \mathbf{1}^{\mathrm{T}}\mathbf{s}_i = 1.\end{aligned} \tag{8}$$

However, it is very difficult to control the growth rate of λ for different tasks. Thus, following [37], we adopt a statistic based adaptive method to update it:

$$\lambda = \mu(L^t) + \frac{t}{T}\sigma(L^t), \tag{9}$$

where L^t is a loss vector at the tth training step, and the value of the ith sample is calculated as:

$$L_i^t = \|\mathbf{x}_i - g_{\boldsymbol{\theta}''}(f_{\boldsymbol{\theta}'}(\mathbf{x}_i))\|_2^2 + \gamma\|f_{\boldsymbol{\theta}'}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i^t\|_2^2. \tag{10}$$

$\mu(L^t)$ and $\sigma(L^t)$ denote the average and standard deviation of vector L^t, respectively. T is the number of training iterations. By introducing the weight vector $\mathbf{v}$ and parameter λ, our model can select the most confident samples for training whose loss values are no more than λ. With increasing iterations, increasing numbers of samples will be utilized in training. This process is similar to human cognitive learning, i.e., learning from easy to hard.

3.3 Optimization

3.3.1 Optimization procedure

Similar to the conventional autoencoder, all parameters of the pretraining network can be directly optimized by stochastic gradient descent (SGD) and backpropagation (BP). Therefore, we only focus on the optimization of the finetuning network, where an iterative alternative optimization strategy is considered to optimize problem (8).

1. Update network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$: With the other irrelevant variables fixed, the optimization problem for network parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ is degraded to:

$$\min_{\boldsymbol{\theta},\boldsymbol{\theta}'} \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_i(\|\mathbf{x}_i - g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))\|_2^2 + \gamma\|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i\|_2^2), \tag{11}$$

which can be adaptively optimized by SGD and BP.

2. Update cluster assignments s: With θ, θ' and $\mathbf{v}$ fixed, $\mathbf{s}$ is updated by resolving the following subproblem:

$$\begin{aligned} &\min_{\mathbf{s}} \frac{1}{n}\sum_{i=1}^{n} \|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i\|_2^2, \\ &s.t.\ \mathbf{s}_i \in \{0,1\}^K,\ \mathbf{1}^{\mathrm{T}}\mathbf{s}_i = 1, \end{aligned} \tag{12}$$

whose optimal solution is:

$$\mathbf{s}_{ij} = \begin{cases} 1, & \text{if } j = \arg\min_k \|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \boldsymbol{m}_k\|_2^2, \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Algorithm 1 Deep convolutional self-paced clustering.

Input: Dataset $\mathbf{X}$; Number of clusters K; Balance factor γ; Maximum iterations T; Stopping threshold δ.

Output: Cluster assignments $\mathbf{s}$.

```
// Pretraining
Initialize θ, θ' by minimizing (6);
// Initialization
Initialize M, s by conducting K-means on Z;
// Finetuning
for t = {0, 1, 2, ..., T} do
    Update network parameters θ, θ' by (11);
    Update cluster assignments s by (13);
    Update sample weights v by (16);
    Update learning pace λ by (9);
    if Stopping criterion (17) is met then
        Stop training.
    end
end
```

3. Update sample weights v: Fixing $\boldsymbol{\theta}$, $\boldsymbol{\theta}'$ and $\mathbf{s}$, the optimization problem to $\mathbf{v}$ is degraded as follows:

$$\begin{aligned} &\min_{\mathbf{v}} \\ &\frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_i\left(\|\mathbf{x}_i - g_{\boldsymbol{\theta}'}\left(f_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)\|_2^2 + \gamma\|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i\|_2^2\right) - \lambda\mathbf{v}_i, \\ &s.t.\ \mathbf{v}_i \in \{0,1\}. \end{aligned} \tag{14}$$

Let

$$L_i = \|\mathbf{x}_i - g_{\boldsymbol{\theta}'}(f_{\boldsymbol{\theta}}(\mathbf{x}_i))\|_2^2 + \gamma\|f_{\boldsymbol{\theta}}(\mathbf{x}_i) - \mathbf{M}\mathbf{s}_i\|_2^2, \tag{15}$$

the optimal solution to problem (14) can be expressed as:

$$\mathbf{v}_i = \begin{cases} 1, & \text{if } L_i \le \lambda, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

4. Update learning pace λ: λ is updated by (9).

3.3.2 Stopping criterion

As described in [32], if the variation of predicted labels between two consecutive iterations is less than a threshold δ, the training procedure will be terminated. Formally, the stopping criterion is:

$$1 - \frac{1}{n}\sum_{i,j} \mathbf{s}_{ij}^{t}\mathbf{s}_{ij}^{t-1} < \delta, \tag{17}$$

where $\mathbf{s}_{ij}^{t}$ and $\mathbf{s}_{ij}^{t-1}$ are indicators for whether sample $\mathbf{x}_i$ is assigned to the jth cluster at the $(t-1)$th and tth iteration, respectively. Following the pioneering work [32], we set $\delta = 0.1\%$ in our experiment.

Finally, by alternatively updating the above variables, the proposed algorithm can theoretically converge to the local optimal solution. The optimization procedure of the objective function (8) is summarized in Algorithm 1.

