Information Sciences 609 (2022) 876–896

Contents lists available at ScienceDirect

Information Sciences

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

Centroids-guided deep multi-view *K*-means clustering

Jing Liu, Fuyuan Cao [*], Jiye Liang

[a] *Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan 030006, China*

A R T I C L E I N F O

Article history:
Received 8 May 2021
Received in revised form 31 May 2022
Accepted 17 July 2022
Available online 20 July 2022

Keywords:
Multi-view clustering
Deep clustering
k-means
Centroids-guided

A B S T R A C T

With the progress of deep learning used in unsupervised learning, deep approach based multi-view clustering methods have been increasingly proposed in recent years. However, in most of these methods, deep representation learning is not organically integrated into the multi-view clustering process. They either conduct deep representation learning and clustering in a separate manner, or use the pseudo cluster labels to supervise deep representation learning. In this paper, we propose a centroids-guided deep multi-view *k*-means clustering method, which organically incorporates deep representation learning into the multi-view *k*-means objective by using the cluster centroids in multi-view *k*-means to guide the deep learning of each view. In turn, more *k*-means-friendly representations are produced to further optimize the multi-view *k*-means objective. The cluster centroids of each view obtained under a common clustering partition not only represent the semantic information of the clusters but also imply consistency among different views. By reducing the loss between each representation and its assigned cluster centroid with respect to the network parameters of each view, the representations of different views will be more *k*-means-friendly toward a common partition. Experiments on several datasets demonstrate the effectiveness of our method.

© 2022 Elsevier Inc. All rights reserved.

1. Introduction

In many real-world applications, the instances can be represented by heterogeneous features from multiple views. For example, images can be represented by different visual descriptors, documents may be translated into multiple languages, and web pages consist of both text and hyperlinks. The features of different views are both compatible and complementary to each other. Multi-view clustering (MVC) utilizes information from multiple views to learn a common clustering partition. It has been extensively studied in recent years [1] and demonstrates better performance than clustering on a single view.

Traditional MVC works mainly focus on shallow methods, in which multi-view representation learning is based on shallow techniques, such as nonnegative matrix factorization (NMF) used in [2–5], subspace representation used in [6,7], and canonical correlation analysis (CCA) used in [8,9]. However, shallow MVC methods may not work well for some complex nonlinear data. In recent years, deep learning has been applied in MVC due to its powerful nonlinear representation capability. Some works conduct deep multi-view representation learning and traditional clustering in a separate manner. For example, the works [10,11] apply deep CCA to learn the deep nonlinear representations between views by using deep neural networks as mapping functions instead of traditional linear transformers. The works [12,13] use deep autoencoders to learn

* Corresponding author.
E-mail addresses: jingliu_sxu@hotmail.com (J. Liu), cfy@sxu.edu.cn (F. Cao), ljy@sxu.edu.cn (J. Liang).

https://doi.org/10.1016/j.ins.2022.07.093
0020-0255/© 2022 Elsevier Inc. All rights reserved.

a shared representation between multiple views by jointly minimizing the reconstruction loss of each view. However, since there is no clustering objective integrated into the deep representation learning process, the learned representations may not be cluster-discriminative, which may lead to unsatisfactory clustering performance. Recent works show that jointly conducting multi-view deep representation learning and clustering can yield better performance. For example, Xie [14] proposed a joint deep multi-view clustering method to optimize the fused multi-view representation simultaneously with a self-training clustering objective by hardening the soft assignment distribution of the representation toward a common auxiliary target distribution. Xu [15] proposed a deep multi-view clustering collaborative training that sequentially uses each view's auxiliary target distribution to refine the deep representation and soft clustering assignments of other views. Du [16] proposed a deep multi-autoencoder based clustering method that designs a cross entropy based regularization to guarantee consistency as well as complementarity between any two views. Sun [17] proposed a deep multi-view subspace clustering method that uses the cluster assignments obtained in spectral clustering to self-supervise the learning of deep representations and the self-expression coefficients.

Although existing deep MVC methods have achieved significant progress, deep learning is not organically integrated into the multi-view clustering process. For the two-step methods, deep learning just acts as a representation tool to produce better representations and is not truly involved in the clustering process. For the joint learning methods, the deep representation learning in most works is guided by pseudo cluster assignments (either soft or hard assignments) and in turn refines the cluster assignments. The cluster assignments acting as the guiding labels cannot reflect any semantic information of the cluster distribution, which makes the deep representation learning process more like a "black box" lacking the interpretability.

For this problem, inspired by the work [18] on single-view deep clustering that jointly performs deep representation learning and *k*-means clustering, we propose a centroids-guided deep multi-view *k*-means clustering method (CDMKM). In this method, the deep representation learning is organically incorporated into the multi-view *k*-means objective by using the cluster centroids in multi-view *k*-means to guide the deep learning of each view. In turn, more *k*-means-friendly representations are produced to further optimize the multi-view *k*-means objective. The framework of the proposed method is presented in Fig. 1. To obtain the well-trained initial representations, the network of each view is pretrained with a deep autoencoder by using only reconstruction loss. Then, in the deep multi-view *k*-means phase, our method is conducted in an end-to-end recurrent manner. In the forward direction, multi-view *k*-means clustering is performed on the deep representations of each view. In the backward direction, the view-specific cluster centroids obtained in multi-view *k*-means guide the deep representation learning of each view. The cluster centroids of each view obtained under a common clustering partition not only represent the semantic information of the clusters but also imply consistency among different views. By using the cluster centroids as the supervised information to reduce the loss between each representation and its assigned cluster centroid with respect to the network parameters of each view, the new representations of each view tend to be more *k*-means-friendly toward a common partition. To avoid the representation capability of the autoencoders being slowly weakened by the clustering loss in the training process, the centroids-guided training for each view is jointly optimized with the reconstruction loss. Experiments on several datasets demonstrate the effectiveness of the proposed method.

The main contributions of this paper are summarized as follows:

- We propose a novel centroids-guided deep multi-view *k*-means clustering model, which organically integrates deep learning and multi-view *k*-means into a unified framework. The proposed model can effectively use the cluster centroids obtained in multi-view *k*-means to guide the deep clustering.
- Compared with most existing deep MVC methods that use cluster labels to guide the deep learning, the cluster centroids used as the guiding signal in our model can possess not only the cluster discriminative information but also the semantic information of the clusters, which makes the clustering process more interpretable.
- Compared with the state-of-the-art shallow and deep MVC methods, our method achieves superior performance on several popular datasets, which demonstrates the effectiveness of the proposed CDMKM framework.

The rest of the paper is organized as follows. Related works on multi-view clustering and deep clustering are reviewed in Section 2. Details of the proposed method are presented in Section 3. The experimental results are shown in Section 4. Finally, the conclusions are given in Section 5.

2. Related Works

In this section, we give a brief review of the recent works on deep clustering and multi-view clustering, which are the two areas that are most related to our work.

2.1. Deep clustering

With the success of deep learning used in supervised learning, in recent years, deep learning has begun to be used in clustering (unsupervised learning). Recent works show that deep clustering by integrating the deep representation learning and clustering process into a joint learning process yields better results than conducting them separately. Xie [19] proposed a deep embedded clustering method to refine the clusters by iteratively optimizing the KL (Kullback Leibler) divergence of

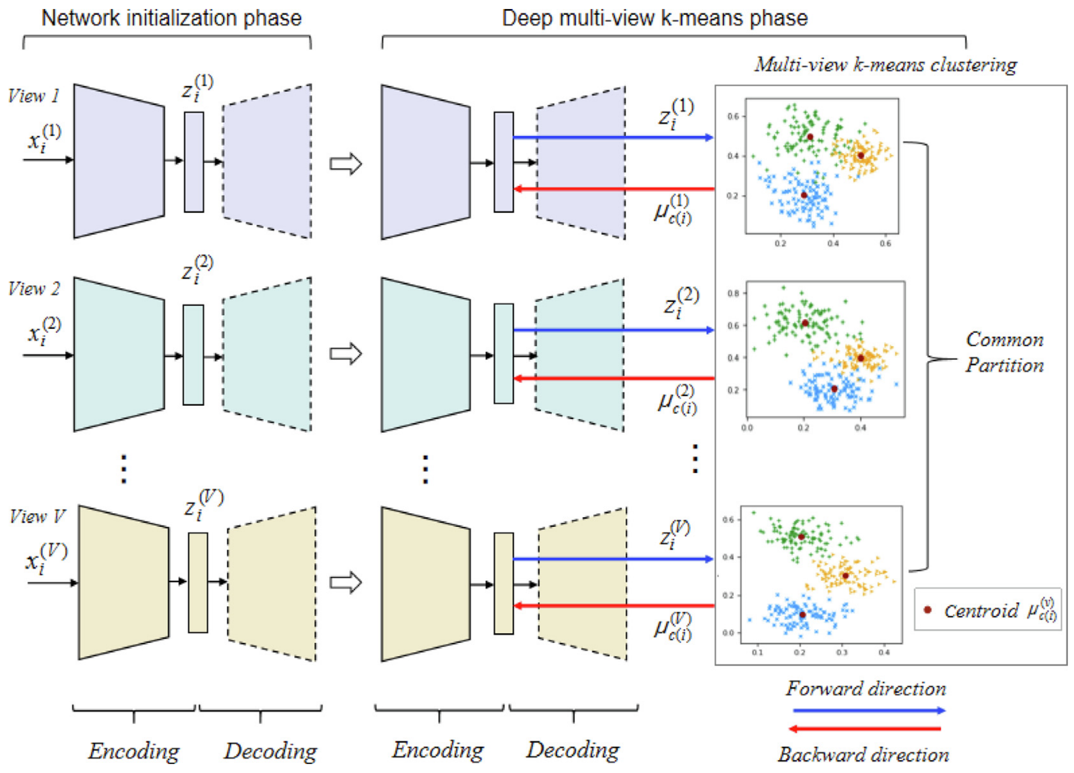


Fig. 1. The framework of CDMKM. (1) Network initialization phase: the autoencoder of each view is pretrained using only reconstruction loss. (2) Deep multi-view k-means phase: multi-view k-means clustering is performed on the deep representations $\boldsymbol{z}_i^{(v)}$ in the forward direction, while in the backward direction, the view-specific cluster centroids $\boldsymbol{\mu}_{c(i)}^{(v)}$ obtained in multi-view k-means guide the deep representation learning, which is optimized together with reconstruction loss.

a defined soft cluster assignment to a self-training target distribution. Guo [20] proposed a deep convolutional embedded clustering method by using a convolutional autoencoder to perform deep embedded clustering on image data. Ji [21] introduced a novel deep autoencoder framework with a novel self-expressive layer at the junction between the encoder and the decoder to learn the deep nonlinear self-expressive coefficients for the subspace clustering. Yang [22] proposed an end-to-end deep clustering model for jointly performing agglomerative clustering and deep representation learning in a recurrent framework. Jiang [23] proposed a variational deep embedding clustering method that integrates the variational autoencoder [24] and Gaussian Mixture Model into a unified generative clustering process. Yang [18] proposed a deep clustering network that jointly performs dimension reduction and k-means clustering, where dimension reduction is conducted by a deep autoencoder. In recent years, deep clustering has been extended into the field of multi-view clustering. The related works on deep MVC are summarized in the ”Deep multi-view clustering methods” part of Section 2.2.

2.2. Multi-view clustering

Shallow multi-view clustering methods. In the past decade, various MVC methods have been proposed. In the earlier years, MVC methods were mainly based on shallow approaches, and most of them were proposed based on the general idea of jointly learning multiple views while maintaining the consistency among views by using common cluster-discriminative information. Based on different kinds of cluster-discriminative information, a variety of shallow MVC methods are derived. For spectral clustering based methods [25–28], the common cluster-discriminative information is expressed in terms of the common eigenvector matrix, which is optimized by jointly using the Laplacian matrix of each view. For subspace clustering based methods [6,7,29], the common cluster-discriminative information is represented as the common coefficient matrix for jointly self-expressing the sample vectors of each view. For NMF based methods [2–5] and k-means based methods [30–33], the common cluster-discriminative information among views is expressed as the common indicator matrix, which is optimized by jointly updating the basis matrix (cluster centroids for k-means based method) of each view. Our proposed method is derived from the idea of k-means based multi-view clustering methods that have already been applied in some existing shallow MVC methods. For example, Tzortzis [30] proposed a weighted multi-view kernel k-means clustering method that optimizes a weighted combination of the loss of kernel k-means in each view under a common clustering partition. Cai [31] proposed a robust multi-view k-means clustering method that jointly optimizes the sparsity-inducing norm of k-means loss in each view under a common cluster indicator matrix across views. Xu [32] proposed weighted multi-view k-means clustering with feature selection that weights the views and the features in a unified process of multi-view k-means clustering. Liu [33] proposed a cluster-weighted multi-view kernel k-means method that jointly learns the clustering results and the weights assigned to the corresponding clusters among views.

