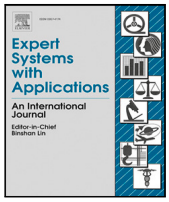
Expert Systems With Applications 236 (2024) 121298

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Deep multi-view graph clustering network with weighting mechanism and collaborative training

Jing Liu [a,b], Fuyuan Cao [a,*], Xuechun Jing [a], Jiye Liang [a]

[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

[b] School of Software, Shanxi Agricultural University, Taigu 030801, China

ARTICLE INFO

Keywords:
Multi-view clustering
Graph clustering
Graph convolutional network
Graph autoencoder

ABSTRACT

With the development of graph convolutional network (GCN), which is powerful in graph embedding learning meanwhile can capture node feature information, deep multi-view graph clustering methods based on graph autoencoder have emerged as a new stream. Although they achieve satisfactory performance, they still have the following weaknesses: (1) some of them model the weights for different views in the encoder part by using attention in the multi-view embedding fusion layer, but fail to consider the weighting in the decoder part to measure the contributions of different views to the reconstruction loss. (2) Most of them directly conduct clustering on the multi-view common embedding layer, but fail to guarantee the alignment of clustering results of different views. To this end, we propose a novel GCN-based deep multi-view graph clustering network with weighting mechanism and collaborative training (DMVGC). The model is composed of multiple view-specific graph encoders and a unified graph decoder. Besides the specially-designed attention module in the encoder part, we construct a reconstruction loss with adaptive weighting mechanism in the decoder part. Additionally, a collaborative self-training clustering objective is jointly conducted on each view-specific embedding layer and the common embedding layer, to make the embedding of each view clustering-friendly toward a common partition. Experiments on several datasets demonstrate the effectiveness of our model.

1. Introduction

Multi-view data are a set of instances that are represented by heterogeneous features from multiple views. For example, images can be represented by different visual descriptors; documents may be translated into multiple languages. The features of different views contain both consistent and complementary information. Multi-view clustering is aimed to employ features from multiple views to learn a common clustering partition, which has been extensively studied (Fang, Li, Li, Gao, Jia & Zhang, 2023; Fu, Lin, Vasilakos, & Wang, 2020) and demonstrates better performance than single-view clustering.

In the past decade, various multi-view clustering methods have been proposed, among which graph-based multi-view methods make up a large proportion for handling graph-structured data. The multi-view graph data reflect multiple graph relationships for the same set of instances, which can be constructed from original multi-view features, or inherently exist in many real applications. For example, a movie network has graphs of two views, including co-author relationship and co-director relationship. Most of existing graph-based methods either focus on learning a consensus graph followed by spectral clustering (Fan, Huang, Cai, Wang, He & Tang, 2023; Huang, Wang, & Lai, 2023; Lin, Kang, Zhang, & Tian, 2023; Nie, Li, & Li, 2017; Pan & Kang, 2021; Tang et al., 2020; Wang, Yang, Liu & Fujita, 2019) or try to learn a consistent graph representation by jointly performing spectral clustering with multiple graphs (Kumar, Rai, & Daumé, 2011; Nie, Li, & Li, 2016; Zhao, Kang, Zou, & Wang, 2023). However, these methods are based on shallow models, which are limited to discover deep relations in graphs and fail to capture the node feature information.

Graph convolutional network (GCN) (Kipf & Welling, 2017) that emerged as a new class of deep neural networks can effectively process graph data, which is powerful in graph embedding learning by exploiting both graph structure and node feature information. For unsupervised graph learning, graph autoencoder (GAE) (Kipf & Welling, 2016) is a commonly-used model to learn deep representations for graph nodes by reconstructing the graph structure. In recent years, with GAE increasingly employed as the deep approach to achieve single-view graph clustering in many works (Li, Zhang, & Zhang, 2022; Wang, Pan, Hu, Long, Jiang & Zhang et al., 2019; Zhang, Li, Zhang, & Li, 2022), there have been some works extending the GAE-based approach into the field of multi-view graph clustering. The first attempt is the

* Corresponding author.
E-mail addresses: jingliu_sxu@hotmail.com (J. Liu), cfy@sxu.edu.cn (F. Cao), jingxuechun123@163.com (X. Jing), ljy@sxu.edu.cn (J. Liang).

https://doi.org/10.1016/j.eswa.2023.121298
Received 10 April 2023; Received in revised form 21 August 2023; Accepted 22 August 2023
Available online 26 August 2023

work (Fan et al., 2020) that proposed a one-to-multi graph autoencoder to employ one informative view to reconstruct multiple graph views, with jointly optimized with a self-training clustering objective to get increasingly refined clusters. Afterwards, one of the representative work (Wang, Chang, Fu, & Zhao, 2023) proposed a multi-to-multi graph autoencoder for clustering with employing a multi-view mutual information maximization module and a graph fusion network. Although these models achieve satisfactory performance, they still have some weaknesses. First, although some of them consider weighting for different views in the encoder part by using attention in the multi-view embedding fusion layer, they all equally treat each view in the decoder part without differentiating the contributions of different views to the reconstruction loss, which may result in suboptimal performance. Second, they focus on learning a common graph embedding among views and directly conduct clustering on the common embedding layer, without guaranteeing the alignment of clustering results of different views.

To address the above issues, we propose a novel GCN-based multi-view graph clustering network, named deep multi-view graph clustering network with weighting mechanism and collaborative training (DMVGC). The model is composed of multiple view-specific graph encoders and a unified graph decoder. A specially-designed view-level attention module is included in the encoder part, and a reconstruction loss with adaptive weighting mechanism is constructed in the decoder part. And a collaborative self-training clustering objective is jointly conducted on each view-specific embedding layer and the common embedding layer. Specifically, to keep both diversity and consistency of different views, the model encodes multi-view graphs separately and decode them jointly through a unified decoder. The forward-propagation of multiple encoders captures the diversity of views, and the back-propagation driven by a single unified decoder guarantees the consistency across views. To measure the importance of different views, in the encoder part the attention-based weighting mechanism is used to model the view-level weights in the multi-view fusion layer, and in the decoder part the adaptive weighting mechanism of the reconstruction loss is used to measure the contributions of different views to the back-propagated loss. To make the embedding of each view clustering-friendly toward a common partition, a common cluster distribution generated from the common embedding is used as the target to collaboratively guide the training of each view-specific distribution. The network is trained jointly with the reconstruction loss and clustering loss to simultaneously optimize the graph embedding and clustering assignments in a unified framework, which mutually benefits both of them.

The main contributions are summarized as follows:

- A novel GCN-based deep multi-view graph clustering network is proposed, which employs the weighting mechanism in both encoder and decoder parts, and constructs a collaborative self-training clustering objective simultaneously optimized with the graph embedding learning.
- The optimizing strategy for the proposed model is well designed, which jointly minimizes the reconstruction loss and clustering loss with a trade-off factor, and alternatively updates the network parameters and the view weights.
- Extensive experiments demonstrate that the proposed model outperforms the state-of-the-art methods and the variant baselines, and validate the effect of the weighting mechanism and collaborative training in our model.

The rest of the paper is organized as follows. Related works are reviewed in Section 2. Details of the proposed model are presented in Section 3. Experimental results are shown in Section 4. Finally, conclusions are given in Section 5.

2. Related works

In this section, we give a brief review of multi-view clustering, and make a survey about the existing works on multi-view graph clustering based on shallow approaches as well as GCN-based deep approaches.

2.1. Multi-view clustering

Based on the type of input data, multi-view clustering methods can be divided into two categories: feature-based methods and graph-based methods. For feature-based methods, the input data are features where each data sample is represented by a set of features. In this category, some methods are based on shallow approaches, such as NMF (non-negative matrix factorization) based methods (Chen, Huang, Wang, & Huang, 2020; Huang et al., 2021; Liang, Yang, Li, Sun, & Xie, 2020; Liu, Wang, Gao, & Han, 2013; Rai, Negi, Chaudhury, & Deshmukh, 2016), k-means based methods (Cai, Nie, & Huang, 2013; Liu, Cao, Gao, Yu, & Liang, 2020; Liu, Dou, Yin, Wang, & Zhu, 2016; Xu, Wang, & Lai, 2016), CCA (canonical correlation analysis) based methods (Chaudhuri, Kakade, Livescu, & Sridharan, 2009; Rasiwasia, Mahajan, Mahadevan, & Aggarwal, 2014) and subspace clustering based methods (Gao, Nie, Li, & Huang, 2015; Kang, Lin, Zhu, & Xu, 2022; Wang et al., 2016; Yin, Wu, He, & Wang, 2015; Zhang et al., 2020). Others are based on deep approaches, namely DNN (deep neural network) or CNN (convolutional neural network) based methods (Du, Zhou, Yang, Lü, & Wang, 2021; Liu, Cao, & Liang, 2022; Xie et al., 2021; Xu et al., 2021, 2023). For the graph-based methods, the input data are graph structures where data samples are connected by edges, and features in this case are optional. In this category, methods can also be divided into shallow methods and deep methods, which are reviewed in detail in the following two sections.