Table 1 Properties of different datasets

	Samples	Classes	Size	Dimensions
MNIST	70000	10	$28 \times 28 \times 1$	784
MNIST-test	10000	10	$28 \times 28 \times 1$	784
USPS	9298	10	$16 \times 16 \times 1$	256
FMNIST	70000	10	$28 \times 28 \times 1$	784
ENGLISHFNT	10160	10	$32 \times 32 \times 1$	1024
COIL20	1440	20	$32 \times 32 \times 1$	1024
COIL100	7200	100	$32 \times 32 \times 1$	1024

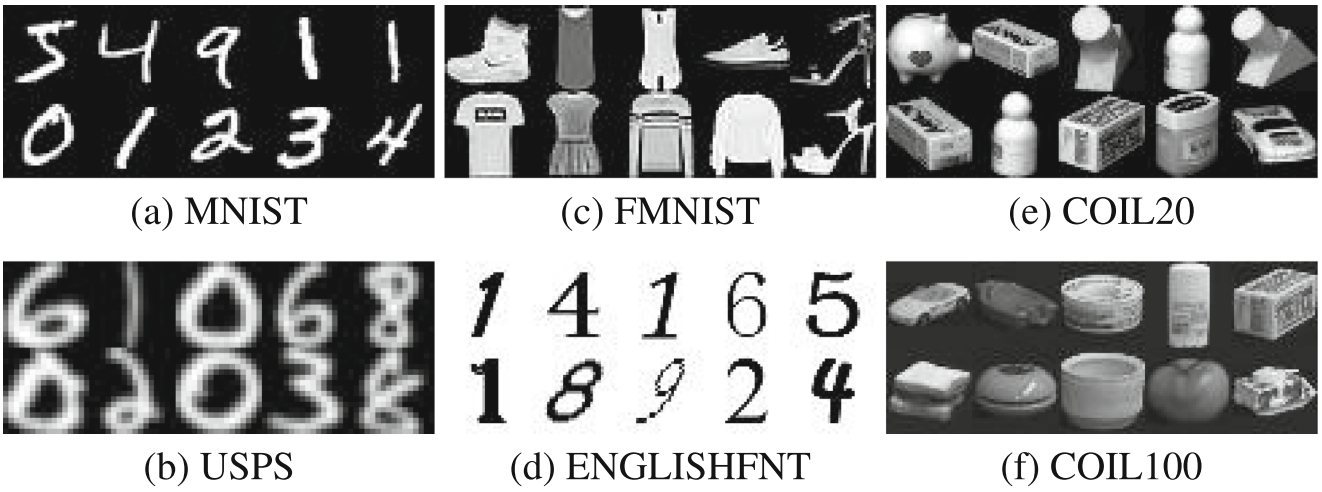


Fig. 2 Examples of different datasets

4 Experiment

In this section, we conduct comprehensive experiments to investigate the performance of our DCSPC. All experiments are implemented on a workstation with an Intel(R) Xeon(R) E5-2640 v4 @ 2.40 GHz CPU, 120 GB RAM, and NVIDIA GeForce GTX 1080 Ti GPU (11 GB caches).

4.1 Datasets

We evaluate our DCSPC on seven popular image datasets, namely, MNIST, MNIST-test, USPS, FMNIST, ENGLISHFNT, COIL20 and COIL100. We will introduce more details about each dataset as follows.

- **MNIST** [52] consists of 70000 grayscale handwritten digit images with a size of 28 × 28 pixels from 10 categories. The training set includes 60000 images, while the test set contains 10000 images.
- **MNIST-test** covers the test set of MNIST, with 10000 samples.
- **USPS** [53] contains 9298 gray handwritten digits with the image size 16 × 16 from 10 categories.
- **FMNIST** [54] is a collection of 70000 fashion product images from 10 classes, with the same image size as MNIST.
- **ENGLISHFNT** [55] consists of more than 60000 128× 128 one-channel character images of English letters from $A(a)$ to $Z(z)$ and Arabic numerals from 0 to 9. We only consider its Arabic numerals subset with 10160 images from 10 categories, where the image size is resized to 32 × 32.
- **COIL20** [56] includes 1440 128 × 128 gray object images from 20 categories and shot from different angles. The resized version of 32 × 32 is adopted in our experiment.
- **COIL100** [57] is similar to COIL20, which incorporates 7200 three-channel images of 100 object

Table 2 Configuration of the proposed DCSPC algorithm

Stage	Parameter	Value
Pretraining	Structure	$\text{conv}^5_{32} - \text{conv}^5_{64} - \text{conv}^3_{128} - \text{fc10} - \text{deconv}^3_{64} - \text{deconv}^5_{32} - \text{deconv}^5_{1}$
	Optimizer	Adam (learning rate = 0.001)
	Epochs	400
	Batch size	256
	Kernel initializer	Glorot uniform
	Activation	ReLU
Finetuning	Structure	$\text{conv}^5_{32} - \text{conv}^5_{64} - \text{conv}^3_{128} - \text{fc10} - \text{deconv}^3_{64} - \text{deconv}^5_{32} - \text{deconv}^5_{1}$
	Optimizer	Adam (learning rate = 0.0001)
	Iterations T	100
	Batch size	256
	Balance factor γ	0.5
	Threshold δ	0.1%

categories. We grayscale it and map the image size to 32×32.

The properties of the above datasets are summarized in Table 1. Examples of some datasets are shown in Fig. 2. In our experiments, all datasets are rescaled to $[-1, 1]$ for each element before being fed to clustering algorithms. Note that for some datasets that have been separated into training set and testing set, both parts are included for clustering.

4.2 Evaluation metrics

To measure the clustering performance, we adopt three popular standard metrics including accuracy (ACC) [59], normalized mutual information (NMI) [60] and adjusted Rand index (ARI) [61]. ACC is defined as follows:

$$\mathrm{ACC} = \max_{m} \frac{\sum_{i=1}^{n} \mathbf{1}\{y_i = map(s_i)\}}{n}, \tag{18}$$

where y_i and s_i denote the ground-truth label and the cluster assignment produced by the model, respectively. $map(s_i)$ is the permutation map function, which embodies all possible one-to-one mappings from clusters to labels. The best mapping can be efficiently computed by the Hungarian algorithm [58]. NMI calculates the normalized measure of similarity between two labels of the same data, which can be formulated as:

$$\mathrm{NMI} = \frac{\mathbf{I}(y; s)}{\max\{H(y), H(s)\}}, \tag{19}$$

where $\mathbf{I}(y; s)$ and H represent the mutual information between y and s and the entropy value, respectively. ARI is the corrected-for-chance version of the Rand index (RI) [62], which is computed as follows:

$$\mathrm{ARI} = \frac{\mathrm{RI} - E(\mathrm{RI})}{\max(\mathrm{RI}) - E(\mathrm{RI})}, \tag{20}$$

where $E(\mathrm{RI})$ is the expectation of RI.