Deep multi-view clustering methods. With the progress of deep learning used in unsupervised learning, deep approach based MVC methods have been proposed in recent years. Deep MVC methods can be classified into two categories: separated learning methods and joint learning methods. For the separated learning methods, deep multi-view representation and clustering are performed separately. The works in this category usually focus on the design of deep multi-view representation learning methods then followed by traditional clustering methods. Andrew [10] proposed deep CCA by using deep neural networks as the mapping function to simultaneously learn deep nonlinear mappings of two views that are maximally correlated. Wang [11] proposed deep CCA based on autoencoders, which is an improved version of deep CCA with reconstruction networks from the latent space and additional reconstruction losses. Ngiam [34] proposed bimodal autoencoders to reconstruct both audio and video views by minimizing the reconstruction loss of the two input views and reconstructed representation. Zhang [35] proposed an autoencoder in autoencoder network, which automatically maps different views into a common representation while adaptively balancing the consistency and complementarity among multiple views. More works about deep multi-view representation learning methods can be found in the review [36]. For the joint learning methods, deep multi-view representation and clustering are jointly performed in a unified framework. Most works follow the basic idea of single-view deep embedded clustering (DEC) [19] and extend the idea to the multi-view field by designing different kinds of multi-view fusion mechanisms, such as the multi-view weighted fusion mechanism in [14], the multi-view collaborative training mechanism in [15], and the multi-view pairwise regularized mechanism in [16]. Different from the DEC based methods that use the soft pseudo labels to guide the deep representation learning, we incorporate the deep representation learning into the multi-view k-means clustering process and use cluster centroids to guide the deep representation learning.

3. The Proposed Method

In this section, we present the overall framework, problem formulation and optimization strategy of the proposed CDMKM method in detail.

3.1. Framework of CDMKM

As shown in Fig. 1, the proposed framework has two phases: the network initialization phase and the optimization phase. Specifically, multiple autoencoders are used for multiple views, with each autoencoder corresponding to one view. In the network initialization phase, each autoencoder is pretrained by the reconstruction loss to obtain good initial representations. This lays a good foundation for subsequent joint learning to iteratively obtain more precise clusters and more clustering-friendly representations. Then, in the optimization phase, deep multi-view k-means is optimized alternately in the forward direction and backward direction. In the forward direction, multi-view k-means is performed on the deep representations $\left\{\boldsymbol{z}_1^{(v)}, \boldsymbol{z}_2^{(v)}, \ldots, \boldsymbol{z}_N^{(v)}\right\}_{v=1}^{V}$. In the backward direction, the view-specific cluster centroids $\boldsymbol{\mu}_{c(i)}^{(v)}$ obtained in multi-view k-means are used to guide the training of the view-specific encoder. To maintain the representation capability of the autoencoders, the centroids-guided training for each view is jointly optimized with the reconstruction loss. More details about the proposed method are described in the following sections. The main mathematical notations used in the paper are listed in Table 1.

3.2. Problem Formulation

Consider a multi-view dataset consisting of N instances represented by V different views, which are to be partitioned into K clusters. The dataset can be denoted by $\left\{\boldsymbol{x}_1^{(v)}, \boldsymbol{x}_2^{(v)}, \ldots, \boldsymbol{x}_N^{(v)}\right\}_{v=1}^{V} \in \mathbb{R}^{d^{(v)}}$, where $\boldsymbol{x}_i^{(v)}$ represents the i-th instance from the v-th view and $d^{(v)}$ is the feature dimensionality of the v-th view. The original data are transformed into the deep representations $\left\{\boldsymbol{z}_1^{(v)}, \boldsymbol{z}_2^{(v)}, \ldots, \boldsymbol{z}_N^{(v)}\right\}_{v=1}^{V} \in \mathbb{R}^{m^{(v)}}$ by the deep nonlinear mapping $f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right) : \mathbb{R}^{d^{(v)}} \to \mathbb{R}^{m^{(v)}}$, where $\boldsymbol{\Theta}_v$ represents the network parameters of the encoder part in the v-th view and $\boldsymbol{z}_i^{(v)} = f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right)$. Following the general approach of multi-view k-means clustering [30–33] that jointly optimizes a weighted combination of the loss of k-means in each view under common clustering partition, we introduce a novel deep multi-view k-means clustering method by solving the following objective function

Table 1
List of mathematical notations.

Symbol	Description
N	number of samples
V	number of views
K	number of clusters
$\boldsymbol{x}_i^{(v)}$	the i-th original sample in the v-th view
$\boldsymbol{z}_i^{(v)}$	the i-th deep representation in the v-th view
$f_{\boldsymbol{\Theta}_v}^{(v)}(\cdot)$	transformation of the encoder part in the v-th view
$g_{\boldsymbol{\Omega}_v}^{(v)}(\cdot)$	transformation of the decoder part in the v-th view
$\boldsymbol{\mu}_k^{(v)}$	centroid of the k-th cluster in the v-th view
$\boldsymbol{\mu}_{c(i)}^{(v)}$	centroid of the cluster in the v-th view to which the i-th representation is assigned
ω_v	weight for the v-th view
p	hyperparameter for controlling the distribution of the view weights
$\mathscr{L}_c^{(v)}$	centroids-guided training loss of the v-th view
$\mathscr{L}_r^{(v)}$	reconstruction loss of the v-th view
$g_1(\cdot)$	activation function for the encoding layer
$g_2(\cdot)$	activation function for the decoding layer

$$\begin{aligned}&\min_{\left\{\boldsymbol{\mu}_1^{(v)},\ldots,\boldsymbol{\mu}_K^{(v)},\omega_v,\boldsymbol{\Theta}_v\right\}_{v=1}^{V},\{c(i)\}_{i=1}^{N}} O=\sum_{v=1}^{V}\omega_v^p\sum_{i=1}^{N}\|f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right)-\boldsymbol{\mu}_{c(i)}^{(v)}\|^2,\\ &s.t.\quad \omega_v>0,\sum_{v=1}^{V}\omega_v=1,p>1,\end{aligned} \tag{1}$$

where $\boldsymbol{\mu}_k^{(v)}$ is the centroid of the k-th ($k\in\{1,2,\ldots,K\}$) cluster in the v-th view, $c(i)\in\{1,2,\ldots,K\}$ represents the cluster label of the i-th instance, and $\boldsymbol{\mu}_{c(i)}^{(v)}$ is the centroid of the cluster to which the mapping $f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right)$ has been assigned. Here, ω_v is the view weight for the v-th view, and the exponent p is a hyperparameter used to control the distribution of the view weights. This objective function can be regarded as two sub-objective functions: the clustering objective function with respect to $\boldsymbol{\mu}_k^{(v)}, c(i)$ and ω_v in the forward direction, and the loss function for deep networks with respect to $\boldsymbol{\Theta}_v$ in the backward direction.

In the forward direction: when fixing the deep network parameters $\boldsymbol{\Theta}_v$, Eq. (1) is equivalent to applying the weighted multi-view k-means clustering on the output deep representations $\left\{\boldsymbol{z}_1^{(v)},\boldsymbol{z}_2^{(v)},\ldots,\boldsymbol{z}_N^{(v)}\right\}_{v=1}^{V}$, as the following objective function

$$\begin{aligned}&\min_{\left\{\boldsymbol{\mu}_1^{(v)},\ldots,\boldsymbol{\mu}_K^{(v)},\omega_v\right\}_{v=1}^{V},\{c(i)\}_{i=1}^{N}} O=\sum_{v=1}^{V}\omega_v^p\sum_{i=1}^{N}\|\boldsymbol{z}_i^{(v)}-\boldsymbol{\mu}_{c(i)}^{(v)}\|^2,\\ &s.t.\quad \omega_v>0,\sum_{v=1}^{V}\omega_v=1,p>1.\end{aligned} \tag{2}$$

The view weight ω_v determines the importance of the v-th view and is learned based on the loss of each view. For the i-th sample, its assignment loss to each cluster k is computed by the weighted sum of squared distances between $\boldsymbol{z}_i^{(v)}$ and $\boldsymbol{\mu}_k^{(v)}$ in all views ($v=1,2,\ldots,V$), and the cluster with the lowest assignment loss is selected as the final cluster assignment $c(i)$.

In the backward direction: when fixing the parameters $\boldsymbol{\mu}_k^{(v)}, c(i)$ and ω_v, Eq. (1) can be regarded as a combination of the loss function for the deep network of each view with respect to the network parameters $\boldsymbol{\Theta}_v$ by using the view-specific cluster centroid $\boldsymbol{\mu}_{c(i)}^{(v)}$ as the supervisory signal, which can be written as the following form

$$\min_{\{\boldsymbol{\Theta}_v\}_{v=1}^{V}} O=\sum_{v=1}^{V}\omega_v^p\sum_{i=1}^{N}L_c^{(v)}\left(f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right),\boldsymbol{\mu}_{c(i)}^{(v)}\right), \tag{3}$$

where $\mathscr{L}_c^{(v)}(\cdot):\mathbb{R}^{m^{(v)}}\to\mathbb{R}$ is a squared loss function that measures the loss between the deep mapping $f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right)$ and its current assigned cluster centroid $\boldsymbol{\mu}_{c(i)}^{(v)}$. Since ω_v is a constant here, minimizing Eq. (3) is equivalent to minimizing the loss function of each view separately. For each view, the network is trained with the following loss function

$$\min_{\boldsymbol{\Theta}_v} O^{(v)}=\sum_{i=1}^{N}\mathscr{L}_c^{(v)}\left(f_{\boldsymbol{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right)\right),\boldsymbol{\mu}_{c(i)}^{(v)}). \tag{4}$$

To avoid the representation capability of the autoencoder being slowly weakened by the centroids-guided loss $\mathscr{L}_c^{(v)}$ in the training process, we jointly minimize $\mathscr{L}_c^{(v)}$ with the reconstruction loss $\mathscr{L}_r^{(v)}$ by solving the following objective

$$\min_{\mathbf{\Theta}_v,\mathbf{\Omega}_v} \hat{O}^{(v)} = \sum_{i=1}^{N} \mathscr{L}_c^{(v)}\left(f_{\mathbf{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right), \boldsymbol{\mu}_{c(i)}^{(v)}\right) + \lambda \sum_{i=1}^{N} \mathscr{L}_r^{(v)}\left(g_{\mathbf{\Omega}_v}^{(v)}\left(f_{\mathbf{\Theta}_v}^{(v)}\left(\boldsymbol{x}_i^{(v)}\right)\right), \boldsymbol{x}_i^{(v)}\right), \tag{5}$$

where $g_{\mathbf{\Omega}_v}^{(v)}(\cdot) : \mathbb{R}^{m^{(v)}} \to \mathbb{R}^{d^{(v)}}$ is the mapping function of the decoder part from the representation layer to the reconstruction layer, and $\mathbf{\Omega}_v$ represents the network parameters of the decoder part in the v-th view. $\lambda > 0$ is a trade-off parameter balancing the two losses. Incorporating the reconstruction loss into the view-specific training of the autoencoder does not affect the monotonically decreasing trend of the total objective of Eq. (1), which is demonstrated in the experiments section as shown in Fig. 3.

Since the cluster centroid $\boldsymbol{\mu}_{c(i)}^{(v)}$ of each view is obtained under the common cluster label $c(i)$, it not only represents the cluster-discriminative information but also implies the consistency among different views. By using it as the supervised information to minimize the loss with respect to the network parameters, the distance between each representation and its assigned cluster centroid will be smaller, which makes the new representations more k-means-friendly toward a common partition. Therefore, applying multi-view k-means on the new representations would lead to better clustering results.