2.2. Shallow multi-view graph clustering methods

Most existing graph-based multi-view clustering methods are based on shallow approaches, which mainly employ the technique of spectral clustering. These methods can be divided into two categories: consensus graph based methods and consensus graph representation based methods. The first category is to learn a consensus graph among views and conduct spectral clustering on it. For example, Nie et al. (2017) proposed a self-weighted multi-view clustering method that weightedly combines each single graph learning model to learn a common graph. Wang, Yang et al. (2019) proposed a graph-based system for multi-view clustering that automatically weights each view-specific graph to generate the unified graph. Tang et al. (2020) proposed a model to learn a unified graph for multi-view clustering via cross-view graph diffusion. Fan, Huang et al. (2023) proposed an efficient multi-view clustering approach via unified bipartite graph learning that jointly learns the view-consensus bipartite graph and the discrete cluster structure. Huang et al. (2023) proposed a fast multi-view clustering that utilizes the idea of ensemble clustering to formulate a unified bipartite graph for final graph partitioning via spectral clustering. The second category is to jointly perform spectral clustering with multiple graphs to learn the consistent graph representation, usually referred to eigenvectors in spectral clustering. For example, Kumar et al. (2011) proposed a co-regularized multi-view spectral clustering method that jointly optimizes the spectral clustering objective of multiple views with minimizing the disagreement of different views to finally obtain the common eigenvectors. Nie et al. (2016) proposed a auto-weighted multi-graph clustering that jointly performs spectral clustering with multi-view graphs to learn the common eigenvectors. Despite acceptable performance of the shallow methods, they are limited to discover complex relations in graphs and fail to capture node feature information.

Table 1
List of mathematical notations.

Symbol	Description
N	number of samples
V	number of views
K	number of clusters
$\boldsymbol{A}^{(v)}$	adjacency matrix of graph structure in the vth view
$\boldsymbol{X}^{(v)}$	node features in the vth view
$\boldsymbol{D}^{(v)}$	degree matrix of $\boldsymbol{A}^{(v)}$
$\boldsymbol{H}^{(v)}$	view-specific graph embedding for all samples in the vth view
$\boldsymbol{h}_i^{(v)}$	view-specific graph embedding for the ith sample in the vth view
$\boldsymbol{H}$	common embedding fused by all views
$\boldsymbol{A}$	adjacency matrix of the common graph obtained from the decoder
ω_v	weight for the vth view
q_{ij}	soft assignment of the ith sample to the jth cluster
p_{ij}	auxiliary target distribution derived from q_{ij}
$\boldsymbol{Q}$	soft assignment matrix with q_{ij} as its element in the ith row and jth column
$\boldsymbol{P}$	auxiliary target distribution matrix with p_{ij} as its element in the ith row and jth column
$\boldsymbol{\mu}_j^{(v)}$	centroid of the jth cluster on the embedding $\boldsymbol{H}^{(v)}$
$\boldsymbol{\mu}_j$	centroid of the jth cluster on the embedding $\boldsymbol{H}$

2.3. Deep multi-view graph clustering methods

In recent years, with the development of GCN, deep multi-view graph clustering methods based on GCN networks have emerged, which are powerful in graph embedding learning meanwhile can capture node features. Fan et al. (2020) proposed an one-to-multi graph autoencoder clustering model (O2MAC) that uses the embedding from one informative view to reconstruct multiple graphs. Cheng, Wang, Tao, Xie, and Gao (2020) proposed a multi-view attribute GCN (MAGCN) that employs two-pathway encoders to map graph embedding features and learn view-consistency information. However, O2MAC can only deal with multi-view graphs with single-view features and MAGCN aims at single graph with multi-view features. Later works target for dealing with multi-view graphs with multi-view features. For example, Wang et al. (2023) proposed a consistent multi-graph embedding clustering framework that employs a multi-view mutual information maximization module and a graph fusion network to learn a view consistent representation. Lin, Chen, Zhu, Wang, and Zhang (2022) employed the mutual information maximization module and specific information reconstruction module to enhance the deep multi-view graph clustering. Xia, Wang, Yang, Gao, Han and Gao (2022) proposed a multi-view graph embedding clustering network that uses the cluster labels learned by subspace clustering module to self-supervise the learning of node representation and the view-consensus coefficient matrix. Xia, Wang, Gao, Yang, and Gao (2022, 2023) focused on two-view graph embedding clustering that uses the pseudo-label to guide inter- and intra-cluster contrastive learning. The study on GCN-based deep multi-view graph clustering is still at the initial stage, and there are still many aspects need to be explored and improved. In this paper, we focus on the improvements on weighting mechanism and collaborative training for deep multi-view graph clustering.

3. Proposed model

3.1. Notations

Consider a multi-view graph dataset consisting of N samples represented by multiple graphs from V different views, which are to be partitioned into K clusters. The multi-view graph dataset can be represented by $\{\boldsymbol{A}^{(v)}, \boldsymbol{X}^{(v)}\}_{v=1}^{V}$, where $\boldsymbol{A}^{(v)} \in \mathbb{R}^{N\times N}$ represents the adjacency matrix of graph structure in vth view, and $\boldsymbol{X}^{(v)} \in \mathbb{R}^{N\times d^{(v)}}$ represents the node features in the vth view. In real applications, the graph structure $\boldsymbol{A}^{(v)}$ can be inherent or constructed from node features, and the node features $\boldsymbol{X}^{(v)}$ of different views can be different or the same. The main mathematical notations used in the paper are listed in Table 1.

3.2. The framework of DMVGC

As shown in Fig. 1, the network of the proposed model contains two parts: the multi-view graph encoder and the unified graph decoder. For the encoder part, multiple graph convolution encoders are constructed, and each encoder corresponds to one view to encode both graph structure and node features. Then the outputs of different encoders are fused into a common embedding with a view-level attention module which can automatically learn weights for different views. For the decoder part, the common embedding is fed into a single inner product decoder to learn a common graph by jointly reconstructing the graphs of different views. The adaptive weighting mechanism is employed to automatically learn the weight for each view-specific reconstruction loss. To align the clustering results of different views, a collaborative self-training clustering objective is incorporated by using a common distribution obtained from the common embedding to guide the distributions of each view-specific embedding.

3.2.1. Multi-view graph encoder with view-level attention

To fully exploit the diverse information of multiple views, we assign each view a GCN encoder to map the graph structure and node features into the graph embedding features. Specifically, for the vth view, the GCN mapping $GCN(\boldsymbol{X}^{(v)}, \boldsymbol{A}^{(v)}) \rightarrow \boldsymbol{H}^{(v)}$ transforms the graph $\boldsymbol{A}^{(v)}$ and node features $\boldsymbol{X}^{(v)}$ into the m-dimensional graph embedding $\boldsymbol{H}^{(v)} \in \mathbb{R}^{N\times m}$. For the lth GCN layer, the graph embedding $\boldsymbol{H}_{(l)}^{(v)}$ is output through the following transformation

$$\boldsymbol{H}_{(l)}^{(v)} = \sigma((\boldsymbol{D}^{(v)})^{-\frac{1}{2}}\boldsymbol{A}^{(v)}(\boldsymbol{D}^{(v)})^{-\frac{1}{2}}\boldsymbol{H}_{(l-1)}^{(v)}\boldsymbol{W}_{(l)}^{(v)}), \tag{1}$$

where $\boldsymbol{D}^{(v)} \in \mathbb{R}^{N\times N}$ is the degree matrix of $\boldsymbol{A}^{(v)}$, $\boldsymbol{W}_{(l)}^{(v)}$ is the trainable weights of the lth layer for the vth view, and $\sigma(\cdot)$ is the activation function. A two-layer GCN is applied for each encoder. Setting $\hat{\boldsymbol{A}}^{(v)} = (\boldsymbol{D}^{(v)})^{-\frac{1}{2}}\boldsymbol{A}^{(v)}(\boldsymbol{D}^{(v)})^{-\frac{1}{2}}$ and $\boldsymbol{X}^{(v)} = \boldsymbol{H}_{(0)}^{(v)}$, the final transformation for the vth view is obtained as the following form

$$\boldsymbol{H}^{(v)} = \sigma_2(\hat{\boldsymbol{A}}^{(v)}\sigma_1(\hat{\boldsymbol{A}}^{(v)}\boldsymbol{X}^{(v)}\boldsymbol{W}_{(1)}^{(v)})\boldsymbol{W}_{(2)}^{(v)}), \tag{2}$$

where σ_1 is the ReLU activation function (Nair & Hinton, 2010) for the first layer, and σ_2 is the linear activation function for the second layer.

To measure the importance of each view, we include a view-level attention module to learn the weights for the embeddings of different views and weightedly fuse them into a common embedding. In order to consider the view as a whole, instead of the common way that assigns each sample of each view a attention weight, we average the weights of all view-specific samples as the importance of the whole view. Specifically, the view-specific importance $\beta^{(v)}$ is computed as follows

$$\beta^{(v)} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{u}^T\tanh(\boldsymbol{W}_a\boldsymbol{h}_i^{(v)} + \boldsymbol{b}), \tag{3}$$

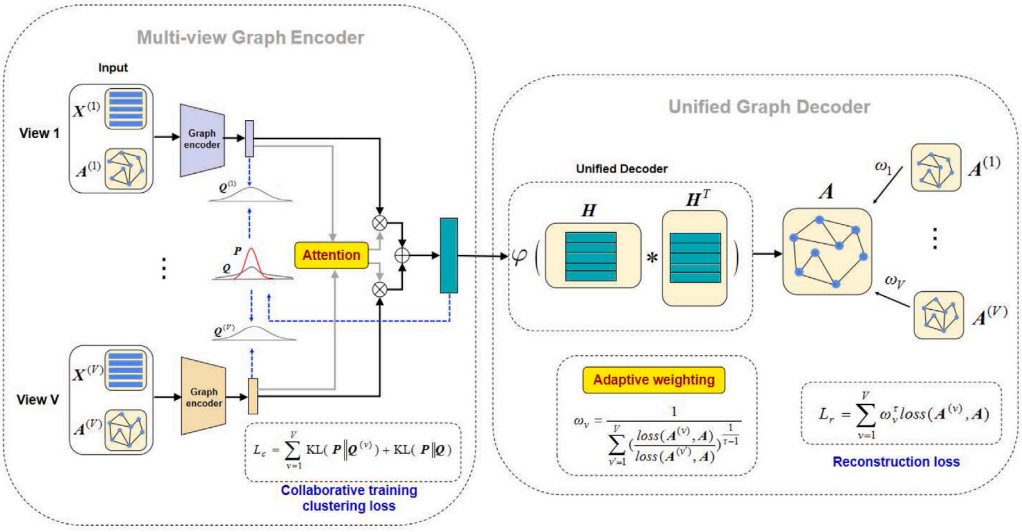


Fig. 1. The framework of the proposed DMVGC.