Generally, the above metrics are commonly adopted in a variety of clustering literature [32–35, 37–42]. Each one offers pros and cons, but using them together is sufficient to demonstrate the effectiveness of the clustering algorithms. Note that ACC and NMI range within [0, 1], while the range of ARI is $[-1, 1]$, and a higher value signifies a better clustering performance.

4.3 Compared methods

Several clustering methods are employed for comparison with our DCSPC, which can be roughly grouped as three categories: 1) *Traditional methods*, containing K-means (KM) [5], Gaussian mixture models (GMM) [6], and spectral clustering (SC) [7]; 2) *Representation-based methods*, including SAE [25] and CAE [26]; 3) *Deep clustering methods*, consisting of deep embedded clustering

Table 3 Clustering performance of different algorithms on seven datasets in terms of ACC and NMI

	MNIST		MNIST-test		USPS		FMNIST		ENGLISHFNT		COIL20		COIL100	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
KM [5]	0.532	0.500	0.546	0.501	0.668	0.627	0.474	0.512	0.541	0.500	0.607	0.487	0.495	0.769
GMM [6]	0.433	0.366	0.540	0.493	0.551	0.530	0.556	0.557	0.520	0.503	0.655	0.775	0.489	0.764
SC [7]	0.656	0.731	0.660	0.704	0.649	0.794	0.508	0.575	–	–	0.628	0.523	–	–
SAE [25]	0.782	0.715	0.668	0.596	0.617	0.573	0.508	0.539	0.577	0.539	0.558	0.715	0.504	0.767
CAE [26]	0.849	0.793	0.790	0.726	0.742	0.733	0.583	0.621	0.717	0.697	0.668	0.782	0.516	0.785
DEC [32]	0.841	0.813	0.699	0.677	0.693	0.702	0.518	0.546	0.595	0.573	0.573	0.759	0.500	0.765
IDEC [33]	0.842	0.838	0.715	0.694	0.721	0.732	0.529	0.557	0.665	0.667	0.586	0.761	0.519	0.792
DCN [34]	0.811	0.757	0.802	0.786	0.730	0.719	0.501	0.558	–	–	–	–	–	–
DKM [35]	0.840	0.796	–	–	0.757	0.776	0.539	0.563	–	–	–	–	–	–
ConvDEC [36]	0.886	0.876	0.848	0.826	0.779	0.811	0.566	0.614	0.681	0.695	0.687	0.798	0.511	0.775
ASPC [37]	0.859	0.842	0.782	0.734	0.753	0.766	0.600	0.633	0.649	0.638	0.569	0.743	0.509	0.782
ASPC w/o SPL	0.844	0.818	0.747	0.684	0.725	0.730	0.597	0.625	0.624	0.598	0.551	0.723	0.500	0.771
SDCN [38]	–	–	–	–	0.781	0.795	–	–	–	–	–	–	–	–
SDEC [39]	0.861	0.829	–	–	0.764	0.777	–	–	–	–	–	–	–	–
DCSPC (ours)	**0.914**	**0.882**	**0.853**	**0.830**	**0.791**	**0.826**	**0.629**	**0.658**	**0.729**	**0.755**	**0.694**	**0.804**	**0.540**	**0.800**

Table 4 Clustering performance of different algorithms with DA tricks in terms of ACC and NMI on the MNIST, USPS, FMNIST, and ENGLISHFNT datasets

	MNIST		USPS		FMNIST		ENGLISHFNT	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
ConvDEC+DA [36]	0.952	**0.945**	0.941	**0.942**	0.586	0.635	0.890	0.935
ASPC+DA [37]	0.913	0.895	0.842	0.879	0.549	0.618	0.799	0.795
DDC+DA [40]	0.932	0.927	0.975	0.935	0.594	0.661	0.686	0.814
DCSPC+DA (ours)	**0.967**	0.923	**0.976**	0.938	**0.623**	**0.662**	**0.980**	**0.949**

(DEC) [32], improved deep embedded clustering (IDEC) [33], deep clustering network (DCN) [34], deep K-means (DKM) [35], convolutional deep embedded clustering (ConvDEC) [36], adaptive self-paced clustering (ASPC) [37], structural deep clustering network (SDCN) [38], semi-supervised deep embedded clustering (SDEC) [39], and deep density-based clustering (DDC) [40].

4.4 Experimental settings

In the pretraining stage, the encoder network structure is $\text{conv}^5_{32} \rightarrow \text{conv}^5_{64} \rightarrow \text{conv}^3_{128} \rightarrow \text{fc10}$, where conv^k_n denotes a convolutional layer with n filters, $k \times k$ kernel size and 2 stride length, whose mirrored version is regarded as the decoder network. The Glorot uniform [63] is employed as the layer kernel initializer. All internal layers (except for the input, output, and embedding layers) are activated by ReLU [64]. We place the decoder on the top of the encoder to construct the CAE, which is trained in an end-to-end manner for 400 epochs using the Adam [65] optimizer with an initial learning rate of 0.001. In the finetuning stage, different from the previous works [31, 32, 37], the decoder is also considered to protect data properties. The number of clusters K is set according to the ground truth labels, i.e., to let K be equal to the ground truth cluster number, and nonparametric Bayesian methods [66] can be adopted to estimate unknown K. The maximum number of iterations is set to $T = 100$. The Adam [65] optimizer with initial learning rate 0.0001 is used. The batch size is fixed to 256. The balance factor is fixed to $\gamma = 0.5$. The threshold of the stopping criterion is set to $\delta = 0.1\%$. A summary of these setups is listed in Table 2. Note that for reasonable evaluation, we perform 5 random restarts for all experiments and report the average results to compare with the others based on Python *3.7* and TensorFlow *2.0.0-beta0*.