3.3. Optimization

Before optimizing the proposed objective, we pretrain the autoencoder of each view by using only the reconstruction loss to obtain initial representations. Then, cluster centroids, assignments and view weights are well initialized based on the initial representations. To optimize the proposed objective function of Eq. (1), we alternately optimize the sub-objective function of Eq. (3) in the backward direction and the sub-objective function of Eq. (2) in the forward direction. For the sub-objective function of Eq. (2), each parameter is also updated alternately. The initialization details and optimization procedures are described in the following sections.

Initialization: The well-trained initial representations can result in relatively good initial clustering results, which may set up a good starting point for the subsequent learning to iteratively obtain more precise clusters. To make the initial representations well represent the original data, we pretrain the network of each autoencoder with the reconstruction loss only. The autoencoder is trained by the layer-wise training method, in which the first layer is trained by the reconstruction loss of the input layer, and each following layer is trained by reconstructing the output of the previous learned layer. Suppose the output of the previous layer is h, the parameters of the next layer are trained by minimizing the following reconstruction loss

$$L(w_1, w_2) = \|g_2(g_1(h; w_1); w_2) - h\|^2, \tag{6}$$

where g_1 and g_2 are activation functions for the encoding and decoding layers respectively, and w_1 and w_2 are model parameters of the encoding and decoding layers respectively.

After pretraining the autoencoder of each view, to make the initial cluster centroids more discriminative and effective for supervising the training of the network at the beginning of the optimization phase, we initialize the cluster centroids $\boldsymbol{\mu}_k^{(v)}$ and assignments $c(i)$ by performing multi-view k-means (using the objective function of Eq. (2) without weights) jointly on the initial representations of each view. The weight ω_v of each view is initialized as $1/V$.

Updating $\mathbf{\Theta}_v$ in the backward direction: for the fixed $\boldsymbol{\mu}_k^{(v)}, c(i)$ and ω_v, the objective function of Eq. (3) is optimized by minimizing the loss function of Eq. (5) with respect to the network parameters $\mathbf{\Theta}_v$ and $\mathbf{\Omega}_v$ of each view. The network parameters are updated by using stochastic gradient descent (SGD) with the gradients computed by backpropagation algorithm. Specifically, we use the Adam optimizer [37] in the training process.

Updating $\boldsymbol{\mu}_k^{(v)}, c(i)$ and ω_v in the forward direction: for the fixed network parameters $\mathbf{\Theta}_v$, the parameters $\boldsymbol{\mu}_k^{(v)}, c(i)$ and ω_v are alternately updated on the learned representations $\left\{\boldsymbol{z}_1^{(v)}, \boldsymbol{z}_2^{(v)}, \ldots, \boldsymbol{z}_N^{(v)}\right\}_{v=1}^{V}$.

First, fixing $c(i)$ and ω_v, the cluster centroid of each view $\boldsymbol{\mu}_k^{(v)}$ is updated by

$$\boldsymbol{\mu}_k^{(v)} = \frac{\sum_{i=1}^{N} \delta_{ik} \boldsymbol{z}_i^{(v)}}{\sum_{i=1}^{N} \delta_{ik}}, \tag{7}$$

where $\delta_{ik} = 1$ if $c(i) = k$ and $\delta_{ik} = 0$ otherwise.

Before updating the cluster labels, in order to make the distances in feature spaces of different views to be comparable, the representations of different views must be normalized to the same level of magnitude. We use L2-norm normalization approach to normalize $\boldsymbol{z}_i^{(v)}$ and denote the normalized representations as $\left\{\tilde{\boldsymbol{z}}_1^{(v)}, \tilde{\boldsymbol{z}}_2^{(v)}, \ldots, \tilde{\boldsymbol{z}}_N^{(v)}\right\}_{v=1}^{V}$. To update the cluster labels in the normalized feature space, we first calculate the cluster centroid $\tilde{\boldsymbol{\mu}}_k^{(v)}$ in the normalized feature space by

$$\tilde{\boldsymbol{\mu}}_k^{(v)} = \frac{\sum_{i=1}^{N} \delta_{ik} \tilde{\boldsymbol{z}}_i^{(v)}}{\sum_{i=1}^{N} \delta_{ik}}. \tag{8}$$

Then, fixing $\tilde{\boldsymbol{\mu}}_k^{(v)}$ and ω_v, the cluster label $c(i)$ is updated by assigning the normalized representation $\tilde{\boldsymbol{z}}_i^{(v)}$ to the cluster with the lowest loss, which is computed by a weighted sum of the squared distances between $\tilde{\boldsymbol{z}}_i^{(v)}$ and $\tilde{\boldsymbol{\mu}}_k^{(v)}$ in different views. Thus, $c(i)$ can be obtained by

$$c(i) = \underset{k}{\operatorname{argmin}} \sum_{v=1}^{V} \omega_v^p \|\tilde{\boldsymbol{z}}_i^{(v)} - \tilde{\boldsymbol{\mu}}_k^{(v)}\|^2. \tag{9}$$

Finally, fixing $\boldsymbol{\mu}_k^{(v)}$ and $c(i)$, the view weight ω_v can be updated by using the Lagrangian multiplier method. We denote the sum-of-squared loss of the v-th view as

$$D_v = \sum_{i=1}^{N} \|\boldsymbol{z}_i^{(v)} - \boldsymbol{\mu}_{c(i)}^{(v)}\|^2. \tag{10}$$

Then, we obtain the Lagrangian formula of Eq. (2) with respect to ω_v and γ as the following function

$$L(\omega_v, \gamma) = \sum_{v=1}^{V} \omega_v^p D_v + \gamma \left(\sum_{v=1}^{V} \omega_v - 1 \right). \tag{11}$$

Taking the derivative with respect to ω_v yields

$$\frac{\partial L(\omega_v, \gamma)}{\partial \omega_v} = p\omega_v^{p-1} D_v + \gamma. \tag{12}$$

Setting this derivative to zero, combined with the constraint on the weights, we obtain the equation set

$$\begin{cases} p\omega_v^{p-1} D_v + \gamma = 0, \\ \sum_{v=1}^{V} \omega_v = 1. \end{cases} \tag{13}$$

Solving this equation set with respect to ω_v, the closed-form solution of ω_v is obtained as following expression

$$\omega_v = \frac{1}{\sum_{v'=1}^{V} \left(\frac{D_v}{D_{v'}} \right)^{\frac{1}{p-1}}}, \quad p > 1. \tag{14}$$

In this expression, we can see that the smaller the view loss D_v is, the larger the view weight ω_v will be. A smaller D_v reflects a better clustering partition in this view, which, by being assigned a larger weight in multi-view clustering, boosts the overall clustering performance.

We summarize the above optimization process in Algorithm 1.

Algorithm 1: The *CDMKM*.

```
Input: Multi-view dataset $\left\{\boldsymbol{x}_1^{(v)}, \ldots, \boldsymbol{x}_N^{(v)}\right\}_{v=1}^{V}$, number of clusters $K$, exponent $p$, trade-off parameter $\lambda$, number of
  training epochs $E$.
Output: Cluster labels $\{c(i)\}_{i=1}^{N}$.
Initialize: Networks parameters pre-trained with reconstruction loss; Initial cluster centroids $\left\{\boldsymbol{\mu}_1^{(v)}, \ldots, \boldsymbol{\mu}_K^{(v)}\right\}_{v=1}^{V}$ and
  assignments $\{c(i)\}_{i=1}^{N}$ on the initial deep representations; View weights $\omega_v = 1/V$.
Method:
1: for $e = 1: E$ do
2:   for $v = 1: V$ do
3:     Update the network parameters $\{\boldsymbol{\Theta}_v, \boldsymbol{\Omega}_v\}$ by minimizing the Eq. (5) via backpropagation algorithm.
4: end for
```

a (*continued*)

Algorithm 1: The *CDMKM*.

```
 5: Compute $\left\{\mathbf{z}_1^{(v)},\ldots,\mathbf{z}_N^{(v)}\right\}_{v=1}^{V}$ with updated $\mathbf{\Theta}_v$;
 6:   Update $\boldsymbol{\mu}_k^{(v)}$ by Eq. (7);
 7:   Normalize $\mathbf{z}_i^{(v)}$ to $\tilde{\mathbf{z}}_i^{(v)}$ and compute $\tilde{\boldsymbol{\mu}}_k^{(v)}$ by Eq. (8);
 8:   Update the cluster labels $\{c(i)\}_{i=1}^N$ by Eq. (9);
 9:   Update the view weights $\omega_v$ by Eq. (14).
10: end for
```

4. Experiments

We implement the proposed method in Python with TensorFlow 1.12.0 and evaluate its performance on six real datasets. We first describe the experimental setup and then present the experimental results in detail.

4.1. Experimental Setup

4.1.1. Datasets Description

Mfeat [1]: This dataset consists of multiple published features extracted from 2000 images of handwritten digits (0–9), with 200 images per digit. We use three published features to construct the multi-view dataset: Fourier coefficients of the character shapes, profile correlations, and pixel averages in 2×3 windows.

Caltech101-20[2]: This dataset is extracted from Caltech101 which consists of 101 categories of images. We select widely-used 20 categories that contain 2386 images, which are represented by five types of features: gabor features, wavelet moments, centrist features, HOG (histogram of oriented gradients) features, gist features, and LBP (local binary patterns) features.

Scene: This dataset [38] contains 2688 outdoor scene images over 8 categories: coast, mountain, forest, street, inside city, open country, highways and buildings. For each image, three different visual features including gist features, color moments and HOG features are extracted as three views.

Corel: This dataset [47] contains 34 categories and each category has 100 images. Each of the images has a salient foreground object that belongs to one of these 34 categories. For each image, three different visual features including color histogram, color coherence and wavelet texture are extracted as three views.

NUS-WIDE-OBJ[3]: We use the training set of NUS-WIDE-OBJECT dataset, which contains 17928 object images over 31 categories. We delete the images that have multiple labels, and the remaining dataset contains 14270 images. Each image is represented by five types of low-level features: color histogram, color moments, color correlogram, edge direction histogram, and wavelet texture.

MNIST&USPS: These are two image datasets of handwritten digits (0–9) from two sources, i.e., MNIST [39] and USPS [40]. We select 10000 images from each source, with 1000 images randomly selected per digit, and regard images from individual sources as two views of the same digit.

Detailed information about these six datasets is summarized in Table 2.

4.1.2. Network Settings

For the datasets Mfeat, Caltech101-20, Scene, Corel and NUS-WIDE-OBJ, whose features are vector-based, we use the DNN (deep neural network) architecture for the autoencoder of each view. For the image dataset MNIST&UPSP, we use the CNN (convolutional neural network) architecture for the autoencoder of each view. The settings of the network architectures for the six datasets are shown in Table 3. For all datasets, we use ReLU (rectified linear unit) [41] as the activation function for the network. The network training has two phases, i.e., the pretraining phase and the fine-tuning phase. In both training phases, for all datasets, we use the Adam optimizer [37] with the learning rate $lr = 0.001$ to train the networks, and set the batch size to approximately $0.01 \times N$. In the pretraining phase, the DNN-based networks for the first five datasets are pretrained for 100 epochs, and the CNN-based network for MNIST&UPSP is pretrained for 200 epochs. In the fine-tuning phase, the number of training epochs is set to 200 for Mfeat and 50 for all other datasets. The hyperparameters p and λ are set to $p = 16$ for all datasets, $\lambda = 1$ for the DNN-based networks for the first five datasets, and $\lambda = 0.01$ for the CNN-based network for MNIST&USPS, based on the analysis in Section 4.6. For the cluster centroids initialization in the fine-tuning phase, we randomly select K samples as the initial cluster centroids, and perform multi-view k-means on the representations output from the pretrained networks to obtain the initial cluster centroids for the fine-tuning phase.

[1] https://archive.ics.uci.edu/ml/datasets/Multiple + Features
[2] http://www.vision.caltech.edu/Image_Datasets/Caltech101/Caltech101.html
[3] https://lms.comp.nus.edu.sg/wp-content/uploads/2019/research/nuswide/NUS-WIDE.html

Table 2
Details of the datasets.