where $\boldsymbol{W}_a$ is the network weight matrix, $\boldsymbol{b}$ is the bias vector, and $\boldsymbol{u}$ is the attention vector, which are all learnable parameters in the attention module. Then we normalize the importance of each view by softmax function to obtain the view-specific attention $\alpha^{(v)}$ as follows

$$\alpha^{(v)} = \frac{exp(\beta^{(v)})}{\sum_{v=1}^{V} exp(\beta^{(v)})}. \tag{4}$$

Using the attention on each view, the common embedding $\boldsymbol{H}$ is obtained by fusing the view-specific embedding as follows

$$\boldsymbol{H} = \sum_{v=1}^{V} \alpha^{(v)} \boldsymbol{H}^{(v)}. \tag{5}$$

3.2.2. Unified graph decoder with adaptive weighting mechanism

The common embedding is fed into a single inner product decoder to generate a adjacency matrix $\boldsymbol{A}$ of the common graph, which is computed by

$$\boldsymbol{A} = \varphi(\boldsymbol{H}\boldsymbol{H}^{\top}), \tag{6}$$

where $\varphi(\cdot)$ is the sigmoid function. To keep the unity of different views, the common graph is jointly reconstructed by graphs of different views with weighting mechanism, which can be expressed by

$$L_r = \sum_{v=1}^{V} \omega_v^{\tau} loss(\boldsymbol{A}^{(v)}, \boldsymbol{A}), \tag{7}$$

where the loss function $loss(\cdot)$ is the binary cross-entropy between the view-specific target graph $\boldsymbol{A}^{(v)}$ and the output graph $\boldsymbol{A}$. The reconstruction loss for each view is assigned with a learnable weight ω_v to measure the importance of each view to the total loss. The weight is adaptively learned based on the loss of each view, which is described in Section 3.3.2. The exponent τ is a hyper-parameter used to control the distribution of the weights.

3.2.3. Clustering objective with collaborative training

To align the clustering results of different views, we incorporate a self-training clustering objective by using a target cluster distribution generated from the common embedding to jointly guide the cluster distribution of each view-specific embedding. This makes the embedding of each view more clustering-friendly toward a common partition. Specifically, following van der Maaten and Hinton (2008), we first obtain a centroid-based soft assignment q_{ij} of the common embedding $\boldsymbol{h}_i$ with respect to the cluster centroid $\boldsymbol{\mu}_j$ as follows

$$q_{ij} = \frac{(1 + \|\boldsymbol{h}_i - \boldsymbol{\mu}_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|\boldsymbol{h}_i - \boldsymbol{\mu}_{j'}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \tag{8}$$

where α is the degree of freedom of the Student's t-distribution. Following Xie, Girshick, and Farhadi (2016) we set $\alpha = 1$ in the experiments. Then an auxiliary target distribution p_{ij} derived from the current soft assignment q_{ij} is defined as

$$p_{ij} = \frac{q_{ij}^2/f_j}{\sum_{j'} q_{ij'}^2/f_{j'}}, \tag{9}$$

where $f_j = \sum_i q_{ij}$ are the soft frequencies per cluster. By raising q_{ij} to its second power and normalizing it with the soft frequencies per cluster, the target distribution p_{ij} actually strengthens the role of highly confident assignments of q_{ij}. Then we obtain the soft assignment $q_{ij}^{(v)}$ from each view-specific embedding as follows

$$q_{ij}^{(v)} = \frac{(1 + \|\boldsymbol{h}_i^{(v)} - \boldsymbol{\mu}_j^{(v)}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + \|\boldsymbol{h}_i^{(v)} - \boldsymbol{\mu}_{j'}^{(v)}\|^2/\alpha)^{-\frac{\alpha+1}{2}}}. \tag{10}$$

Then we use the common target p_{ij} to jointly guide the optimization of each view-specific distribution $q_{ij}^{(v)}$, by jointly minimizing the KL divergence loss between the common distribution p_{ij} and each view-specific distribution $q_{ij}^{(v)}$. The objective function is defined as follows

$$\begin{aligned} L_c &= \sum_{v=1}^{V} \mathrm{KL}(\boldsymbol{P}\|\boldsymbol{Q}^{(v)}) + \mathrm{KL}(\boldsymbol{P}\|\boldsymbol{Q}) \\ &= \sum_{v=1}^{V}\sum_{i=1}^{N}\sum_{j=1}^{K} p_{ij} log \frac{p_{ij}}{q_{ij}^{(v)}} + \sum_{i=1}^{N}\sum_{j=1}^{K} p_{ij} log \frac{p_{ij}}{q_{ij}}, \end{aligned} \tag{11}$$

where the loss between the distribution q_{ij} and the target p_{ij} is also included, to strengthen the consistency among views. By iteratively forcing the distribution $\boldsymbol{Q}^{(v)}$ of each view approaching to the common target distribution $\boldsymbol{P}$ with respect to the network weights, the soft assignments of samples in each view will increasingly become similar.

3.2.4. Total objective

To simultaneously learn the graph embedding and get gradually refined clustering predictions in a unified process, the reconstruction loss and clustering loss can be jointly reduced by minimizing the total objective

$$L = L_r + \lambda L_c, \tag{12}$$

where $\lambda > 0$ is a trade-off parameter between L_r and L_c.

3.3. Optimization procedure

3.3.1. Pre-training and initialization

In order to obtain the well-initialized network before optimizing the total objective, we pre-train the model by only using the reconstruction loss L_r, with the weight ω_v of each view is initialized as $1/V$. After pre-training, we perform standard k-means (MacQueen et al., 1967) on the common embedding $\boldsymbol{H}$ and on the view-specific embedding $\boldsymbol{H}^{(v)}$ to obtain initial cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^K$ and $\{\boldsymbol{\mu}_j^{(v)}\}_{j=1}^K$ respectively.

3.3.2. Joint optimization

After pre-training and initialization, we optimize the total objective function of Eq. (12) with respect to the network parameters $\boldsymbol{W} = \{\{\boldsymbol{W}^{(v)}\}_{v=1}^V, \boldsymbol{W}_a, \boldsymbol{b}, \boldsymbol{u}\}$, the cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^K$, $\{\boldsymbol{\mu}_j^{(v)}\}_{j=1}^K$, and the view weights ω_v. In each iteration of the optimization, the cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^K$, $\{\boldsymbol{\mu}_j^{(v)}\}_{j=1}^K$ and network parameters $\boldsymbol{W}$ are simultaneously updated by using stochastic gradient descent (SGD), then the view weight ω_v is updated with Lagrangian multiplier method.

Computing the target distribution $\boldsymbol{P}$: The target distribution $\boldsymbol{P}$ that works as the supervised signal should be firstly computed by Eq. (9). Since it depends on the self-generated distribution $\boldsymbol{Q}$, to avoid instability of the self-training process, we update $\boldsymbol{P}$ every T epochs.

Fixing ω_v and updating $\boldsymbol{W}$, $\{\boldsymbol{\mu}_j\}_{j=1}^K$ and $\{\boldsymbol{\mu}_j^{(v)}\}_{j=1}^K$: The gradients of L_c with respect to the cluster centroids $\boldsymbol{\mu}_j^{(v)}$ and $\boldsymbol{\mu}_j$ are computed as

$$\begin{aligned}\frac{\partial L_c}{\partial \boldsymbol{\mu}_j^{(v)}} &= -\frac{\alpha+1}{\alpha}\sum_i (1+\frac{\|\boldsymbol{h}_i^{(v)}-\boldsymbol{\mu}_j^{(v)}\|^2}{\alpha})^{-1}(p_{ij}-q_{ij}^{(v)})(\boldsymbol{h}_i^{(v)}-\boldsymbol{\mu}_j^{(v)}),\\ \frac{\partial L_c}{\partial \boldsymbol{\mu}_j} &= -\frac{\alpha+1}{\alpha}\sum_i (1+\frac{\|\boldsymbol{h}_i-\boldsymbol{\mu}_j\|^2}{\alpha})^{-1}(p_{ij}-q_{ij})(\boldsymbol{h}_i-\boldsymbol{\mu}_j).\end{aligned} \tag{13}$$

Since the reconstruction loss L_r is irrelevant to cluster centroids $\boldsymbol{\mu}_j^{(v)}$ and $\boldsymbol{\mu}_j$, the gradients of the total loss L with respect to $\boldsymbol{\mu}_j^{(v)}$ and $\boldsymbol{\mu}_j$ are equal to the gradients of the L_c with respect to $\boldsymbol{\mu}_j^{(v)}$ and $\boldsymbol{\mu}_j$, as shown as follows

$$\frac{\partial L}{\partial \boldsymbol{\mu}_j^{(v)}} = \frac{\partial L_c}{\partial \boldsymbol{\mu}_j^{(v)}}, \frac{\partial L}{\partial \boldsymbol{\mu}_j} = \frac{\partial L_c}{\partial \boldsymbol{\mu}_j}. \tag{14}$$