4.5 Performance comparison

Table 3 reports the clustering results of all compared algorithms, where the mark "−" denotes that the experimental results or code are inaccessible from the corresponding paper, and the bold number refers to the best clustering result. As is shown, in most cases, deep clustering algorithms perform better than traditional algorithms by a large margin, reflecting the effectiveness of deep neural networks in nonlinear feature learning. In fact, deep clustering algorithms also achieve better performance than the corresponding representation-based approaches, i.e., SAE [25] and CAE [26], in almost all cases for all metrics, which clearly demonstrates that combining feature learning and clustering can bring about more suitable features for clustering, indicating the advantage of using the joint optimization criterion.

Moreover, comparison between SAE and CAE validates the superiority of CAE. This is because CAE is equipped with the convolutional neural network (CNN) instead of the fully connected network adopted in SAE, such that the information of spatial correlation lying in raw data can be effectively captured and the quality of feature embedding is enhanced. In addition, the performance of IDEC [33] over DEC [32] shows that the LSP mechanism can help protect feature properties to improve clustering performance. Additionally, the superior result of ASPC [37] over ASPC w/o SPL (i.e., ASPC without SPL) indicates that the SPL paradigm is provided with the advantage of eliminating the negative effects from marginal samples by gradually selecting increasing numbers of high-confidence samples into training. In fact, the proposed DCSPC succeeds in achieving the best performance in terms of all metrics on all datasets; in particular, the state-of-the-art performance (achieved by ASPC) on FMNIST is improved from 60.0% (63.3%) to 62.9% (65.8%) by our method with respect to ACC (NMI).

We also find that a novel data augmentation (DA) trick is employed by [36, 37, 40] and improved clustering performance is acquired. In this study, we incorporate such DA strategy with our DCSPC algorithm. On the one hand, we can evaluate whether the proposed DCSPC could also benefit from DA. On the other hand, we can arrive at more fair comparison with the rivals [36, 37, 40] that use the DA trick. It is observed from Tables 3 and 4 that the DA strategy can obviously promote the clustering performance of our DCSPC on the MNIST, USPS, and ENGLISHFNT datasets. Compared with other competitors adopting a DA strategy,

Table 5 Performance of DCSPC with different configurations on MNIST, MNIST-test, USPS, and FMNIST

	CNN	LSP	SPL	MNIST			MNIST-test			USPS			FMNIST		
				ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
BasicDC	×	×	×	0.8442	0.8175	0.7748	0.7469	0.6838	0.6054	0.7246	0.7301	0.6166	0.5973	0.6254	0.4771
ASPC	×	×	✓	0.8593	0.8423	0.8046	0.7820	0.7337	0.6694	0.7533	0.7661	0.6679	0.5996	0.6332	0.4804
BDCM	×	✓	×	0.8523	0.8338	0.7918	0.7491	0.6855	0.6114	0.7304	0.7343	0.6235	0.5996	0.6386	0.4829
–	×	✓	✓	0.8682	0.8598	0.8214	0.7910	0.7439	0.6871	0.7594	0.7725	0.6664	0.6001	0.6372	0.4807
–	✓	×	×	0.8916	0.8740	0.8353	0.8308	0.8110	0.7497	0.7783	0.8058	0.7157	0.6229	0.6489	0.4932
–	✓	×	✓	0.9028	0.8795	0.8490	0.8358	0.8292	0.7651	0.7861	0.8203	0.7288	0.6276	0.6531	0.4978
–	✓	✓	×	0.8977	0.8771	0.8421	0.8345	0.8131	0.7551	0.7801	0.8049	0.7164	0.6276	0.6551	0.4982
SPL w/o LSP	✓	✓	✓	0.9087	0.8801	0.8519	0.8429	0.8294	0.7668	0.7893	0.8220	0.7343	0.6288	0.6564	0.4991
DCSPC (ours)	✓	✓	✓	**0.9142**	**0.8820**	**0.8546**	**0.8461**	**0.8297**	**0.7745**	**0.7914**	**0.8258**	**0.7372**	**0.6294**	**0.6582**	**0.5005**

DCSPC+DA achieves the highest ACC score on all four datasets and works the best on two datasets with respect to NMI. In particular, our DCSPC+DA obtains a great advantage over other methods on the ENGLISHFNT dataset and surpasses the second-best method ConvDEC+DA [36] by 9.0% and 1.4% in terms of ACC and NMI, respectively.

Overall, these superior results are quite in line with our conjecture, owing to the fact that CNN plays a key role in feature extraction during pretraining and that both LSP and SPL are conducive to promoting the model training process, leading to a satisfactory result.

4.6 Ablation study

This subsection describes an ablation study to analyze the contributions of three parts of our DCSPC, i.e., convolutional neural network (CNN), local structure preservation (LSP), and self-paced learning (SPL). Removing the CNN from DCSPC means that the stacked autoencoder (SAE) is considered instead of convolutional autoencoder (CAE) as our basic model. Freezing LSP in DCSPC is equivalent to merely keeping the encoder unchanged during finetuning. Disabling SPL in DCSPC equates to fixing $\mathbf{v} = [1, 1, \ldots, 1]^{\mathrm{T}} = \mathbf{1}$ in (8). We regard the configuration with all three parts removed as BasicDC, as named in [37].

The results of DCSPC with different configurations are shown in Tables 5 and 6, where whether to use a specific part in DCSPC is marked by "✓" or "×". When we individually add one of three parts to the BasicDC, improved clustering performance can be observed in most cases. In fact, the most considerable improvement is observed by adding the CNN. As we can see, the results of the last 5 rows are consistently much better than those of the first 4 rows, which reveals that the CNN is the most important part of our DCSPC algorithm for learning robust features and creating the right initialization atmosphere for the following clustering task. When all three parts are simultaneously considered, our DCSPC algorithm achieves the best performance in terms of all metrics on all datasets. Moreover, when introducing a self-paced learning mechanism in our DCSPC, we simultaneously include the reconstruction and clustering losses instead of merely considering the clustering loss. The superior performance of DCSPC as shown in the second row from the bottom confirms that our DCSPC can describe the confidence of the samples more comprehensively.