Datasets	#Size	#Class	#View	$d^{(1)}$	$d^{(2)}$	$d^{(3)}$	$d^{(4)}$	$d^{(5)}$	$d^{(6)}$
Mfeat	2000	10	3	76	216	240	-	-	-
Caltech101-20	2386	20	6	48	40	254	1984	512	928
Scene	2688	8	3	512	432	256	-	-	-
Corel	3400	34	3	64	128	104	-	-	-
NUS-WIDE-OBJ	14270	31	5	65	226	145	74	129	-
MNIST&USPS	10000	10	2	28×28	16×16	-	-	-	-

Table 3
Network architecture of the encoder of each view for each dataset. The architecture of the decoder is symmetric with the encoder. For the first five datasets that use DNN architecture, each number represents the wide of each hidden layer. For the last dataset that uses CNN architecture, conv (a×a×b) represents the convolutional layer with kernel size = a and channel number = b (stride = 2 is set as default), and fc(10) represents the fully connected layer with the dimension reduced to 10.

Datasets	Network architecture of the encoder of each view
Mfeat	100–50–30
Caltech101-20	100–50–30
Scene	200–100–50
Corel	100–50–30
NUS-WIDE-OBJ	200–100–50
MNIST&USPS	conv(5 × 5 × 32)-conv(5 × 5 × 64)-conv(3 × 3 × 128)-fc(10)

4.1.3. Baseline Methods

The proposed method is compared with several baseline methods. First, we compare our method with some representative state-of-the-art multi-view clustering methods including the NMF based method **GNMF** [42], the subspace clustering based method **LMSC** [43], the spectral clustering based methods **CoregSC** [26] and **AWGL** [28], the deep clustering based method **DEMVC** [15] and **DMJC**[14], and the CCA based deep representation methods **DCCA**[10] and **DCCAE**[11]. To illustrate the superiority of our method compared with single-view deep clustering methods, we compare our method with the single-view deep k-means method **DCN**[18], which is the method that our method is based on.

For the shallow clustering methods GNMF, LMSC, CoregSC and AWGL, we follow the experimental settings in their papers. For the deep clustering methods DEMVC, DMJC and DCN, for fair comparison, the same initialized autoencoder for each view as our method is used, and other hyperparameters in the fine-tuning phase are set following their papers. The single-view method DCN is conducted on each view, and the results of the view with the best performance are reported. For the deep representation methods DCCA and DCCAE, since they can only handle two views, we first select two views with better k-means performance as the input of the methods and then follow the settings in their papers to obtain the deep CCA representations, on which the traditional k-means is performed to obtain the final results.

Then, to further verify the superiority of our method, we compare our method with two variant methods as follows: **MV*k*-means** that conducts weighted multi-view k-means in the original space by solving the following objective function

$$\begin{aligned} &\min_{\{\boldsymbol{\mu}_1^{(v)},\ldots,\boldsymbol{\mu}_K^{(v)},\omega_v\}_{v=1}^V,\{c(i)\}_{i=1}^N} O = \sum_{v=1}^V \omega_v^p \sum_{i=1}^N \|\boldsymbol{x}_i^{(v)} - \boldsymbol{\mu}_{c(i)}^{(v)}\|^2, \\ &s.t. \quad \omega_v > 0, \sum_{v=1}^V \omega_v = 1, p > 1, \end{aligned} \tag{15}$$

and **AE + MV*k*-means** that first extracts the deep representations by autoencoders and then performs weighted multi-view k-means in a separate manner, by solving the following objective function

$$\begin{aligned} &\min_{\{\boldsymbol{\mu}_1^{(v)},\ldots,\boldsymbol{\mu}_K^{(v)},\omega_v\}_{v=1}^V,\{c(i)\}_{i=1}^N} O = \sum_{v=1}^V \omega_v^p \sum_{i=1}^N \|\boldsymbol{z}_i^{(v)} - \boldsymbol{\mu}_{c(i)}^{(v)}\|^2, \\ &s.t. \quad \omega_v > 0, \sum_{v=1}^V \omega_v = 1, p > 1, \end{aligned} \tag{16}$$

where $\boldsymbol{z}_i^{(v)}$ is the deep representation of the i-th sample extracted by the v-th autoencoder with reconstruction loss. All the mathematical notations in Eq. (15) and Eq. (16) represent the same meaning as in Section 3. For these two methods, we use the same experimental settings as the proposed method.

For fair comparisons, all methods involving k-means operation apply random initialization. And we perform all methods 10 times and report the average performance.

4.1.4. Evaluation Metrics

Clustering evaluation metrics: We employ three metrics to evaluate the clustering performance for all methods. They are normalized mutual information (NMI), accuracy (ACC) and adjusted rand index (ARI). Before computing the evaluation metrics, the assigned cluster labels are best mapped to the true labels by Munkres algorithm [44]. Suppose the true label of each sample belongs to $\{Y_1, Y_2, \ldots, Y_k\}$, where k is the number of clusters. The contingency table is given in Table 4.

Then the three evaluation metrics are defined as follows

$$NMI = \frac{-2\sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij} log\left(\frac{n_{ij}n}{n_{i\cdot}n_{\cdot j}}\right)}{\sum_{i=1}^{k} n_{i\cdot} log\left(\frac{n_{i\cdot}}{n}\right) + \sum_{j=1}^{k} n_{\cdot j} log\left(\frac{n_{\cdot j}}{n}\right)}, \tag{17}$$

$$ACC = \frac{1}{n}\sum_{i=1}^{k} n_{ii}, \tag{18}$$

$$ARI = \frac{\sum_{ij} C_{n_{ij}}^2 - \left[\sum_i C_{n_{i\cdot}}^2 \sum_j C_{n_{\cdot j}}^2\right]/C_n^2}{\frac{1}{2}\left[\sum_i C_{n_{i\cdot}}^2 + \sum_j C_{n_{\cdot j}}^2\right] - \left[\sum_i C_{n_{i\cdot}}^2 \sum_j C_{n_{\cdot j}}^2\right]/C_n^2}. \tag{19}$$

For each of them, the higher the value is, the better the clustering performance is. For all methods, we report the average results with the standard deviation from 10 executions.

Non-parametric tests: After obtaining the results for all methods on all datasets under each metric, we use non-parametric tests to test whether the improvements of the proposed method over the baseline methods are significant. We first use the Friedman test to test the difference among all methods under each metric. If there is a significant difference among all methods, we use the Nemenyi test and Wilcoxon signed-rank test to further test the difference of the proposed method against each baseline method. The detailed descriptions about each type of non-parametric test is as following.

For the Friedman test, suppose the number of methods is k and the number of datasets is N. Firstly rank the methods within each dataset, and let r_i^j be the rank of the j-th method on the i-th of N datasets. Then the average rank for the j-th method among all datasets is $R_j = \frac{1}{N}\Sigma_i r_i^j$. Then the Friedman statistic can be calculated by the following expression

$$Q = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]. \tag{20}$$

The probability distribution of Q is approximated by that of a chi-squared distribution with $k-1$ degrees of freedom. Then the probability value can be given by $P(\mathscr{X}^2 > Q)$. If the probability value is less than the significance level α (we set $\alpha = 0.1$ in our experiment), the null-hypothesis (performance of all the methods are equivalent) is rejected, demonstrating a significant difference among all methods. Then we can proceed with the post hoc tests Nemenyi test and Wilcoxon signed-rank test.

For the Nemenyi test, the test statistic for comparing the i-th and j-th method is calculated by the following expression

$$z = (R_i - R_j)/\sqrt{\frac{k(k+1)}{6N}}. \tag{21}$$

Then use the z value to find the corresponding probability from the table of normal distribution, which is then compared with the significant level α. If the probability value is less than the significant level, the performance of two methods is sig-

Table 4
The contingency table.

True label	Assigned label				
	Y_1	Y_2	$\cdots$	Y_k	Sums
Y_1	n_{11}	n_{12}	$\cdots$	n_{1k}	$n_{1\cdot}$
Y_2	n_{21}	n_{22}	$\cdots$	n_{2k}	$n_{2\cdot}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
Y_k	n_{k1}	n_{k2}	$\cdots$	n_{kk}	$n_{k\cdot}$
Sums	$n_{\cdot 1}$	$n_{\cdot 2}$	$\cdots$	$n_{\cdot k}$	n

nificantly different. We also plot the critical distance (CD) diagram for the Nemenyi test. In the CD diagram, the average rank of each method is marked along an axis (the smaller the better), and any two methods are significantly different if the distance of their average ranks exceeds the following critical distance

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \tag{22}$$

where the critical value q_α is based on the Studentized range statistic divided by $\sqrt{2}$. Groups of methods that are not significantly different are connected.

For the Wilcoxon signed-rank test, to compare two methods, first calculate the difference between the performance of two methods on each dataset, which is denoted by D_i for the i-th of N datasets. Then sort $|D_1|, \ldots, |D_N|$ and assign ranks $R_1, \ldots, R_N$. Now reassign the symbol "+" or "$-$" to each of the N ranks R_i, depending on whether D_i was originally positive or negative. Let sgn denote the sign function: $sgn(x) = 1$ if $x > 0$ and $sgn(x) = -1$ if $x < 0$. The test statistic is the signed-rank W defined by

$$W = \sum_{i=1}^{N} sgn(D_i) R_i. \tag{23}$$

Then use the statistical software or check the distribution table of the Wilcoxon signed-rank test to obtain the probability value. If the probability value is less than the significant level α, the performance of two methods is significantly different.

4.2. Clustering Performance Comparisons

The clustering performance of all methods on six datasets is shown in Tables 5–10. The best results of all methods on each dataset are indicated in bold, and the second best results are indicated with underline. It can be seen that the proposed method outperforms all baseline methods on most datasets, except that some metrics of the proposed method on Caltech101-20, NUS-WIDE-OBJ and MNIST&USPS are slightly lower than those of DEMVC and DMJC. For the dataset NUS-WIDE-OBJ, since the number of classes is very large and the data distribution is inherently poor, the clustering results for all methods are very low. In this case, the metrics NMI and ARI make more sense than ACC because the calculations for NMI and ARI have no need to map the cluster labels to the true labels and thus are not influenced by inappropriate label mapping. Therefore, although the ACC metric of our method is slightly lower than that of DEMVC, the more convincing metrics NMI and ARI of our method are much higher than those of DEMVC, which still demonstrates that the clustering performance of our method is overall better than that of DEMVC. For the image dataset MNIST&USPS, since DMJC was originally proposed for image clustering, it performs well on this image dataset and slightly outperforms our method under ACC and ARI metrics. Even so, our method still achieves comparable performance with DMJC on this dataset, and greatly outperforms it on all other vector-based datasets, demonstrating that our method has more universal applicability than DMJC. Additionally, on most datasets except for NUS-WIDE-OBJ, multi-view k-means on deep representations (AE + MVk-means) obtains better performance than in the original data space (MVk-means). Furthermore, on all datasets, our method by jointly conducting deep representation learning and multi-view k-means in an end-to-end manner substantially improves the clustering performance over AE + MVk-means that performs deep representation extraction and multi-view k-means in a separate manner.

We also conduct the non-parametric tests for different methods under each metric. First, the Friedman test is used to test the difference among all methods. The probability values attained when conducting the Friedman test for NMI, ACC and ARI are $3.217 \times 10^{-11}, 2.685 \times 10^{-11}$ and 4.220×10^{-11} respectively, which are all very small, demonstrating a significant difference among the different methods. Then, it is necessary to further test the significance degree of the improvements of the proposed method over other methods. So we conduct the Nemenyi test and Wilcoxon signed-rank test for the proposed method over other methods under each metric, and the corresponding probability values are shown in Table 11. From the table, we find that for the Nemenyi test, our method has significant improvements over DCCA, DCCAE and DCN under all metrics, and LSMC, CoregSC and AWGL under NMI metric. To further illustrate the performance ranking relationship among all methods, we plot the critical distance (CD) diagram of the Nemenyi test under the NMI metric which is the most informative metric that can best reflect the clustering performance. In the CD diagram shown in Fig. 2, the average rank of each method is marked along the axis. We can see that our method ranks first and significantly outperforms most shallow methods AWGL, LMSC and CoregSC, the single-view method DCN, and the double-view methods DCCA and DCCAE. The improvements of our method over some of the methods, especially for the deep multi-view methods, are not significant. There may be two reasons for this. One is because the difference in the performance among the deep approach based multi-view methods is inherently lower. The other is because the number of the compared methods is much higher than the number of the datasets, which may limit the capability of the Nemenyi test. Therefore, we also conduct the Wilcoxon signed-rank test, which is more suitable in this case. From Table 11, we can see that our method demonstrates significant improvements over all compared methods under all metrics for the Wilcoxon signed-rank test.