Then the cluster centroids $\boldsymbol{\mu}_j^{(v)}$ and $\boldsymbol{\mu}_j$ are updated by

$$\boldsymbol{\mu}_j^{(v)} = \boldsymbol{\mu}_j^{(v)} - \eta\frac{\partial L}{\partial \boldsymbol{\mu}_j^{(v)}}, \boldsymbol{\mu}_j = \boldsymbol{\mu}_j - \eta\frac{\partial L}{\partial \boldsymbol{\mu}_j}, \tag{15}$$

where η is the learning rate. The gradient of the total loss L with respect to the network parameters $\boldsymbol{W}$ can be computed as

$$\frac{\partial L}{\partial \boldsymbol{W}} = \frac{\partial L_r}{\partial \boldsymbol{W}} + \lambda\frac{\partial L_c}{\partial \boldsymbol{W}}, \tag{16}$$

which can be obtained by standard back-propagation through the network. Then the network parameters $\boldsymbol{W}$ are updated by

$$\boldsymbol{W} = \boldsymbol{W} - \eta\frac{\partial L}{\partial \boldsymbol{W}}. \tag{17}$$

Fixing $\boldsymbol{W}$, $\{\boldsymbol{\mu}_j\}_{j=1}^K$ and $\{\boldsymbol{\mu}_j^{(v)}\}_{j=1}^K$ and updating ω_v: The weight ω_v can be updated by using the Lagrangian multiplier method. We denote

$$D_v = loss(\boldsymbol{A}^{(v)}, \boldsymbol{A}). \tag{18}$$

Then we get the Lagrangian function of Eq. (7) with respect to ω_v and γ as the follows

$$L(\omega_v, \gamma) = \sum_{v=1}^V \omega_v^\tau D_v + \gamma(\sum_{v=1}^V \omega_v - 1). \tag{19}$$

Taking derivative with respect to ω_v yields

$$\frac{\partial L(\omega_v, \gamma)}{\partial \omega_v} = \tau\omega_v^{\tau-1}D_v + \gamma. \tag{20}$$

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

$$\begin{cases}\tau\omega_v^{\tau-1}D_v + \gamma = 0,\\ \sum_{v=1}^V \omega_v = 1.\end{cases} \tag{21}$$

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

$$\omega_v = \frac{1}{\sum_{v'=1}^V (\frac{D_v}{D_{v'}})^{\frac{1}{\tau-1}}}, \quad \tau > 1. \tag{22}$$

In this expression, we can see that the smaller the loss D_v of a view is, the larger the weight ω_v on this view will be. A smaller D_v of a view reflects that the common graph is closer to the graph of this view, which means this view contributes more to the generation of the common graph and thereby should be given a larger weight.

After iteratively updating the above parameters for a specific number of iterations, the final clustering result is obtained from the final optimized $\boldsymbol{Q}$, and the cluster label δ_i for the ith sample can be obtained by

$$\delta_i = \arg\max_j q_{ij}. \tag{23}$$

The above optimization procedure is summarized in Algorithm 1.

Algorithm 1 The *DMVGC*.

Input: Multi-view dataset $\{\boldsymbol{X}^{(v)}, \boldsymbol{A}^{(v)}\}_{v=1}^V$; number of clusters K; trade-off parameter λ; exponent τ; number of training epochs E; Updating interval T; view weight $\omega_v = 1/V$.
Output: Cluster labels $\{\delta_i\}_{i=1}^N$.
Initialize: Network parameters $\boldsymbol{W}$ pre-trained by minimizing Eq. (7); Initial cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^K$ and $\{\boldsymbol{\mu}_j^{(v)}\}_{j=1}^K$.
Method:

```
for e = 0 : E − 1 do
  if e%T == 0 then
    Calculate soft assignment distribution Q by Eq. (8);
    Calculate target distribution P by Eq. (9);
  end if
  Update μ_j^(v) and μ_j by Eq. (15);
  Update W by Eq. (17);
  Update ω_v by Eq. (22);
end for
Calculate the final optimized Q by Eq. (8);
Get the cluster labels {δ_i}_{i=1}^N with final Q by Eq. (23).
```

4. Experiments

We implement the proposed model in Python with Tensorflow 1.12.0, and evaluate its performance on six real datasets. We first describe the experimental settings, and then present the experimental results in detail.

4.1. Experimental settings

4.1.1. Datasets descriptions

We conduct the experiments on two types of multi-view datasets. One type is multi-view graphs with common features (Cora, Citeseer and IMDB). The other type is multi-view features with no pre-defined graphs (Mfeat, Scene, Reuters), and the graph for each view is constructed from sample features. The detailed information about these datasets are shown as the following.

- **Cora**[1] is a citation network dataset about scientific publications, which consists of 2708 documents classified into seven classes.

[1] https://relational.fit.cvut.cz/dataset/CORA

Table 2
Datasets details of multi-view graphs with common features.

Datasets	Nodes	Classes	Views	Edges1	Edges2	Features
Cora	2708	7	2	5429	6391	1433
Citeseer	3312	6	2	4598	4894	3703
IMDB	3287	3	2	23736	4217	1232

Table 3
Datasets details of multi-view features with no pre-defined graph.

Datasets	Nodes	Classes	Views	Features1	Features2	Features3	Features4	Features5
Mfeat	2000	10	3	216	76	240	–	–
Scene	2688	8	3	512	432	256	–	–
Reuters	1200	6	5	2000	2000	2000	2000	2000

Each document in the citation network is described by a bag-of-word feature vector. For our multi-view graph learning, in addition to the original citation graph, we use k-NN algorithm (Fix & Hodges, 1989) to construct an additional graph view from the feature vectors by setting $k = 3$, which makes the number of edges most approximate to the original graph. And two views share the common bag-of-words features.

- **Citeseer**[2] is also a citation network dataset about scientific publications, which consists of 3312 documents classified into six classes. It also contains one original citation graph and bag-of-words features. We use the same way as Cora to construct an additional graph view by k-NN algorithm with setting $k = 2$.
- **IMDB**[3] is a movie network from the IMDB dataset which contains 3287 movies. Graphs of two views are extracted including co-actor (movies are acted by the same actor) relationship and co-director (movies are directed by the same director) relationship. Movie features are the bag-of-words features of plots shared by two views.
- **Mfeat**[4] consists of multiple published features extracted from 2000 handwritten digit (0–9) images, with 200 images per digit. We use three published features to construct the multi-view dataset: profile correlations, Fourier coefficients of the character shapes, and pixel averages in 2×3 windows.
- **Scene** (Monadjemi, Thomas, & Mirmehdi, 2002) 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.
- **Reuters**[5] consists of documents which are written in five different languages and their translations. All the documents are classified into six categories. We choose the subset that are written in English and translated in all the other 4 languages (French, German, Spain, Italian). Each language can be regarded as a view. Following the work Bisson and Grimal (2012), 1200 documents are randomly sampled over six categories in a balanced manner, with the dimensions of words vector reduced to 2000 by applying k-medoids algorithm.[6]

Detailed information about these datasets is summarized in Tables 2 and 3.

4.1.2. Implementation details

For the datasets that have multi-view features with no pre-defined graph, we construct each view-specific graph using k-NN algorithm based on the similarities among samples. For the datasets Mfeat and Scene, since the features are continuous and relatively lower dimensional, we apply Gaussian kernel to compute similarities among samples. For the dataset Reuters, since the features are very sparse and high dimensional, we apply cosine similarity to compute similarities among samples. For all these three datasets, we set $k = 9$ nearest neighbors to generate the k-NN graph for each view.

For the network settings of the proposed model, the dimensions of the two-layer GCN encoder for each view are set to 32 and 16 respectively, and hyperparameters λ and τ are set to $\lambda = 1.0$ and $\tau = 16$ for all datasets, based on the analysis in Section 4.2.6. We train the network with Adam optimizer (Kingma & Ba, 2015) and set the learning rate to 0.001 for Citeseer, Cora and IMDB and 0.01 for Mfeat, Scene and Reuters . For the pre-training phase, the network is trained for 400 epochs. For the joint optimization phase, the network is fine-tuned for 100 epochs. And we set the updating interval $T = 5$ for Citeseer, IMDB and Mfeat and $T = 20$ for Cora, Scene and Reuters. For the cluster centroids initialization in the joint optimization phase, we randomly select K samples as the initial cluster centroids, and perform k-means on the initial common embedding and each view-specific embedding to obtain the initial cluster centroids for the subsequent training.

4.1.3. Baseline methods

The proposed method DMVGC is compared with several state-of-the-art methods including CoregSC, AWGL and SwMC that only use multi-view graphs, LMSC and MKKM that only use multi-view features, and GMNMF, O2MAC, CMGEC and MVGC that use both multi-view graphs and features. Since the datasets Citeseer, Cora and IMDB only have single-view features, LMSC and MKKM are only performed on the datasets Mfeat, Scene and Reuters that have multi-view features. Additionally, DMVGC is compared with BSV/WSV that represents the best and worst results of the variant single-view method on each single-view data, and DMVGA that performs multi-view graph embedding learning and clustering in two-step manner.

(1) Methods only using multi-view graphs

- **CoregSC** (Kumar et al., 2011): A multi-view clustering method by using the co-regularized idea on spectral clustering of multiple views.
- **AWGL** (Nie et al., 2016):An automatically weighted multi-view spectral clustering method.
- **SwMC**(Nie et al., 2017): A self-weighted multi-view clustering with multiple graphs.