4.7 Parameter analysis

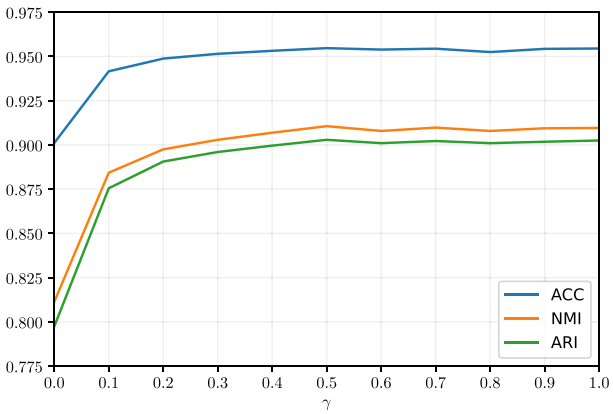
Since our model is based on a DNN, which inevitably involves the necessity of tuning some hyperparameters, and because it is impractical to search for an optimal value in the whole parameter space, we determine most of the parameters by following the previous work [32]. Here,

Table 6 Performance of DCSPC with different configurations on ENGLISHFNT, COIL20, and COIL100

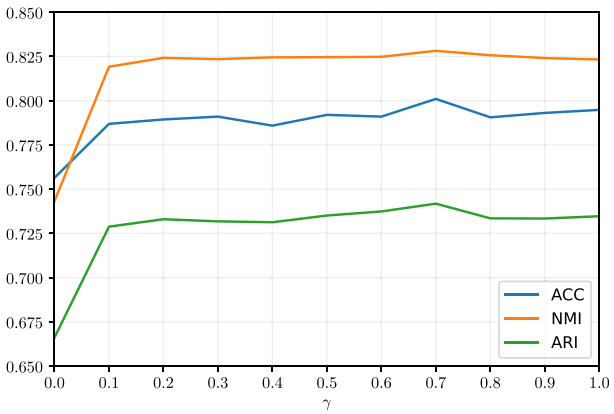
	CNN	LSP	SPL	ENGLISHFNT			COIL20			COIL100		
				ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
BasicDC	×	×	×	0.6243	0.5984	0.4667	0.5507	0.7234	0.5030	0.5004	0.7707	0.4457
ASPC	×	×	✓	0.6491	0.6379	0.4961	0.5693	0.7431	0.5184	0.5090	0.7824	0.4519
BDCM	×	✓	×	0.6388	0.6049	0.4764	0.5703	0.7327	0.5178	0.5099	0.7748	0.4555
–	×	✓	✓	0.6438	0.6338	0.4926	0.5878	0.7592	0.5421	0.5240	0.7860	0.4565
–	✓	×	×	0.7140	0.7260	0.6162	0.6671	0.7821	0.6050	0.5144	0.7873	0.4645
–	✓	×	✓	0.7276	0.7497	0.6319	0.6674	0.8035	0.6072	0.5170	0.7936	0.4580
–	✓	✓	×	0.7273	0.7337	0.6262	0.6850	0.7842	0.6134	0.5156	0.7872	0.4714
SPL w/o LSP	✓	✓	✓	0.7282	0.7547	0.6347	0.6889	0.8038	0.6227	0.5350	0.7962	0.4766
DCSPC (ours)	✓	✓	✓	**0.7291**	**0.7554**	**0.6483**	**0.6944**	**0.8043**	**0.6240**	**0.5404**	**0.7997**	**0.4789**

due to limited space, we focus only on the importance coefficient γ and the cluster number K.

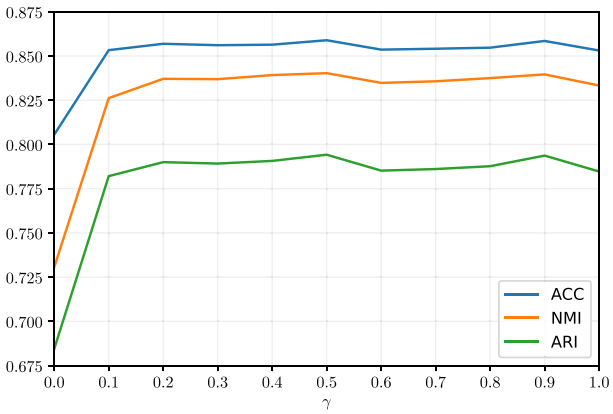
We first study the sensitivity of the importance coefficient γ, which trades off between the reconstruction loss and the clustering loss. As seen from the comprehensive experiments in Section 4.5, the proposed algorithm works well with fixed $\gamma = 0.5$. Figure 3 shows how our model performs with different γ values on the MNIST, MNIST-test, USPS, and ENGLISHFNT datasets. When $\gamma = 0$, the objective only contains the reconstruction part and the clustering constraint loses efficacy, resulting in poor performance. When γ increases gradually, the clustering