Table 5
Performance comparisons on Mfeat.

Method	NMI	ACC	ARI
GMNMF	<u>0.8625(0.0369)</u>	0.8416(0.0968)	0.7918(0.0822)
LMSC	0.7887(0.0113)	0.8269(0.0469)	0.7324(0.0249)
CoregSC	0.7595(0.0201)	0.8086(0.0497)	0.6930(0.0393)
AWGL	0.7641(0.0258)	0.8206(0.0508)	0.6994(0.0471)
DEMVC	0.8010(0.0173)	0.7884(0.0318)	0.7080(0.0317)
DMJC	0.8354(0.0195)	<u>0.8949(0.0505)</u>	<u>0.8103(0.0446)</u>
DCCA	0.6562(0.0137)	0.6765(0.0523)	0.5424(0.0258)
DCCAE	0.6403(0.0142)	0.6213(0.0422)	0.5006(0.0248)
DCN	0.7594(0.0219)	0.7645(0.0727)	0.6686(0.0479)
MV*k*-means	0.7453(0.0251)	0.7414(0.0463)	0.6490(0.0387)
AE + MV*k*-means	0.7683(0.0406)	0.8148(0.0683)	0.7056(0.0681)
CDMKM	**0.8705(0.0309)**	**0.9156(0.0524)**	**0.8466(0.0587)**

Table 6
Performance comparisons on Caltech101-20.

Method	NMI	ACC	ARI
GMNMF	0.6071(0.0089)	0.4390(0.0120)	<u>0.3788(0.0272)</u>
LMSC	0.4173(0.0092)	0.3640(0.0157)	0.1748(0.0066)
CoregSC	0.5803(0.0129)	0.4085(0.0316)	0.2946(0.0279)
AWGL	0.5462(0.0110)	<u>0.4527(0.0268)</u>	0.3519(0.0270)
DEMVC	0.5612(0.0202)	0.4174(0.0459)	**0.3912(0.0726)**
DMJC	0.6103(0.0207)	0.4496(0.0503)	0.3587(0.0617)
DCCA	0.6094(0.0063)	0.4094(0.015)	0.3392(0.0275)
DCCAE	<u>0.6219(0.0100)</u>	0.4126(0.0317)	0.3312(0.0280)
DCN	0.5944(0.0062)	0.4157(0.0269)	0.3111(0.0269)
MV*k*-means	0.5686(0.0208)	0.4286(0.0440)	0.3519(0.0565)
AE + MV*k*-means	0.6075(0.0229)	0.4436(0.0447)	0.3537 (0.0600)
CDMKM	**0.6312(0.0239)**	**0.4634(0.0511)**	0.3706(0.0637)

Table 7
Performance comparisons on Scene.

Method	NMI	ACC	ARI
GMNMF	0.4900(0.0161)	0.5949(0.0347)	0.3833(0.0117)
LMSC	0.5227(0.0086)	<u>0.6810(0.0364)</u>	<u>0.4533(0.0125)</u>
CoregSC	0.4709(0.0178)	0.6313(0.0428)	0.4014(0.0228)
AWGL	0.4459(0.0132)	0.5557(0.0274)	0.3577(0.0140)
DEMVC	<u>0.5454(0.0051)</u>	0.6304(0.0106)	0.4289(0.0075)
DMJC	0.5301(0.0042)	0.6618(0.0239)	0.4375(0.0094)
DCCA	0.4511(0.0204)	0.5296(0.0294)	0.3139(0.0329)
DCCAE	0.4679(0.0336)	0.5210(0.0677)	0.3408(0.0393)
DCN	0.4967(0.0151)	0.6005(0.0143)	0.3820(0.0294)
MV*k*-means	0.4468(0.0196)	0.5623(0.0332)	0.3515 (0.0219)
AE + MV*k*-means	0.4962(0.0153)	0.6087(0.0419)	0.3992(0.0250)
CDMKM	**0.5709(0.0131)**	**0.7135(0.0181)**	**0.4832(0.0207)**

Table 8
Performance comparisons on Corel.

Method	NMI	ACC	ARI
GMNMF	0.3125(0.0048)	0.2096(0.0087)	0.1174(0.0054)
LMSC	0.2914(0.0057)	0.2002(0.0077)	0.1043(0.0025)
CoregSC	0.3289(0.0064)	0.2357(0.0112)	0.1205(0.0068)
AWGL	0.3183(0.0066)	0.2145(0.0111)	0.1145(0.0081)
DEMVC	0.3138(0.0048)	0.2086(0.0073)	0.1127(0.0064)
DMJC	0.3086(0.0032)	0.2045(0.0058)	0.1128(0.0017)
DCCA	0.2943(0.0025)	0.1936(0.0044)	0.1058(0.0023)
DCCAE	0.2917(0.0020)	0.1883(0.0027)	0.1092(0.0024)
DCN	0.2961(0.0040)	0.1979(0.0062)	0.0847(0.0034)
MV*k*-means	0.3206(0.0033)	0.2305(0.0091)	0.1111(0.0052)
AE + MV*k*-means	<u>0.3333(0.0041)</u>	<u>0.2400(0.0071)</u>	<u>0.1248(0.0036)</u>
CDMKM	**0.3568(0.0079)**	**0.2590(0.0119)**	**0.1395(0.0065)**

Table 9
Performance comparisons on NUS-WIDE-OBJ.

Method	NMI	ACC	ARI
GMNMF	0.0042(0.0000)	0.1329(0.0000)	0.0000(0.0000)
LMSC	0.0883(0.0015)	0.1085(0.0011)	0.0274(0.0005)
CoregSC	0.1232(0.0027)	0.1346(0.0039)	0.0448(0.0016)
AWGL	0.1281(0.0022)	0.1378(0.0047)	0.0491(0.0029)
DEMVC	0.1075(0.0140)	**0.1761(0.0137)**	0.0539(0.0116)
DMJC	0.1354(0.0025)	0.1420(0.0032)	0.0504(0.0027)
DCCA	0.0567(0.0013)	0.1355(0.0013)	0.0013(0.0002)
DCCAE	0.0580(0.0005)	0.1245(0.0009)	0.0011(0.0001)
DCN	0.1163(0.0013)	0.1503(0.0056)	0.0471(0.0023)
MV*k*-means	<u>0.1588(0.0019)</u>	0.1572(0.0026)	<u>0.0593(0.0021)</u>
AE + MV*k*-means	0.1415(0.0031)	0.1430(0.0037)	0.0510(0.0024)
CDMKM	**0.1670(0.0023)**	<u>0.1619(0.0037)</u>	**0.0615(0.0019)**

Table 10
Performance comparisons on MNIST&USPS.

Method	NMI	ACC	ARI
GMNMF	0.7109(0.0139)	0.6251(0.0397)	0.5622(0.0201)
LMSC	0.8041(0.0300)	0.8018(0.0692)	0.7314(0.0645)
CoregSC	0.6413(0.0228)	0.7289(0.0528)	0.5724(0.0417)
AWGL	0.6838(0.0207)	0.6871(0.0348)	0.5712(0.0334)
DEMVC	0.8732(0.0217)	0.9006(0.0645)	0.8550(0.0538)
DMJC	<u>0.9341(0.0252)</u>	**0.9266(0.0701)**	**0.9109(0.0629)**
DCCA	0.8351(0.0085)	0.7122(0.0188)	0.5981(0.0431)
DCCAE	0.8183(0.0230)	0.6604(0.0332)	0.5914(0.0823)
DCN	0.6907(0.0210)	0.6975(0.0369)	0.5725(0.0336)
MV*k*-means	0.8180(0.0393)	0.7939(0.0658)	0.7424(0.0707)
AE + MV*k*-means	0.8536(0.0286)	0.8709(0.0731)	0.8203(0.0615)
CDMKM	**0.9362(0.0187)**	<u>0.9244(0.0661)</u>	<u>0.9079(0.0550)</u>

Table 11
Nemenyi test (NT) and Wilcoxon signed-rank test (WT) for methods comparisons on all datasets under each metric. The values in the table represent the probability value, and the value less than the significance level $\alpha = 0.1$ is indicated in bold, which demonstrates a significant improvement of CDMKM over this compared method.

Method	NMI		ACC		ARI	
	NT	WT	NT	WT	NT	WT
GMNMF	0.1203	**0.0277**	0.1470	**0.0277**	0.3524	**0.0464**
LMSC	**0.0218**	**0.0277**	0.1470	**0.0277**	0.1786	**0.0277**
CoregSC	**0.0776**	**0.0277**	0.2561	**0.0277**	0.1470	**0.0277**
AWGL	**0.0165**	**0.0277**	0.3524	**0.0277**	0.1626	**0.0277**
DEMVC	0.5665	**0.0277**	0.7678	**0.0464**	0.9000	**0.0747**
DMJC	0.9000	**0.0277**	0.9000	**0.0464**	0.9000	**0.0464**
DCCA	**0.0165**	**0.0277**	**0.0026**	**0.0277**	**0.0050**	**0.0277**
DCCAE	**0.0218**	**0.0277**	**0.0010**	**0.0277**	**0.0026**	**0.0277**
DCN	**0.0478**	**0.0277**	**0.0776**	**0.0277**	**0.0124**	**0.0277**
MV*k*-means	0.1203	**0.0277**	0.4081	**0.0277**	0.2786	**0.0277**
AE + MV*k*-means	0.9000	**0.0277**	0.9000	**0.0277**	0.9000	**0.0277**

4.3. Running Time Comparisons

In this section, we compare the running time of CDMKM with other methods except for the single-view method DCN and the variant methods MV*k*-means and AE + MV*k*-means on Caltech101-20, Scene, MNIST&USPS and NUS-WIDE-OBJ. The corresponding results are shown in Table 12. It can be seen that the shallow methods GMNMF, LMSC, CoregSC and AWGL demonstrate obviously lower running time than the deep methods on the smaller-size datasets Caltech101-20 and Scene, but for larger-size datasets MNIST&USPS and NUS-WIDE-OBJ, CoregSC shows no obvious efficiency over the deep methods, and LMSC and AWGL show extremely high running time compared with the others. Among the deep methods, CDMKM shows average-level running time on each dataset. Comparing the running time of CDMKM on different datasets, we can

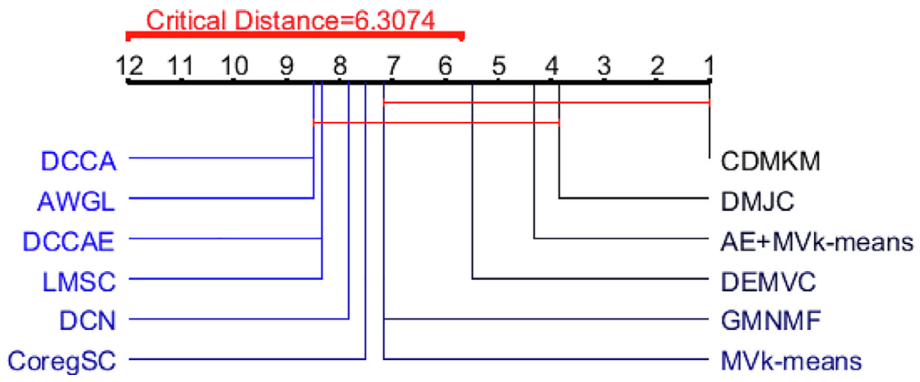


Fig. 2. CD diagram of Nemenyi test among all methods under NMI metric.