(2) Methods only using multi-view features

- **LMSC** (Zhang et al., 2020): A multi-view subspace clustering method based on latent representation.
- **MKKM** (Liu et al., 2016): A multiple kernel k-means clustering method with matrix-induced regularization.

(3) Methods using both multi-view graphs and features

- **GMNMF** (Rai et al., 2016): A graph regularized nonnegative matrix factorization based multi-view clustering method.

[2] https://relational.fit.cvut.cz/dataset/CiteSeer
[3] https://www.imdb.com
[4] https://archive.ics.uci.edu/ml/datasets/Multiple+Features
[5] https://archive.ics.uci.edu/ml/datasets
[6] http://membres-lig.imag.fr/grimal/data.html

- **O2MAC** (Fan et al., 2020):A GCN-based deep model that conducts multi-view graph clustering based on the one-to-multi graph autoencoder that encodes one most informative view and extracts the shared representations by all views.
- **CMGEC** (Wang et al., 2023): A GCN-based deep model that conducts multi-view graph clustering based on a novel consistent multiple graph embedding clustering framework.
- **MVGC** (Xia, Wang, Yang et al., 2022):A GCN-based deep model that uses the learned clustering labels to self-supervise the learning of node representation and the view-consensus coefficient matrix.
- **DMVGA**: A variant of DMVGC, which does not include the self-training clustering loss in the objective function. Instead, the embedding is trained with reconstruction loss only, then followed by k-means performed on the embedding, in two-step manner.

(4) Single-view clustering

- **BSV/WSV**: A single-view variant of DMVGC, which is a single-view graph autoencoder (Kipf & Welling, 2016) jointly trained with the self-training clustering. We report the best single-view results and the worst single-view results for each dataset.

For the state-of-the-art methods, we follow the experimental settings in their papers. For the variant method BSV/WSV, we set its encoder network with the same settings as the view-specific encoder network of DMVGC, and set it with the same trade-off parameter as DMVGC. For the variant method DMVGA, we set it with the same network settings as DMVGC, and perform k-means on the embedding obtained from the pre-trained network. For fair comparisons, all graph-based methods employ the same graph construction approach (k-NN with $k = 9$) as DMVGC on Mfeat, Scene and Reuters, and all methods involving k-means operation apply random initialization. Except for SwMC that we only run once since it has no random initializations, we perform all methods 10 times and report the average clustering performance. The clustering performance is measured by three clustering evaluation metrics including NMI (normalized mutual information), ACC (accuracy) and ARI (adjusted rand index).

4.1.4. Evaluation metrics

The mathematical definitions about the clustering evaluation metrics NMI, ACC and ARI are given in the this section. Firstly, the assigned cluster labels are best mapped to the true labels by Munkres algorithm (Munkres, 1957). 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

$$NMI = \frac{-2\sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij} log(\frac{n_{ij}n}{n_{i\cdot}n_{\cdot j}})}{\sum_{i=1}^{k} n_{i\cdot} log(\frac{n_{i\cdot}}{n}) + \sum_{j=1}^{k} n_{\cdot j} log(\frac{n_{\cdot j}}{n})}, \tag{24}$$

$$ACC = \frac{1}{n}\sum_{i=1}^{k} n_{ii}, \tag{25}$$

$$ARI = \frac{\sum_{ij} C_{n_{ij}}^2 - [\sum_i C_{n_{i\cdot}}^2 \sum_j C_{n_{\cdot j}}^2]/C_n^2}{\frac{1}{2}[\sum_i C_{n_{i\cdot}}^2 + \sum_j C_{n_{\cdot j}}^2] - [\sum_i C_{n_{i\cdot}}^2 \sum_j C_{n_{\cdot j}}^2]/C_n^2}. \tag{26}$$

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 of 10 executions.

4.2. Experimental results

4.2.1. Clustering performance comparisons

The clustering evaluation metrics on six datasets are shown in Tables 5 and 6. For each dataset, the best result among all methods under each metric is indicated in bold, and the second best result is indicated with both italics and underline. It can be seen that DMVGC outperforms the baseline methods on most datasets, except for a few results slightly lower than certain other methods. Especially for the dataset Mfeat, all three metrics of DMVGC are lower than MVGC which iteratively uses the spectral clustering labels to self-supervise the learning of deep representation. This two-step self-supervision mechanism based on hard labels benefits for the dataset with inherently better cluster separability, since more samples are likely to obtain accurate clustering labels in the initial stages thereby will propagate more correct information to produce increasingly more accurate results. On the contrary, for the dataset with inherently poor cluster separability, more inaccurate labels supervise the learning and dominate the network to produce inferior results. Therefore, MVGC performs exceptionally well on the dataset Mfeat which exhibits inherently good cluster separability, but performs much poorer than our method on the datasets Citeseer, Cora, IMDB and Reuters. According to the results of WSV and BSV on all datasets, it can be observed that DMVGC based on multi-view information outperforms the variant single-view model WSV and BSV. Furthermore, on all datasets, DMVGC by jointly performing graph embedding learning and self-training clustering improves the clustering performance over DMVGA that performs graph embedding extraction and k-means clustering in two-step manner.

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

We also conduct the non-parametric tests for different methods (except for LMSC and MKKM that do not perform on all datasets) over all datasets under each metric. We first employ Friedman test to test difference degree among methods. The probability values in Friedman test for NMI, ACC and ARI are 1.133×10^{-4}, 3.731×10^{-6} and 4.072×10^{-6} respectively, which are all less than the significant level 0.1, demonstrating a significant difference among different methods. Then we conduct Nemenyi test and Wilcoxon signed-rank test under each metric to test whether DMVGC significantly outperforms each baseline method. The numerical results are shown in Table 7, and the CD diagram (Critical Difference diagram) visualization for Nemenyi test is shown in Fig. 2. For the numerical results, the value less than the significant level 0.1 demonstrates a significant improvement of DMVGC over the baseline method. For the CD diagram, the average ranking of each method over all datasets is marked along the axis, and any two methods are significantly different if the distance of their average rankings exceeds the critical distance (unconnected by red line). For Nemenyi test, both the numerical results and CD diagram indicate that, under each metric, DMVGC has significant improvements against the methods CoregSC, AWGL, SwMC that only use multi-view graphs, and the method WSV that uses the worst single view. According to the rankings of methods shown in CD diagrams, we can see that DMVGC and its variant method DMVGA rank the first and the second respectively, and all GCN-based multi-view graph clustering methods including DMVGC, DMVGA, O2MAC, CMGEC and MVGC rank in the top five. Even the GCN-based single-view model BSV ranks ahead of the shallow multi-view methods. Since the number of the compared methods is much higher than the number of the datasets, the capability of the Nemenyi test may be limited. Thus we also conduct the Wilcoxon signed-rank test, which is used to compare the differences between paired methods. From Table 7, it can be seen that, for Wilcoxon signed-rank test, DMVGC has significant improvement over almost all baseline methods under all metrics.

Table 5
Clustering performance comparisons on datasets that are composed of multi-view graphs with common features.

Dataset	Method	NMI	ACC	ARI
Citeseer	CoregSC	0.1122(0.0748)	0.2988(0.0655)	0.0806(0.0659)
	AWGL	0.0101(0.0018)	0.2131(0.0008)	0.0005(0.0004)
	SwMC	0.0349(0.0000)	0.2135(0.0000)	0.0006(0.0000)
	GMNMF	0.3953(0.0006)	0.6553(0.0009)	0.4026(0.0010)
	O2MAC	0.3685(0.0116)	0.6255(0.0172)	0.3716(0.0158)
	CMGEC	0.3001(0.0194)	0.5488(0.0262)	0.2756(0.0274)
	MVGC	0.3520(0.0486)	0.5747(0.0887)	0.3156(0.0789)
	WSV	0.2221(0.0315)	0.4852(0.0429)	0.2040(0.0348)
	BSV	0.2617(0.0282)	0.5000(0.0474)	0.2189(0.0388)
	DMVGA	*0.4130(0.0044)*	*0.6691(0.0128)*	*0.4248(0.0089)*
	DMVGC	**0.4380(0.0064)**	**0.6908(0.0124)**	**0.4570(0.0114)**
Cora	CoregSC	0.1821(0.0169)	0.3459(0.0162)	0.0571(0.0208)
	AWGL	0.1861(0.0789)	0.3493(0.0623)	0.0745(0.0878)
	SwMC	0.1076(0.0000)	0.3146(0.0000)	0.0071(0.0000)
	GMNMF	0.4202(0.0030)	0.5039(0.0241)	0.2910(0.0262)
	O2MAC	0.4747(0.0161)	0.6280(0.0259)	0.3953(0.0196)
	CMGEC	0.4319(0.0174)	0.6051(0.0300)	0.3660(0.0229)
	MVGC	0.4217(0.0581)	0.5466(0.0448)	0.2632(0.0744)
	WSV	0.2981(0.0185)	0.5037(0.0299)	0.2437(0.0185)
	BSV	0.4502(0.0285)	0.5877(0.0400)	0.3541(0.0398)
	DMVGA	*0.5186(0.0082)*	*0.6858(0.0273)*	*0.4553(0.0148)*
	DMVGC	**0.5359(0.0079)**	**0.6934(0.0235)**	**0.4702(0.0139)**
IMDB	CoregSC	0.0063(0.0015)	0.3876(0.0017)	0.0045(0.0010)
	AWGL	0.0027(0.0015)	0.3874(0.0016)	0.0007(0.0008)
	SwMC	**0.0943(0.0000)**	0.3623(0.0000)	0.0047(0.0000)
	GMNMF	0.0308(0.0060)	0.4174(0.0188)	0.0156(0.0079)
	O2MAC	0.0846(0.0143)	*0.4897(0.0216)*	*0.0961(0.0172)*
	CMGEC	0.0494(0.0060)	0.4448(0.0197)	0.0450(0.0100)
	MVGC	0.0112(0.0005)	0.3887(0.0035)	0.0070(0.0006)
	WSV	0.0138(0.0079)	0.3893(0.0183)	0.0138(0.0092)
	BSV	0.0318(0.0107)	0.4153(0.0156)	0.0302(0.0097)
	DMVGA	0.0703(0.0020)	0.4488(0.0050)	0.0552(0.0008)
	DMVGC	*0.0846(0.0044)*	**0.5107(0.0195)**	**0.1047(0.0087)**