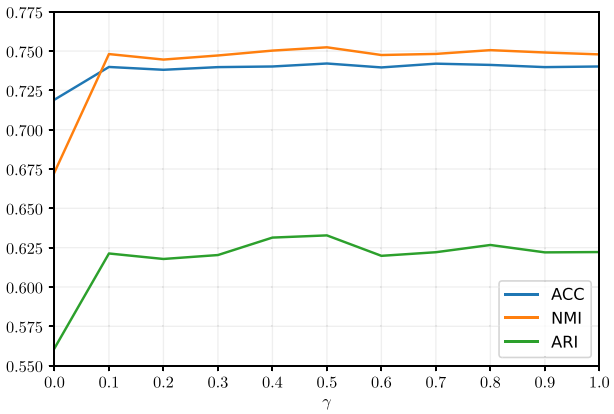
(a) MNIST



(c) USPS



(b) MNIST-test



(d) ENGLISHFNT

Fig. 3 Clustering performance on four datasets with different γ

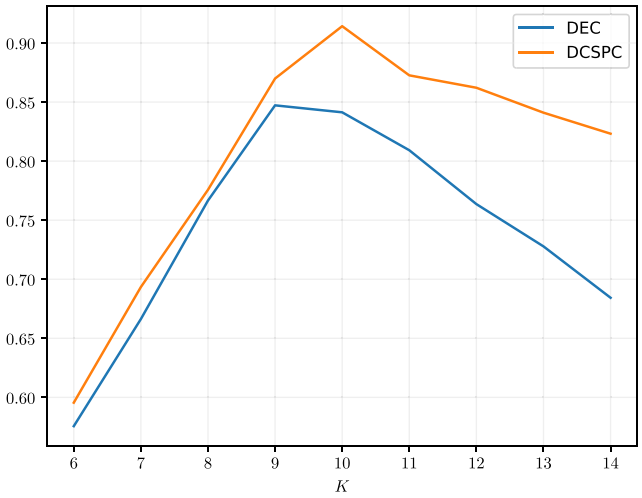


Fig. 4 Clustering performance with different K

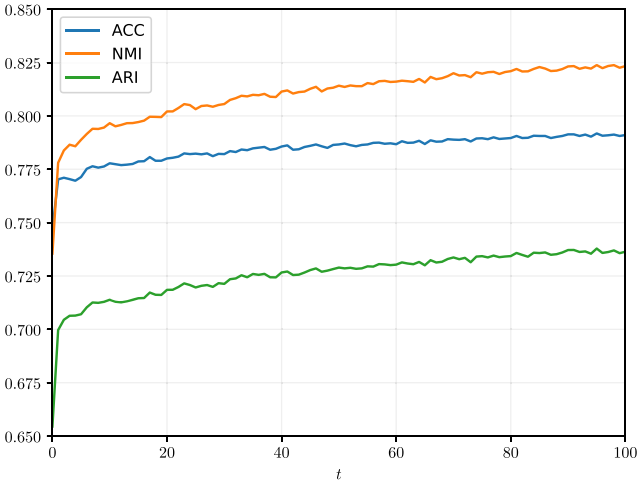


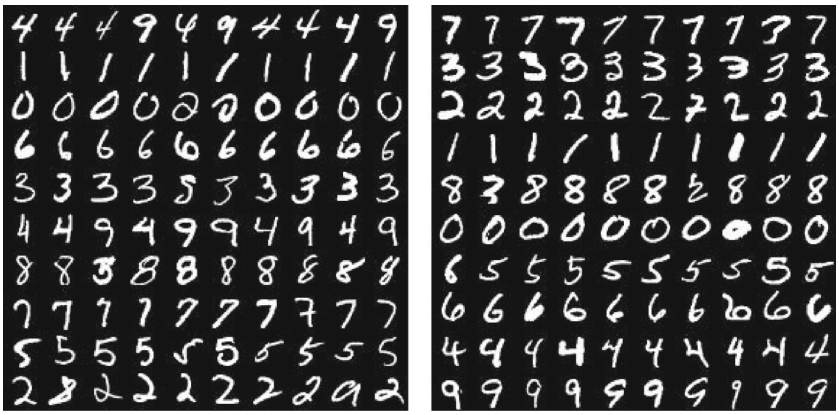
Fig. 6 Clustering performance vs. iterations

constraint efficacy is rejuvenated and improved clustering performance is obtained. Moreover, as γ increases, the fluctuation of metrics is extremely mild, which signifies that the proposed model yields satisfactory performance for a suitable range of γ and suggests that our approach is desensitized to the specific value of γ.

We next discuss the cluster number K. In the previous experiments in Sections 4.5 and 4.6, we have assumed that the cluster number K on each dataset is predefined based on the ground truth labels. However, in many real-world applications, K is usually unknown in unsupervised settings. Therefore, here we run our model on the MNIST dataset with different K to search for the optimal value, whose quality is measured by ACC. As shown in Fig. 4, we can see that our DCSPC achieves the highest ACC when $K = 10$, according with the ground truth labels. For a visual understanding, we also show the clustering results of DEC [32] and our method in Fig. 5. We can observe that our DCSPC tends to partition all handwritten digit images into 10 clusters, which is consistent with the ground truth labels. However, DEC confuses the numbers 4 and 9, i.e., grouping all images into 9 clusters, which conflicts with the ground truth labels.

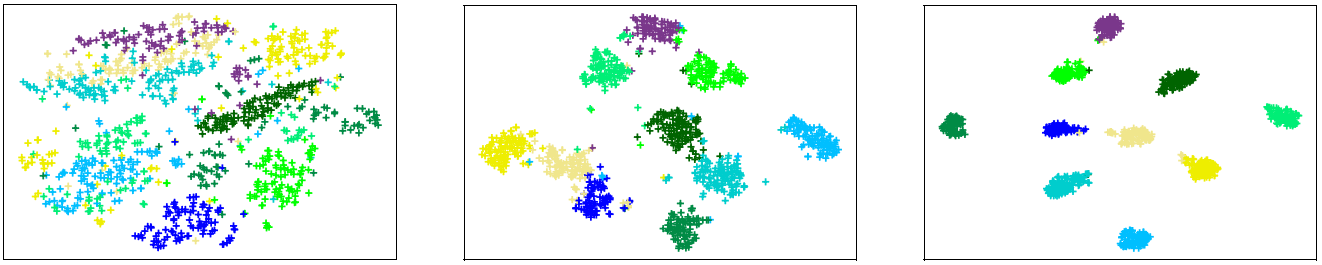
4.8 Convergence analysis

In this section, we record the evolution of three metrics over iterations on the USPS dataset to study the convergence of our DCSPC. The results are displayed in Fig. 6. We can observe a clear ascending trend of each metric in the first few iterations, and all metrics ultimately achieve stability. Furthermore, we visualize the 10-dimensional learned feature embeddings in different periods of the optimization on a subset of the MNIST dataset with 1000 examples, using tSNE [17], as described in Fig. 7. We can observe that data points projected from raw pixel space are highly overlapped, indicating the difficulty of the clustering task. After initialization, feature points extracted from the CAE are more discrete than raw samples, and although most points in the same cluster have crowded together,



(a) DEC (b) DCSPC

Fig. 5 Visualization of the clustering results



(a) Original (b) Initial (c) Final

Fig. 7 Visualization of the clustering process using tSNE

there are still many inseparable points near cluster borders (i.e., marginal samples). As the finetuning process proceeds until the model reaches convergence, feature points reach stability and are well separated. Figures 6 and 7 demonstrate that the proposed algorithm usually converges in practice.