Table 12
Running time of different methods on four datasets. (unit: seconds)

Method	Caltech101-20	Scene	MNIST&USPS	NUS-WIDE-OBJ
GMNMF	156	24	60	462
LMSC	164	222	9644	28131
CoregSC	37	23	355	1709
AWGL	61	74	3657	12304
DEMVC	777	361	420	632
DMJC	481	232	272	473
DCCA	146	161	163	1051
DCCAE	472	508	673	3106
CDMKM	678	294	387	1749

observe that when the size of the dataset (in terms of the number of samples, the number of clusters or the number of views) scales up, the running time of CDMKM also scales up, but the increase is not too great.

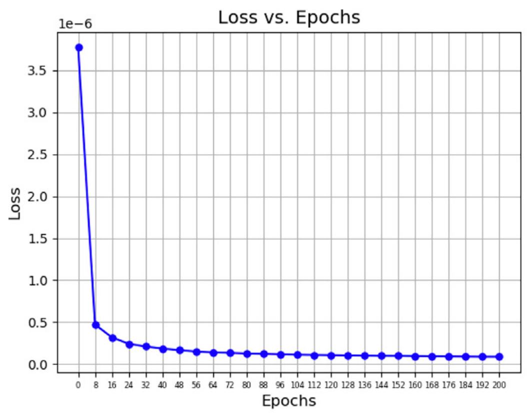
4.4. Performance over Epochs

To observe how the loss and the clustering performance of the proposed method change during the optimization process, we plot the changing curves of the loss for Eq. (1) and the metric NMI over the training epochs for each dataset, as presented in Figs. 3 and 4 respectively. From Fig. 3, we can see that the loss on all datasets continuously decreases over the training epochs, with decreasing very rapidly in the first few epochs and slowly down in the latter epochs. From Fig. 4, the NMI curve for each dataset shows an overall ascending trend with some fluctuations until rising to a certain stable level and then fluctuates around this level. These results show that our model on all datasets works in the desired direction.

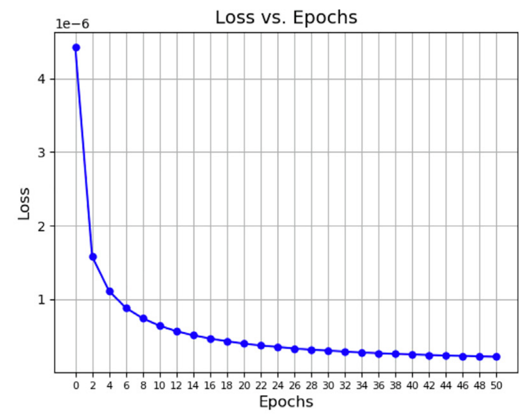
4.5. Visualization of Results

To show the superiority of the embedded features obtained by our method over the original features, we choose three datasets, Mfeat, Scene and MNIST&USPS, to visualize their original features and the learned embedded features in 2-dimensional space via *t*-SNE [45]. The visualizations for view 1 of Scene, view 1 of Mfeat, and both views of MNIST&USPS in both the original space and the deep embedding space are shown in Fig. 5. The shape of each point is plotted as its ground-truth label, and points of the same label are plotted with the same color. We can observe that for all datasets, after transformation by our method, the points of different labels are more cluster-discriminative than in the original space. Especially for Mfeat and MNIST&USPS, the effect of our method is more significant. After transformation by our method, points of the same class are more concentrated, and points of different classes are more linearly separable, demonstrating a more *k*-means-friendly data distribution than in the original space. These comparisons between features in the original space and embedding space of CDMKM verify the good representation capability of our method and thus satisfactory clustering performance.

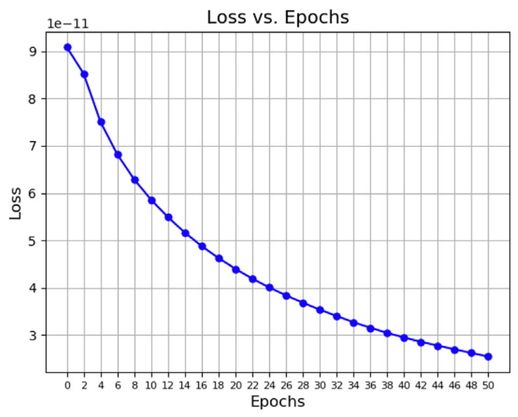
Although the proposed method demonstrates satisfactory numerical results for all datasets, the *t*-SNE visualization results show that for datasets with originally more cluster-separable data distributions, our method performs more effectively than for datasets that are originally poorly-separated. The reason for this is because the representation learning is guided by the continually refined cluster centroids, which highly depend on the initial centroids learned by multi-view *k*-means. If the dataset is relatively well-separated, the initial learned cluster centroids are prone to be more cluster-



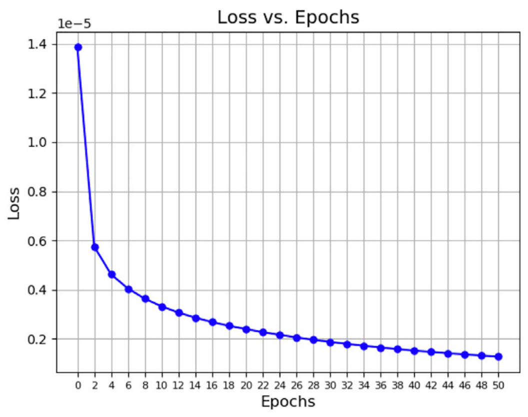
(a) Mfeat



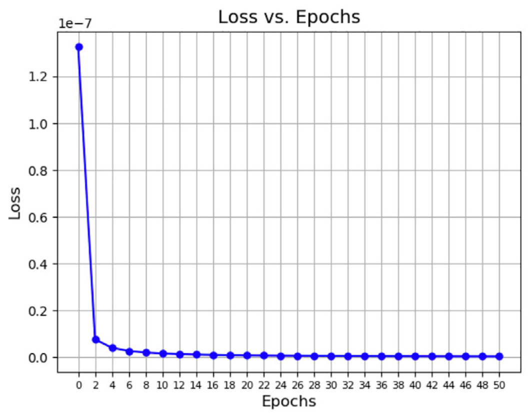
(b) Caltech101-20



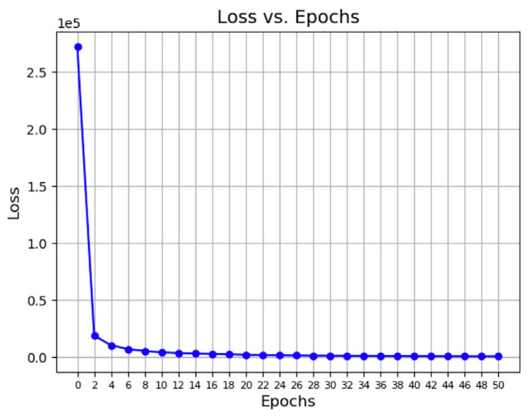
(c) Scene



(d) Corel



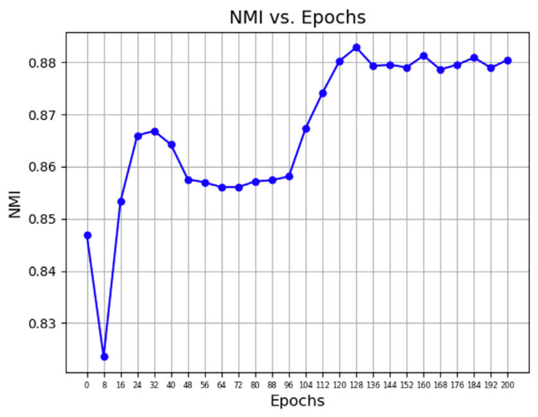
(e) NUS-WIDE-OBJ



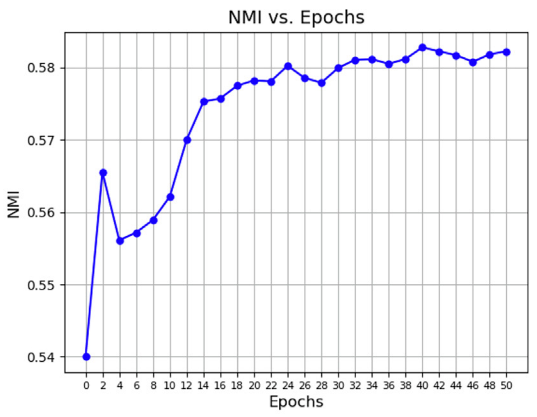
(f) MNIST&USPS

Fig. 3. The loss variation with training epochs on six datasets.

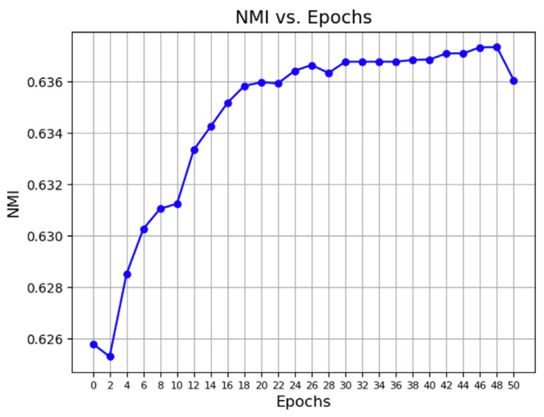
discriminative and thus will effectively guide the distribution of the learned representation improvingly toward the true class distribution. While if the dataset is originally very scattered and poorly-separated, the initial learned cluster centroids may be far from the true class distribution and thus may wrongly guide parts of data points to wrong classes. Therefore, our method is more applicable for datasets with relatively well-separated data distributions, and the effectiveness of our method



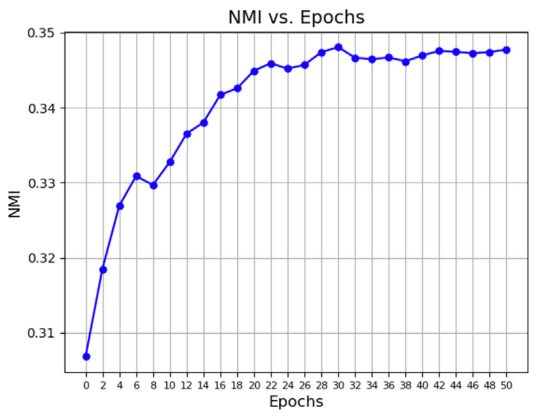
(a) Mfeat



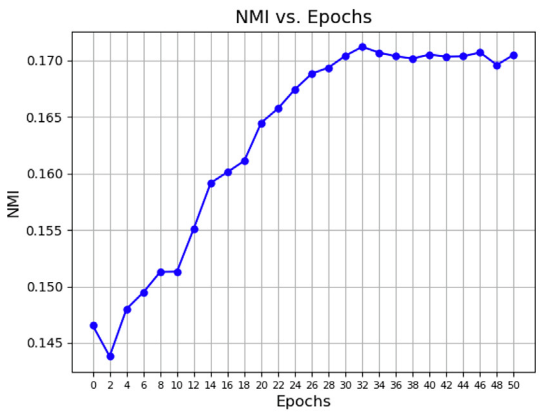
(b) Caltech101-20



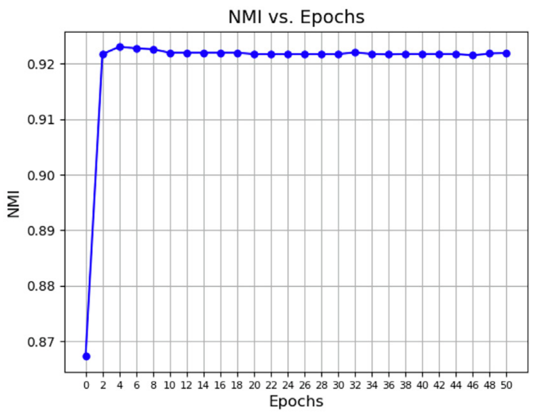
(c) Scene



(d) Corel



(e) NUS-WIDE-OBJ



(f) MNIST&USPS

Fig. 4. The NMI variation with training epochs on six datasets.

on datasets with poorly-separated data distributions may not be very significant. In real applications, to make our method more applicable for datasets that are not well-separated, we can explore or make use of more advanced centroids initialization algorithms to make the initialized cluster centroids more cluster-discriminative, or we can introduce the semi-