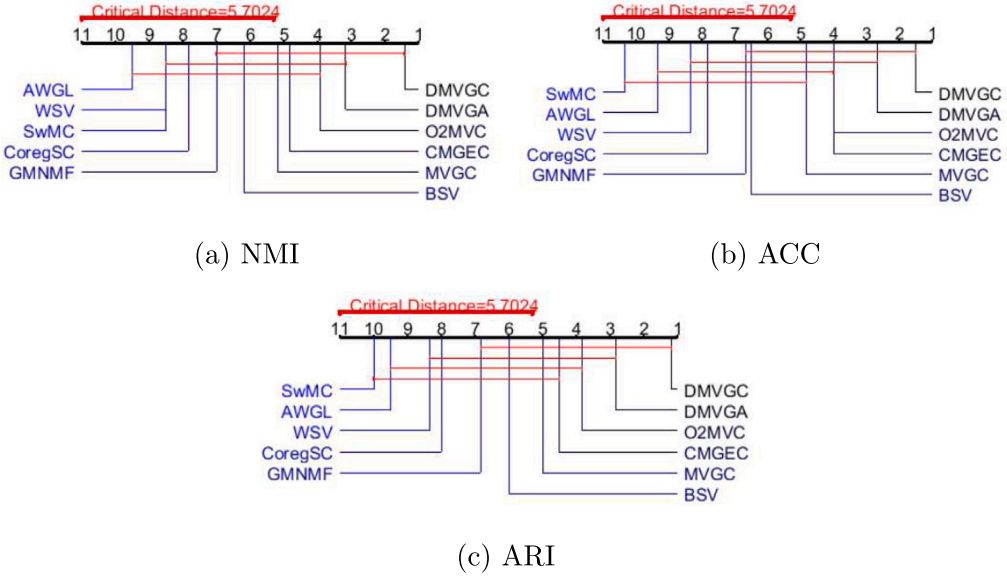


Fig. 2. Critical difference diagram under each metric.

4.2.2. *Effect of weighting mechanism*

To see the effect of two weighting mechanisms: the attention module for the embedding fusion and the adaptive weighting for the joint reconstruction, we evaluate the performance of three variant models: the proposed model without attention module (DMVGC-a), the proposed model without adaptive weights on reconstruction loss (DMVGC-w), and the proposed model without both attention module and adaptive weights (DMVGC-aw). For DMVGC-a, we remove the attention module and set the view-specific attention $\alpha^{(v)} = 1/V$. For DMVGC-w, we remove the adaptive weights ω_v and construct the reconstruction loss as $L_r = \sum_{v=1}^{V} loss(\boldsymbol{A}^{(v)}, \boldsymbol{A})$. The clustering performance of these variant models are shown in Table 8. It can be seen that the proposed model outperforms all variant models on all datasets, except for the NMI metric of DMVGC slightly lower than DMVGC-a on IMDB. This indicates that both the attention module and the adaptive weighting for the reconstruction loss can enhance the performance of the proposed model, which demonstrates the effectiveness of these two weighting mechanisms.

Table 6
Clustering performance comparisons on datasets that are composed of multi-view features with no pre-defined graphs.

Dataset	Method	NMI	ACC	ARI
Mfeat	LMSC	0.7887(0.0113)	0.8269(0.0469)	0.7324(0.0249)
	MKKM	0.7876(0.0022)	0.8673(0.0016)	0.7427(0.0031)
	CoregSC	0.9188(0.0216)	0.9401(0.0597)	0.9058(0.0542)
	AWGL	0.8528(0.0244)	0.7861(0.0549)	0.7406(0.0648)
	SwMC	0.8903(0.0000)	0.8660(0.0000)	0.8184(0.0000)
	GMNMF	0.8717(0.0286)	0.8573(0.0668)	0.8123(0.0631)
	O2MAC	0.9199(0.0241)	0.9527(0.0426)	0.9120(0.0456)
	CMGEC	0.8875(0.0297)	0.9125(0.0534)	0.8563(0.0570)
	MVGC	**0.9429(0.0016)**	**0.9749(0.0006)**	**0.9447(0.0014)**
	WSV	0.4462(0.1468)	0.4068(0.1580)	0.2866(0.1471)
	BSV	0.8971(0.0294)	0.9053(0.0744)	0.8642(0.0705)
	DMVGA	0.9183(0.0119)	0.9530(0.0332)	0.9111(0.0316)
	DMVGC	*0.9359(0.0097)*	*0.9638(0.0288)*	*0.9314(0.0282)*
Scene	LMSC	0.5227(0.0086)	0.6810(0.0364)	0.4533(0.0125)
	MKKM	0.4736(0.0042)	0.6218(0.0036)	0.3981(0.0033)
	CoregSC	0.5426(0.0177)	0.6447(0.0505)	0.4541(0.0249)
	AWGL	0.5264(0.0007)	0.5786(0.0004)	0.3674(0.0012)
	SwMC	0.0394(0.0000)	0.1574(0.0000)	0.0000(0.0000)
	GMNMF	0.4263(0.0115)	0.5339(0.0292)	0.3140(0.0114)
	O2MAC	0.4467(0.0247)	0.5519(0.0368)	0.3372(0.0342)
	CMGEC	*0.5674(0.0072)*	**0.7058(0.0057)**	0.4737(0.0094)
	MVGC	0.5622(0.0038)	*0.7003(0.0351)*	*0.4793(0.0146)*
	WSV	0.3219(0.0100)	0.4316(0.0234)	0.2161(0.0203)
	BSV	0.4216(0.0683)	0.5285(0.1082)	0.3221(0.0837)
	DMVGA	0.5560(0.0099)	0.6845(0.0303)	0.4652(0.0183)
	DMVGC	**0.5735(0.0109)**	0.6936(0.0315)	**0.4810(0.0199)**
Reuters	LMSC	0.1816(0.0031)	0.3950(0.0070)	0.1287(0.0038)
	MKKM	0.2845(0.0007)	0.4941(0.0012)	0.2066(0.0008)
	CoregSC	0.2149(0.0038)	0.3600(0.0125)	0.1122(0.0046)
	AWGL	0.1900(0.0209)	0.2682(0.0373)	0.0469(0.0317)
	SwMC	0.1310(0.0000)	0.2258(0.0000)	0.0066(0.0000)
	GMNMF	0.2812(0.0081)	0.4208(0.0152)	0.1716(0.0073)
	O2MAC	0.3207(0.0263)	0.5310(0.0358)	0.2490(0.0234)
	CMGEC	0.3560(0.0269)	0.5583(0.0391)	0.2632(0.0278)
	MVGC	0.2960(0.0105)	0.4268(0.0131)	0.1872(0.0124)
	WSV	0.3028(0.0259)	0.5212(0.0463)	0.2327(0.0228)
	BSV	0.3173(0.0320)	0.5226(0.0440)	0.2395(0.0313)
	DMVGA	*0.3653(0.0349)*	*0.5780(0.0536)*	*0.3009(0.0367)*
	DMVGC	**0.3769(0.0415)**	**0.5887(0.0580)**	**0.3138(0.0367)**

Table 7
Nemenyi test (NT) and Wilcoxon signed-rank test (WT) for indicating the significant difference of DMVGC with each baseline method over all datasets under each metric. The probability values in the table indicate the significance degree, and the value less than the significance level 0.1 demonstrates a significant improvement of DMVGC over this baseline method.

Method	NMI		ACC		ARI	
	NT	WT	NT	WT	NT	WT
CoregSC	**0.0330**	**0.0277**	**0.0380**	**0.0277**	**0.0158**	**0.0277**
AWGL	**0.0012**	**0.0277**	**0.0021**	**0.0277**	**0.0010**	**0.0277**
SwMC	**0.0099**	**0.0464**	**0.0010**	**0.0277**	**0.0010**	**0.0277**
GMNMF	0.1184	**0.0277**	0.2004	**0.0277**	0.1052	**0.0277**
O2MAC	0.9000	**0.0431**	0.9000	**0.0277**	0.9000	**0.0277**
CMGEC	0.7626	**0.0277**	0.9000	**0.0464**	0.7896	**0.0277**
MVGC	0.6543	**0.0464**	0.7896	0.1159	0.6272	**0.0747**
WSV	**0.0099**	**0.0277**	**0.0158**	**0.0277**	**0.0084**	**0.0277**
BSV	0.3160	**0.0277**	0.2425	**0.0277**	0.2905	**0.0277**
DMVGA	0.9000	**0.0277**	0.9000	**0.0277**	0.9000	**0.0277**

4.2.3. Analysis for collaborative training

To demonstrate that the collaborative training for clustering of DMVGC can gradually align the clustering results of different views during the training process, we randomly select a sample from Mfeat dataset and visualize its cluster distribution $q_i^{(v)}$ of each view-specific embedding and the cluster distribution q_i of the common embedding during the training process. As shown in Fig. 3, the view-specific distributions $q_i^{(1)}$, $q_i^{(2)}$ and $q_i^{(3)}$ in the first three columns and the common distribution q_i in the last column along rows of epoch 0, 25, 50, 75, 100 are presented. Before the collaborative training (in epoch 0), the cluster with the highest probability in $q_i^{(1)}$ is the 9-th cluster while the highest probability in others lies in the 8th cluster. This indicates that the cluster assignment of the first view is not aligned with other views. Then after starting the collaborative training, the probability of the 8th cluster in each view gradually rises with training epochs. Finally in epoch 100, the probability of the 8th cluster in $q_i^{(1)}$ also becomes the highest, meaning that the cluster assignment of the first view achieves alignment with other views. This demonstrates the effectiveness of the collaborative training in our model.