5 Conclusion

In this paper, we present a new approach for jointly learning representations and clustering by incorporating a convolutional network, local structure preservation and a self-paced learning mechanism. Using a convolutional network in the training process is beneficial for capturing high-quality features with spatial relationship information for subsequent clustering tasks. Local structure preservation effectively prevents feature space from being distorted by the clustering loss. Then, we select samples from easy to hard by integrating the self-paced learning mechanism to make the training process more stable. In comparison with existing approaches, our approach has achieved superior performance on different datasets concurrently. Future work may include conducting more experiments on three-channel image, text, and audio datasets and exploring more advanced convolutional networks for feature learning to improve clustering performance.

Acknowledgements The authors are thankful for the financial support in part by the Key-Area Research and Development Program of Guangdong Province (2019B010153002), by the National Natural Science Foundation of China (U1936206, 61806202, 61803087, 61803086), by the Feature Innovation Project of Guangdong Province Department of Education (2019KTSCX192), by the Guangdong Basic and Applied Basic Research Fund (2020B1515310003), and by the Foshan Core Technology Research Project (1920001001367). Rui Chen and Yongqiang Tang contribute equally to this article.

References

1. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556
2. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions, in IEEE Conference on Computer Vision and Pattern Recognition, pp 1–9
3. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition, in IEEE Conference on Computer Vision and Pattern Recognition, pp 770–778
4. Hayashi T, Fujita H, Hernandez-Matamoros A (2021) Less complexity one-class classification approach using construction error of convolutional image transformation network. Inf Sci 560:217–234
5. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Berkeley Symposium on Mathematical Statistics and Probability, vol 1(14):281–297. Oakland
6. Bishop CM (2006) Pattern recognition and machine learning. Springer
7. Shi J, Malik J (2000) Normalized cuts and image segmentation. IEEE Trans Pattern Anal Mach Intell 22(8):888–905
8. Wold S, Esbensen K, Geladi P (1987) Principal component analysis. Chemometrics Intell Labor Syst 2(1–3):37–52
9. Cox TF, Cox MAA (2001) Multidimensional scaling. J R Stat Soc 46(2):1050–1057
10. Xu W, Liu X, Gong Y (2003) Document clustering based on non-negative matrix factorization. In: Annual Conference on Research and Development in Informaion Retrieval. ACM, pp 267–273
11. Peng H, Hu Y, Chen J, Wang H, Li Y, Cai H (2020) Integrating Tensor Similarity to Enhance Clustering Performance. IEEE Transactions on Pattern Analysis and Machine Intelligence
12. Tang Y, Xie Y, Zhang C, Zhang Z, Zhang W (2021) One-step multi-view subspace segmentation via joint skinny tensor learning and latent clustering. IEEE Transactions on Cybernetics. https://doi.org/10.1109/TCYB.2021.3053057
13. Zhang Y, Yang Y, Li T, Fujita H (2019) A multitask multiview clustering algorithm in heterogeneous situations based on LLE and LE. Knowl-Based Syst 163:776–786
14. Wang H, Yang Y, Liu B, Fujita H (2019) A study of graph-based system for multi-view clustering. Knowl-Based Syst 16:1009–1019
15. Deng T, Ye D, Ma R, Fujita H, Xiong L (2020) Low-rank local tangent space embedding for subspace clustering. Inf Sci 508:1–21
16. Hayashi T, Fujita H (2021) Cluster-based zero-shot learning for multivariate data. J Ambient Intell Human Comput 12:1897–1911
17. Maaten L. v. d., Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9:2579–2605
18. Roweis S, Saul L (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–6
19. Hinton G, Salakhutdinov R (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507