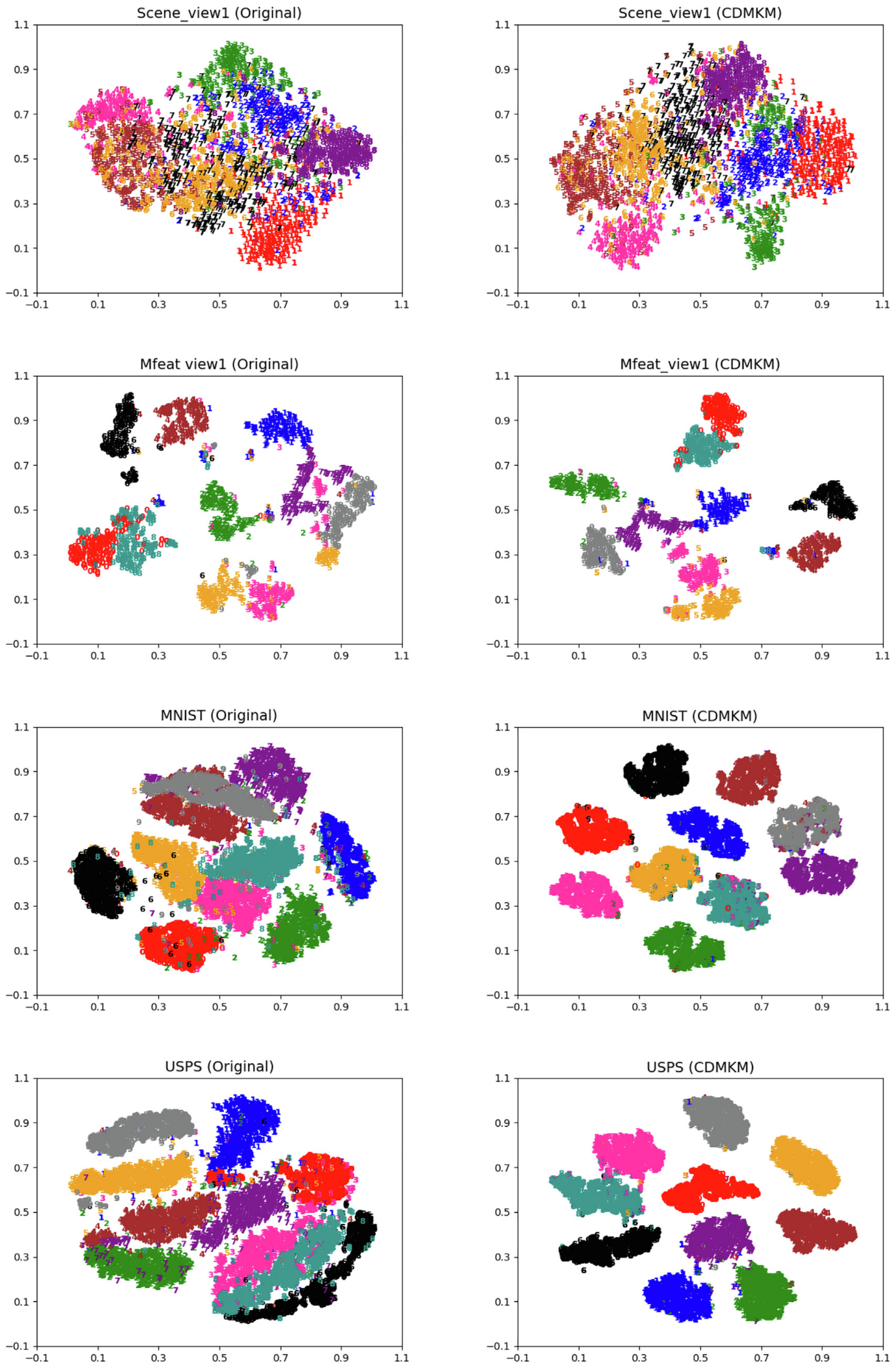


Fig. 5. *t*-SNE visualization of features in original space and features in deep embedding space of CDMKM. (The shape of each point is plotted as its ground-truth label.).

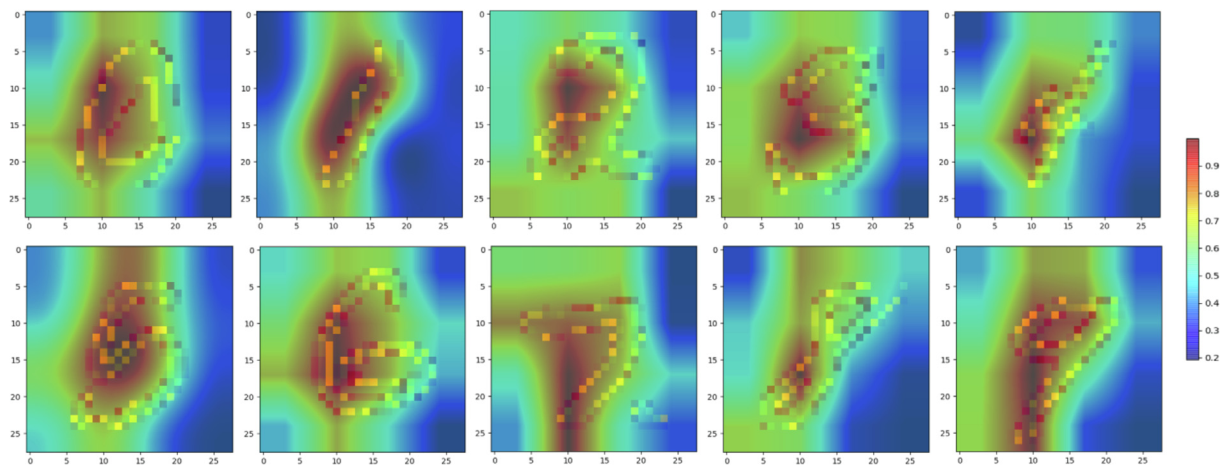


Fig. 6. Visualizing the feature maps of the last convolution layer of the CDMKM encoder part on MNIST by mapping the feature maps to the heat map over the input image.

supervised idea to use parts of labeled data to assist the cluster centeroids determination, making the subsequent learning more accurate.

For MNIST&USPS that uses the CNN architecture, we further visualize the learned feature maps of the last convolution layer of the encoder part. Since the raw feature maps cannot show any meaningful information, we use the Grad-CAM [46] method to map the feature maps to the heat map over the input image. The heat map can show the degree to which the last convolutional layer responds to each region of the input image. Fig. 6 shows the heat maps for 10 randomly selected samples from 0–9. We can observe that all these heat maps highlight the area around the digit, which is exactly the class-discriminative region. This indicates that the proposed method makes the convolutional neural network more focused on the class-discriminative region of the input image.

For the clustering results on MNIST&USPS obtained by our method, we randomly select 20 samples of MNIST from each cluster and visualize their original images and reconstructed images in Fig. 7. Since our model incorporates the multi-view k-means objective into the training of the autoencoders, we can observe that for most of the samples, the reconstructed images not only restore the original digit shape but also adjust the digit to a more standard shape. Fig. 8 shows some typical instances (selected from Fig. 7) whose reconstructed images obviously adjust the shape of the original digit. These results demonstrate the strong reconstruction ability of our model, which further implies the good representation capability of the representation layer, thereby leading to better clustering performance. As a result, we can observe in Fig. 7 that most samples can be clustered correctly, except for only a few misclassified digits that are inherently easy to be confused in the original shape or over-adjusted by reconstruction in the training process.

4.6. Impact of Parameters

In our model, there are two hyperparameters p and λ that should be set properly. In our experiments, we tune p and λ from $\{2, 4, 8, 16, 32\}$ and $\{0.01, 0.1, 1, 10, 100\}$ respectively. The results of the NMI metric under different parameters on different datasets are shown in Fig. 9. For the parameter p, the larger p value generally produces better performance, and the performance becomes stable when $p > 8$. For the parameter λ, our model obtains the best clustering performance when $\lambda = 1$ on the first five datasets that use the DNN architecture, and on the last dataset that uses the CNN architecture, the best clustering performance occurs when $\lambda = 0.01$. Therefore, in our experiments, we set $p = 16$ for all datasets, $\lambda = 1$ for the DNN-based networks on the first five datasets and $\lambda = 0.01$ for the CNN-based network on the MNIST&USPS dataset.

5. Conclusion

In this paper, we have proposed a centroids-guided deep multi-view k-means clustering method that incorporates deep representation learning into the multi-view k-means framework. The cluster centroids of each view obtained in the multi-view k-means objective guide the deep representation learning of each view to produce more k-means-friendly representations toward a common partition, thereby improving the clustering accuracy. Experimental results on six datasets demonstrate that the proposed method is superior to the state-of-the-art multi-view clustering methods. And the proposed method by jointly conducting deep representation learning and multi-view k-means substantially improves the clustering performance over conducting multi-view k-means in the original space as well as conducting deep representation extraction and multi-view k-means in a separate manner. Furthermore, the visualizations of the data distribution for some datasets demonstrate that the deep embedding space obtained by our method is more cluster-discriminative and k-means-

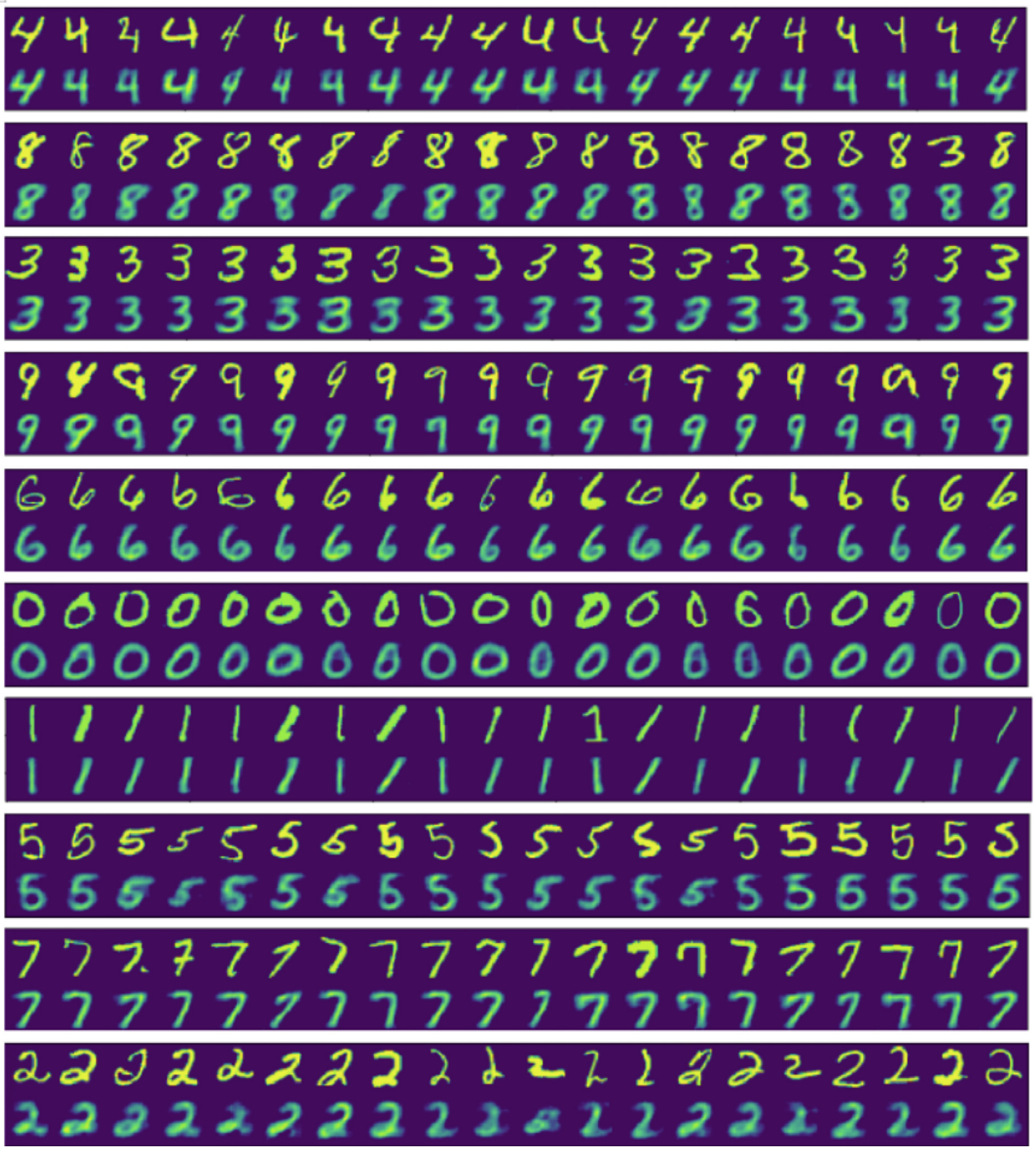


Fig. 7. Parts of clustering results of the proposed method on MNIST. Each two rows represent 20 samples randomly selected from one cluster with the first row representing the input images and the second row representing their corresponding reconstructed images.