4.2.4. Clustering performance during training

To observe how the clustering performance of DMVGC changes during the optimization process, we plot the changing curve of the NMI metric over the training epochs for each dataset, as shown in Fig. 4. The curve for each dataset shows an overall ascending trend despite some fluctuations, until rising to a certain level, then it fluctuates around

Table 8
Clustering performance comparisons of DMVGC with the variant models.

Dataset	Method	NMI	ACC	ARI
Citeseer	DMVGC	**0.4380(0.0064)**	**0.6908(0.0124)**	**0.4570(0.0114)**
	DMVGC-a	0.3867(0.0367)	0.6320(0.0506)	0.3853(0.0545)
	DMVGC-w	0.2819 (0.0401)	0.5425(0.0394)	0.2594(0.0452)
	DMVGC-aw	0.2794 (0.0208)	0.5361(0.0204)	0.2595(0.0195)
Cora	DMVGC	**0.5359(0.0079)**	**0.6934(0.0235)**	**0.4702(0.0139)**
	DMVGC-a	0.4936(0.0236)	0.6493(0.0303)	0.4261(0.0337)
	DMVGC-w	0.4538(0.0303)	0.6198(0.0560)	0.3814(0.0456)
	DMVGC-aw	0.4387(0.0256)	0.6091(0.0336)	0.3621(0.0279)
IMDB	DMVGC	0.0846(0.0044)	**0.5107(0.0195)**	**0.1047(0.0087)**
	DMVGC-a	**0.0870(0.0072)**	0.4905(0.0137)	0.0961(0.0105)
	DMVGC-w	0.0707(0.0049)	0.4641(0.0118)	0.0636(0.0063)
	DMVGC-aw	0.0696(0.0075)	0.4617(0.0143)	0.0653(0.0077)
Mfeat	DMVGC	0.9359(0.0097)	**0.9638(0.0288)**	**0.9314(0.0282)**
	DMVGC-a	**0.9376(0.0123)**	0.9594(0.0347)	0.9257(0.0356)
	DMVGC-w	0.9286(0.0177)	0.9488(0.0440)	0.9122(0.0459)
	DMVGC-aw	0.9256(0.0179)	0.9542(0.0350)	0.9147(0.0372)
Scene	DMVGC	**0.5735(0.0109)**	**0.6936(0.0315)**	**0.4810(0.0199)**
	DMVGC-a	0.5514(0.0153)	0.6573(0.0460)	0.4455(0.0095)
	DMVGC-w	0.5540(0.0371)	0.6775(0.0613)	0.4576(0.0380)
	DMVGC-aw	0.5552(0.0264)	0.6729(0.0510)	0.4580(0.0324)
Reuters	DMVGC	**0.3769(0.0415)**	**0.5887(0.0580)**	**0.3138(0.0367)**
	DMVGC-a	0.3687(0.0346)	0.5781(0.0513)	0.3043(0.0336)
	DMVGC-w	0.3478(0.0248)	0.5765(0.0435)	0.2979(0.0239)
	DMVGC-aw	0.3717(0.0203)	0.5786(0.0446)	0.3031(0.0275)

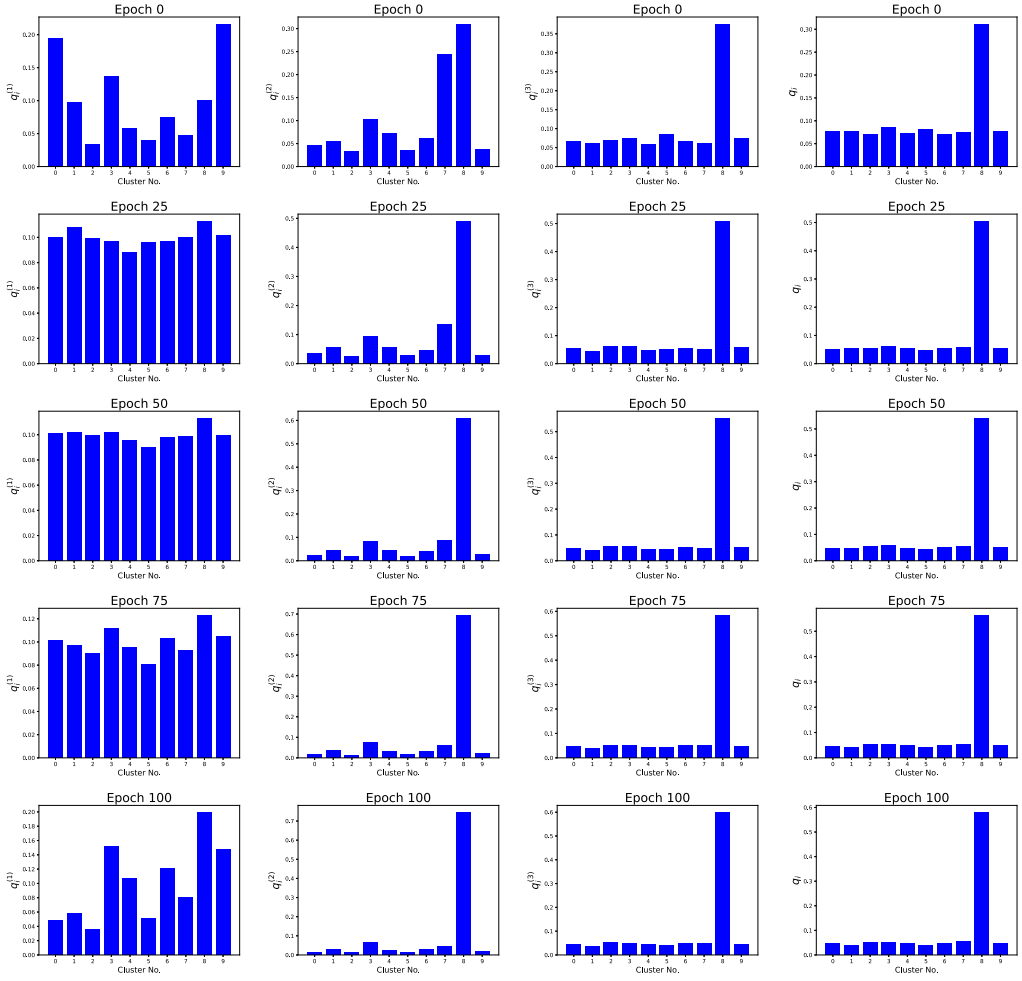


Fig. 3. Cluster distributions of a randomly selected sample from Mfeat on each view-specific embedding and common embedding during the collaborative training process.