20. Schroff F, Kalenichenko D, Philbin J (2015) A unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 815–823
21. Hershey J, Chen Z, Leroux J, Watanabe S (2016) Deep clustering: Discriminative embeddings for segmentation and separation. In: IEEE International Conference on Acoustics, Speech and Signal Processing, pp 31–35
22. Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2(5):359–366
23. Bengio Y, Courville A, Vincent P (2013) Representation learning: A review and new perspectives
24. Bruna J, Mallat S (2013) Invariant scattering convolution networks. IEEE Trans Pattern Anal Mach Intell 35(8):1872–1886
25. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P.-A. (2010) Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res 11:3371–3408
26. Guo X, Liu X, Zhu E, Yin J (2017) Deep clustering with convolutional autoencoders. In: International Conference on Neural Information Processing, pp 373–382
27. Peng X, Xiao S, Feng J, Yau W, Yi Z (2016) Deep subspace clustering with sparsity prior. In: International Joint Conference on Artificial Intelligence
28. Ji P, Zhang T, Li H, Salzmann M, Reid ID (2017) Deep subspace clustering networks. In: Annual Conference on Neural Information Processing Systems, pp 23–32
29. Yang J, Parikh D, Batra D (2016) Joint unsupervised learning of deep representations and image clusters. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 5147–5156
30. Chang J, Wang L, Meng G, Xiang S, Pan C (2017) Deep adaptive image clustering. In: International Conference on Computer Vision, pp 5880–5888
31. Li F, Qiao H, Zhang B (2017) Discriminatively boosted image clustering with fully convolutional auto-encoders. Pattern Recogn 83:161–173
32. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: International Conference on Machine Learning, pp 478–487
33. Guo X, Gao L, Liu X, Yin J (2017) Improved deep embedded clustering with local structure preservation. In: International Joint Conference on Artificial Intelligence, pp 1753–1759
34. Yang B, Fu X, Sidiropoulos ND, Hong M (2017) Towards kmeans-friendly spaces: Simultaneous deep learning and clustering. Int Conf Mach Learn 70:3861–3870
35. Fard MM, Thonet T, Gaussier E (2020) Deep k-means: Jointly clustering with k-means and learning representations. Pattern Recognition Letters
36. Guo X, Zhu E, Liu X, Yin J (2018) Deep embedded clustering with data augmentation. In: Asian Conference on Machine Learning, pp 550–565
37. Guo X, Liu X, Zhu E, Zhu X, Li M, Xu X, Yin J (2020) Adaptive self-paced deep clustering with data augmentation. IEEE Trans Knowl Data Eng 32(9):1680–1693
38. Bo D, Wang X, Shi C, Zhu M, Lu E, Cui P (2020) Structural deep clustering network in international world wide web conferences
39. Ren Y, Hu K, Dai X, Pan L, Hoi SCH, Xu Z (2019) Semi-supervised deep embedded clustering. Neurocomputing 325:121–130
40. Ren Y, Wang N, Li M, Xu Z (2020) Deep density-based image clustering. Knowledge-Based Systems
41. Huang Q, Zhang Y, Peng H, Dan T, Weng W, Cai H (2020) Deep subspace clustering to achieve jointly latent feature extraction and discriminative learning. Neurocomputing 404:340–350
42. Chen R, Tang Y, Zhang C, Zhang W, Hao Z (2021) Deep multi-network embedded clustering. Pattern Recogn Artif Intell 34(1):14–24
43. Khan F, Mutlu B, Zhu X (2011) How do humans teach: On curriculum learning and teaching dimension. In: Annual Conference on Neural Information Processing Systems, pp 1449–1457
44. Kumar MP, Packer B, Koller D (2010) Self-paced learning for latent variable models. In: Annual Conference on Neural Information Processing Systems, pp 1189–1197
45. Tang Y, Xie Y, Yang X, Niu J, Zhang W (2021) Tensor multi-elastic kernel self-paced learning for time series clustering. IEEE Trans Knowl Data Eng 33(3):1223–1237
46. Jiang L, Meng D, Zhao Q, Shan S, Hauptmann AG (2015) Self-paced curriculum learning. In: AAAI Conference on Artificial Intelligence
47. Pi T, Li X, Zhang Z, Meng D, Wu F, Xiao J, Zhuang Y (2016) Self-paced boost learning for classification. In: International Joint Conference on Artificial Intelligence
48. Ren Y, Zhao P, Sheng Y, Yao D, Xu Z (2017) Robust softmax regression for multi-class classification with self-paced learning. In: International Joint Conference on Artificial Intelligence
49. Pan L, Ai S, Ren Y, Xu Z (2020) Self-paced deep regression forests with consideration on underrepresented examples. In: European Conference on Computer Vision
50. Ren Y, Huang S, Zhao P, Han M, Xu Z (2020) Self-paced and auto-weighted multi-view clustering. Neurocomputing 383:248–256
51. Meng D, Zhao Q, Jiang L (2017) A theoretical understanding of self-paced learning. Inf Sci 414:319–328
52. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
53. Hull JJ (1994) A database for handwritten text recognition research. IEEE Trans Pattern Anal Mach Intell 16(5):550–554
54. Han X, Rasul K, Vollgraf R (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv:1708.07747v2
55. de Campos TE, Babu BR, Varma M (2009) Character recognition in natural images. In: International Conference on Computer Vision Theory and Applications, Lisbon
56. Nene SA, Nayar SK, Murase H (1996) Columbia object image library (COIL-20). Technical report CUCS-006-96
57. Nene SA, Nayar SK, Murase H (February 1996) Columbia object image library (COIL-100), Technical report CUCS-006-96
58. Kuhn HW (1955) The hungarian method for the assignment problem. Naval Res Logist Quart 2(1):83–97
59. Li T, Ding C (2006) The relationships among various nonnegative matrix factorization methods for clustering. In: International Conference on Data Mining, pp 362–371
60. Strehl A, Ghosh J (2002) Cluster ensembles — a knowledge reuse framework for combining multiple partitions. J Mach Learn Res 3:583–617
61. Hubert L, Arabie P (1985) Comparing partitions. J Classif 2(1):193–218
62. Rand WM (1971) Objective criteria for the evaluation of clustering methods. J Am Stat Assoc 66(336):846–850
63. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. J Mach Learn Res 9:249–256
64. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. J Mach Learn Res 15:315–323
65. Kingma D, Ba J (2014) Adam: A method for stochastic optimization, arXiv:1412.6980

66. Ma Z, Lai Y, Kleijn W, Song Y, Wang L, Guo J (2018) Variational bayesian learning for dirichlet process mixture of inverted dirichlet distributions in non-gaussian image feature modeling. IEEE Trans Neural Netw Learn Syst 30:449–463

Rui Chen received the M.E. degree from the Department of Automation, Foshan University, Foshan, China, in 2021. He is currently pursuing the Ph.D. degree at Hainan University, Haikou, China. His research interests include machine learning, data mining, and computer vision.

Yongqiang Tang received the B.S. degree from the Department of Automation, Central South University, Changsha, China, in 2014, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, in 2019. He is currently an Assistant Professor with the Research Center of Precision Sensing and Control, Institute of Automation, CAS. His research interests include machine learning, computer vision, and data mining.

Lei Tian is currently a Ph.D. candidate in the Institute of Automation, Chinese Academy of Sciences (CAS). He received the B.S. degree from the Department of Precision Instrument, Tsinghua University, Beijing, China, in 2017. His research interests include machine learning, pattern recognition, computer vision, transfer learning, and domain adaptation.

Caixia Zhang received the Ph.D. degree in control theory and control engineering from Guangdong University of Technology, Guangzhou, China, in 2012. She is currently a Full Professor with the Department of Automation, Foshan University, Foshan, China. Her research interests include intelligent computing, intelligent control system, and multi-source information fusion.

Wensheng Zhang received the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, in 2000. He joined the Institute of Software, CAS, in 2001, where he is a Professor of machine learning and data mining and the Director of the Research and Development Department, Institute of Automation. His research interests include computer vision, pattern recognition, and artificial intelligence.