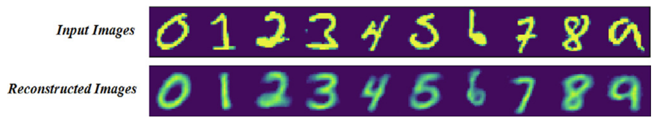


Fig. 8. Some typical instances that the reconstructed images obviously adjust the shape of the original digit.

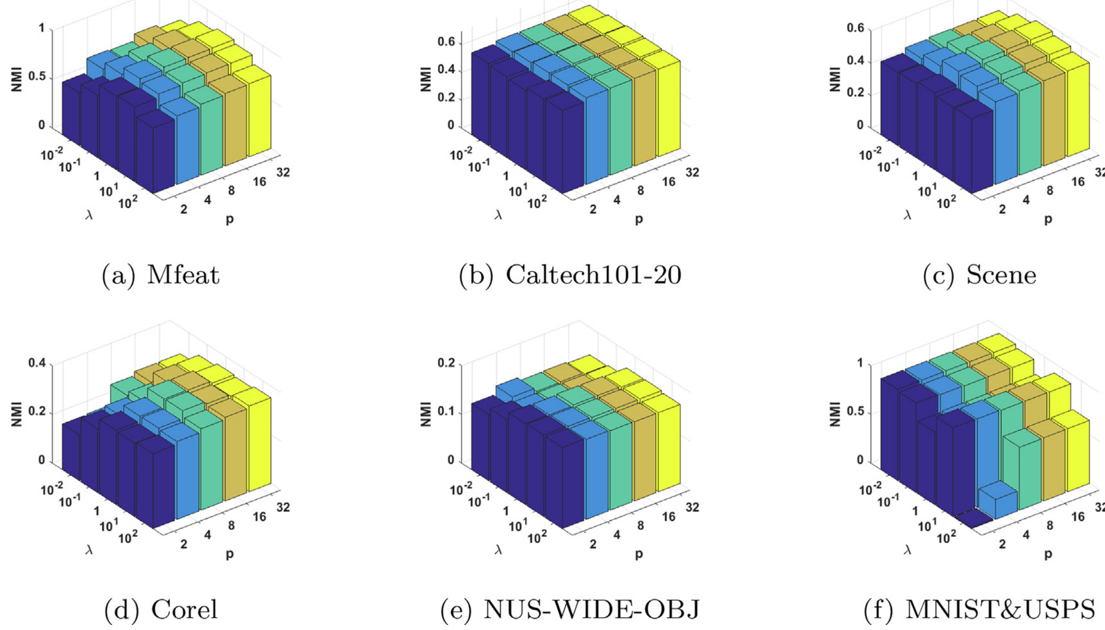


Fig. 9. The parameters effects of p and λ on different datasets with NMI metric.

friendly than the original data space. In future studies, this framework can be extended to the semi-supervised context by using parts of labeled data to assist the cluster centroids optimization in the learning process, ensuring that the cluster centroids are more in line with the true class distribution, thereby making the centroids-guided learning more effective and accurate.

CRediT authorship contribution statement

Jing Liu: Investigation, Conceptualization, Methodology, Validation, Visualization, Writing - original draft, Writing - review & editing. **Fuyuan Cao:** Resources, Project administration, Funding acquisition, Supervision, Writing - review & editing. **Jiye Liang:** Resources, Supervision, Project administration.

Declaration of Competing Interest

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

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61976128, 62072293, 62022052), the Open Project Foundation of Intelligent Information Processing Key Laboratory of Shanxi Province (Grant No. CICIP2021005), the Excellent Graduate Innovation Project of Shanxi Province (Grant No. 2020BY007), the Shanxi Province Science Foundation for Youths (Grant No. 201901D211168), the Key Research and Development Project (Grant No. 2020AAA0106100), and the 1331 Engineering Project of Shanxi Province, China.

References

[1] L. Fu, P. Lin, A.V. Vasilakos, S. Wang, An overview of recent multi-view clustering, Neurocomputing 402 (2020) 148–161.
[2] J. Liu, C. Wang, J. Gao, J. Han, Multi-view clustering via joint nonnegative matrix factorization, in: Proceedings of the 2013 SIAM International Conference on Data Mining, 2013, pp. 252–260.
[3] N. Rai, S. Negi, S. Chaudhury, O. Deshmukh, Partial multi-view clustering using graph regularized nmf, in: 2016 23rd International Conference on Pattern Recognition (ICPR), IEEE, 2016, pp. 2192–2197.
[4] M. Chen, L. Huang, C. Wang, D. Huang, Multi-view clustering in latent embedding space, Proceedings of the AAAI conference on artificial intelligence 34 (2020) 3513–3520.
[5] N. Liang, Z. Yang, Z. Li, W. Sun, S. Xie, Multi-view clustering by non-negative matrix factorization with co-orthogonal constraints, Knowledge-Based Systems 194 (2020) 105582.
[6] Q. Yin, S. Wu, R. He, L. Wang, Multi-view clustering via pairwise sparse subspace representation, Neurocomputing 156 (2015) 12–21.

[7] H. Gao, F. Nie, X. Li, H. Huang, Multi-view subspace clustering, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4238–4246.
[8] K. Chaudhuri, S.M. Kakade, K. Livescu, K. Sridharan, Multi-view clustering via canonical correlation analysis, in: Proceedings of the 26th annual International Conference on Machine Learning, 2009, pp. 129–136.
[9] N. Rasiwasia, D. Mahajan, V. Mahadevan, and G. Aggarwal. Cluster canonical correlation analysis. In Artificial Intelligence and Statistics, pages 823–831, 2014.
[10] G. Andrew, R. Arora, J.A. Bilmes, and K. Livescu. Deep canonical correlation analysis. In Proceedings of the 30th International Conference on Machine Learning, volume 28, pages 1247–1255, 2013.
[11] W. Wang, R. Arora, K. Livescu, J. Bilmes, On deep multi-view representation learning, in: International Conference on Machine Learning, 2015, pp. 1083–1092.
[12] W. Wang, B.C. Ooi, X. Yang, D. Zhang, Y. Zhuang, Effective multi-modal retrieval based on stacked auto-encoders, Proceedings of the VLDB Endowment 7 (8) (2014) 649–660.
[13] S. Rastegar, M. Soleymani, H.R. Rabiee, S.M. Shojaee, Mdl-cw: A multimodal deep learning framework with crossweights, Computer Vision and Pattern Recognition (2016).
[14] Y. Xie, B. Lin, Y. Qu, C. Li, W. Zhang, L. Ma, Y. Wen, D. Tao, Joint deep multi-view learning for image clustering, IEEE Transactions on Knowledge and Data Engineering (2020), pages 1–1.
[15] Xu. Jie, Yazhou Ren, Guofeng Li, Lili Pan, Ce Zhu, Xu. Zenglin, Deep embedded multi-view clustering with collaborative training, Information Sciences 573 (2021) 279–290.
[16] G. Du, L. Zhou, Y. Yang, K. Lü, L. Wang, Deep multiple auto-encoder-based multi-view clustering, Data Science and Engineering 6 (3) (2021) 323–338.
[17] X. Sun, M. Cheng, C. Min, L. Jing, Self-supervised deep multi-view subspace clustering, in: Proceedings of The 11th Asian Conference on Machine Learning, volume 101 of Proceedings of Machine Learning Research, 2019, pp. 1001–1016.
[18] B. Yang, X. Fu, N.D. Sidiropoulos, and M. Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In Proceedings of the 34th International Conference on Machine Learning, volume 70, pages 3861–3870, 2017.
[19] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In Proceedings of the 33nd International Conference on Machine Learning, volume 48, pages 478–487, 2016.
[20] X. Guo, X. Liu, E. Zhu, J. Yin, Deep clustering with convolutional autoencoders, in: International Conference on Neural Information Processing, 2017, pp. 373–382.
[21] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid. Deep subspace clustering networks. Advances in Neural Information Processing Systems, 30, 2017.
[22] J. Yang, D. Parikh, D. Batra, Joint unsupervised learning of deep representations and image clusters, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5147–5156.
[23] Z. Jiang, Y. Zheng, H. Tan, B. Tang, H. Zhou, Variational deep embedding: An unsupervised and generative approach to clustering, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, 2017, pp. 1965–1972.
[24] D.P. Kingma, M. Welling, Auto-encoding variational bayes, in: 2nd International Conference on Learning Representations, 2014.
[25] A. Kumar, H. Daumé, A co-training approach for multi-view spectral clustering, in: Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 393–400.
[26] A. Kumar, P. Rai, and H. Daumé. Co-regularized multi-view spectral clustering. In Advances in Neural Information Processing Systems, pages 1413–1421, 2011.
[27] Y. Li, F. Nie, H. Huang, J. Huang, Large-scale multi-view spectral clustering via bipartite graph, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 2750–2756.
[28] F. Nie, J. Li, X. Li, Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 1881–1887.
[29] Y. Wang, W. Zhang, L. Wu, X. Lin, M. Fang, and S. Pan. Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering. arXiv preprint arXiv:1608.05560, 2016.
[30] G. Tzortzis, A. Likas, Kernel-based weighted multi-view clustering, in: 2012 IEEE 12th International Conference on Data Mining, 2012, pp. 675–684.
[31] X. Cai, F. Nie, H. Huang, Multi-view k-means clustering on big data, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013, pp. 2598–2604.
[32] Y.M. Xu, C.D. Wang, J.H. Lai, Weighted multi-view clustering with feature selection, Pattern Recognition 53 (2016) 25–35.
[33] J. Liu, F. Cao, X.Z. Gao, L. Yu, and J. Liang. A cluster-weighted kernel k-means method for multi-view clustering. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, pages 4860–4867, 2020.
[34] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A.Y. Ng, Multimodal deep learning, in: Proceedings of the 28th International Conference on Machine Learning, 2011.
[35] C. Zhang, Y. Liu, H. Fu, Ae2-nets: Autoencoder in autoencoder networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2577–2585.
[36] X. Yan, S. Hu, Y. Mao, Y. Ye, H. Yu, Deep multi-view learning methods: A review, Neurocomputing 448 (2021) 106–129.
[37] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, 2015.
[38] A. Monadjemi, B. Thomas, and M. Mirmehdi. Experiments on high resolution images towards outdoor scene classification, technical report, tech. rep. University of Bristol, Department of Computer Science, 2002.
[39] Y. LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 2010.
[40] J.J. Hull, A database for handwritten text recognition research, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (5) (1994) 550–554.
[41] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 807–814.
[42] N. Rai, S. Negi, S. Chaudhury, O. Deshmukh, Partial multi-view clustering using graph regularized NMF, in: 23rd International Conference on Pattern Recognition ICPR, 2016, pp. 2192–2197.
[43] C. Zhang, H. Fu, Q. Hu, X. Cao, Y. Xie, D. Tao, D. Xu, Generalized latent multi-view subspace clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 42 (1) (2020) 86–99.
[44] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the Society for Industrial and Applied Mathematics 5 (1) (1957) 32–38.
[45] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, Journal of Machine Learning Research 9 (11) (2008).
[46] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626.
[47] J.C. French, J.V. Watson, X. Jin, W.N. Martin, Integrating multiple multi-channel CBIR systems, in: In: Proc. Inter. Workshop on Multimedia Information Systems (MIS), Citeseer, 2003.