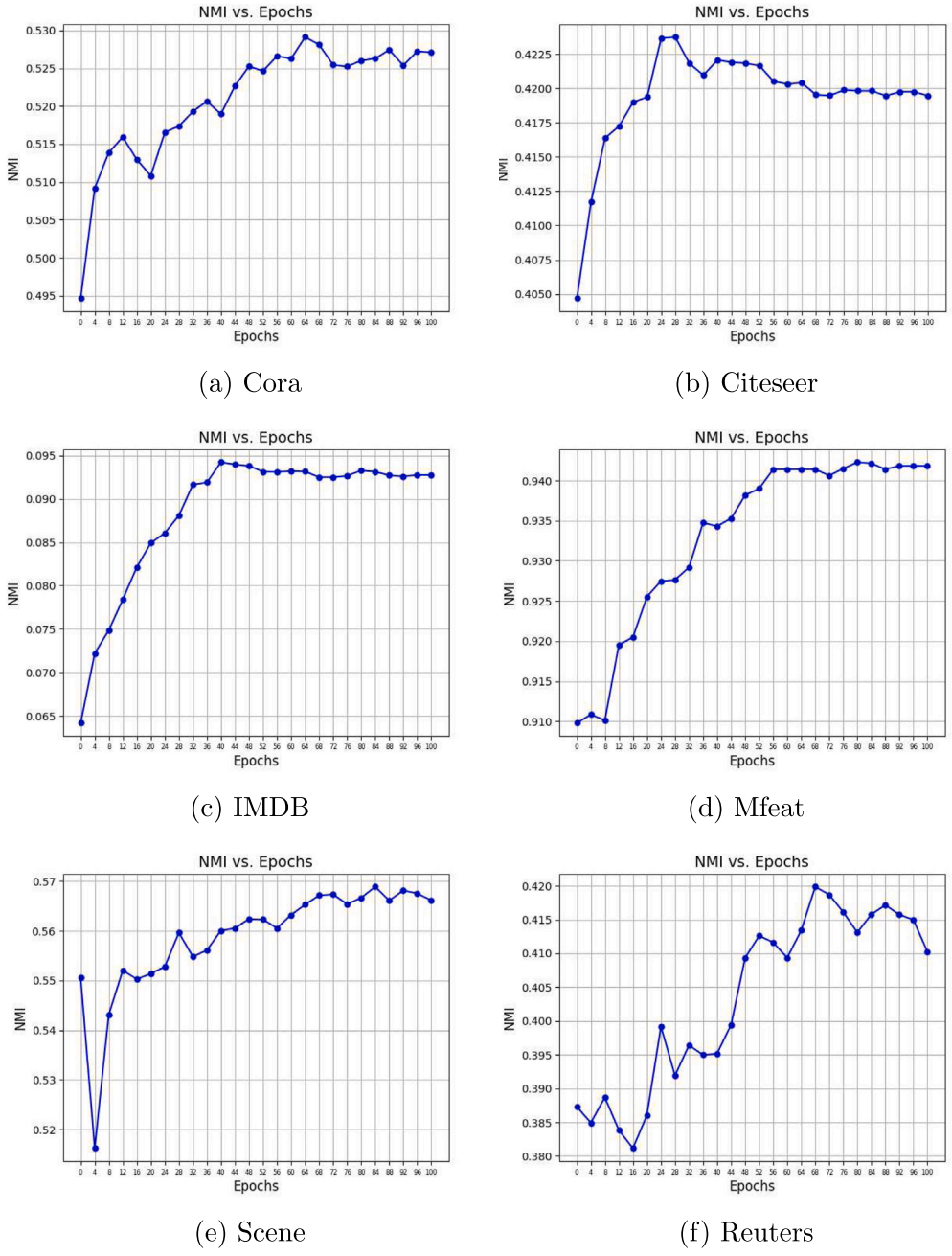


Fig. 4. The NMI over training epochs for each dataset.

this level. These results show that our model works toward a desired direction.

4.2.5. Visualization

(1) Visualization of gradients

The underlying assumption of the self-training clustering module of DMVGC is that the initial highly confident assignments are mostly correct and contribute more gradients to the back-propagation network training. To verify this assumption for our task, we plot the magnitude of the gradient of L_c with respect to each embedded point $\boldsymbol{h}_i$, i.e. $|\frac{\partial L_c}{\partial \boldsymbol{h}_i}|$, against its soft assignment q_{ij}, for a random chosen cluster j for each dataset, at the start of KL divergence minimization, as shown in Fig. 5. We observe that points with higher confidence (larger q_{ij}) are more likely to contribute higher gradient (with higher $|\frac{\partial L_c}{\partial \boldsymbol{h}_i}|$), which will back propagates through the GCN encoder of each view, making the learned embedded points become increasingly cluster-discriminative. Additionally, we best map the cluster assignments of the initial embedded points with the true labels, and color the points with right predictions as blue and color the points with wrong predictions as red. We can see that the number of right points is much greater than the number of wrong points, and most of the right points are in the high confidence area while most of the wrong points are in the low confidence area. Therefore, the right points will dominate the training process to increasingly produce refined clusters.

(2) Visualization of embedding features

We visualize the original features of the first view for Mfeat dataset and the common graph embedding features of DMVGC during the training process in the epoch 0, 50 and 100, by using t-SNE method (van der Maaten & Hinton, 2008). As shown in Fig. 6, the visualization in epoch 0 represents the embedding features pre-trained with the reconstruction

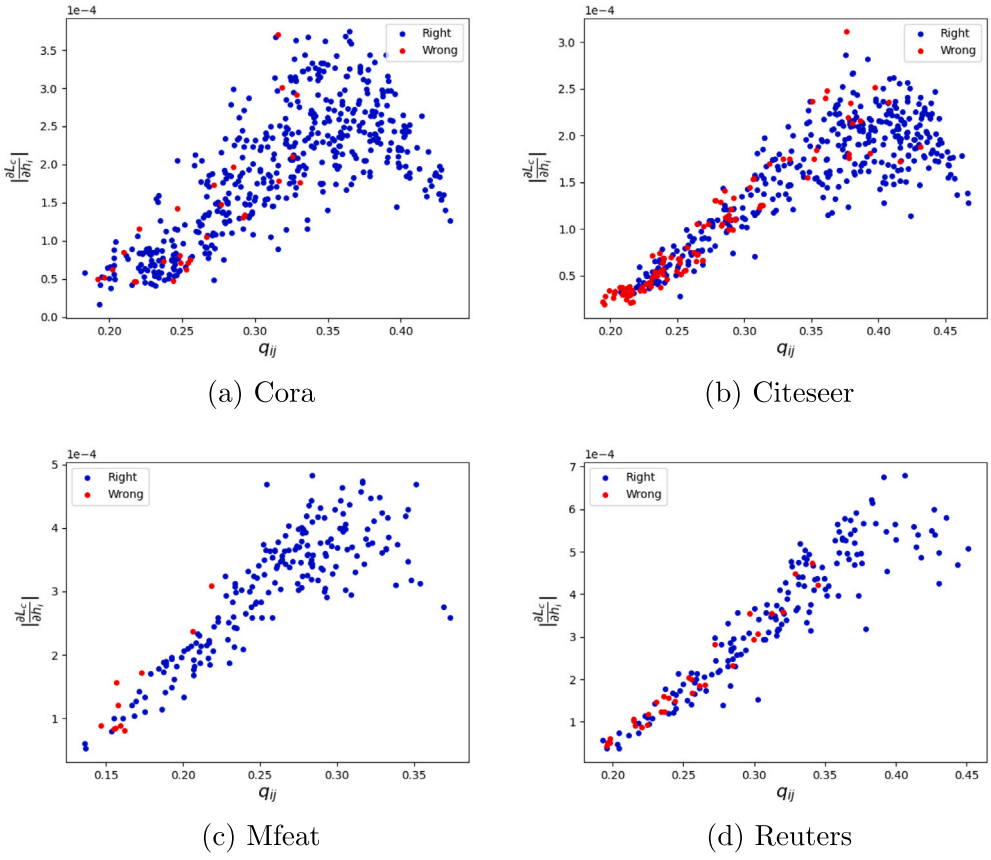


Fig. 5. Gradients visualization at the start of KL divergence minimization.

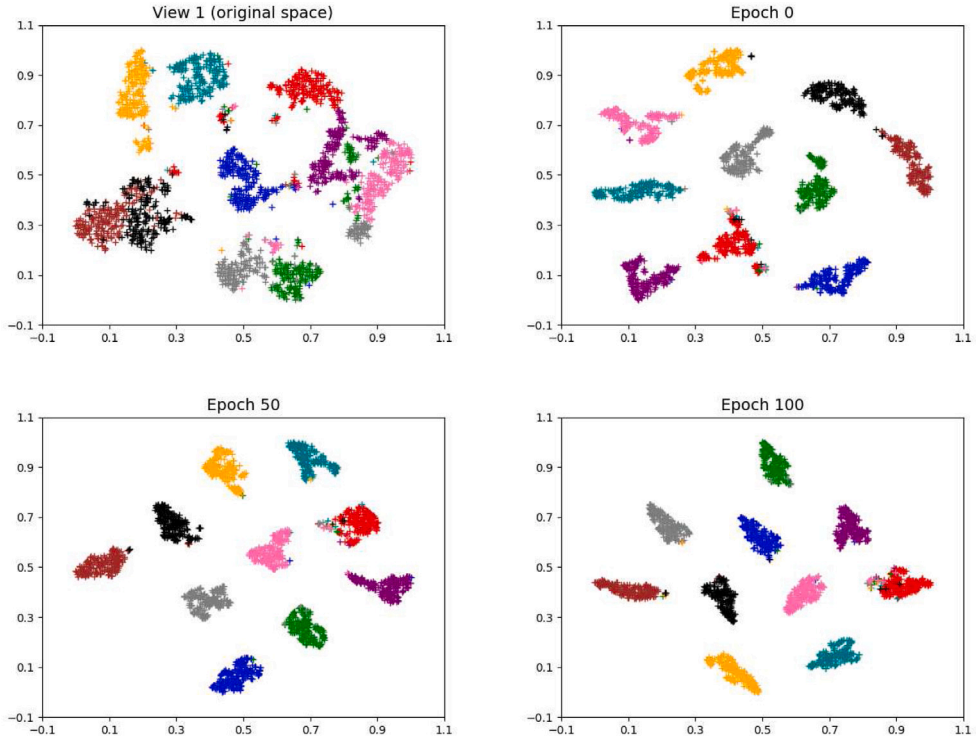
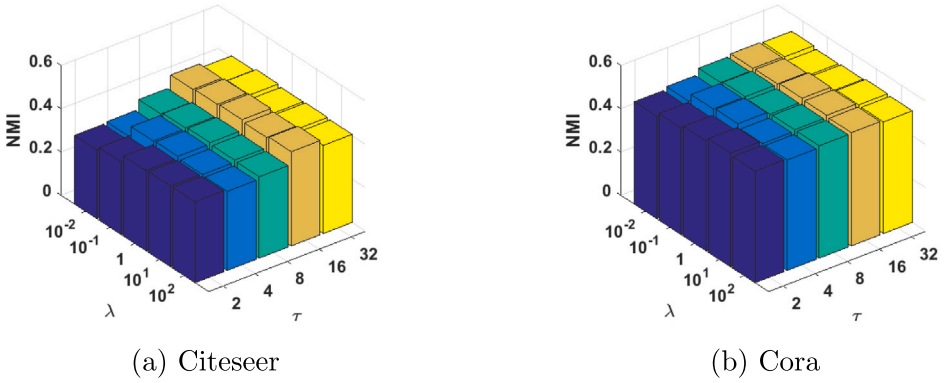
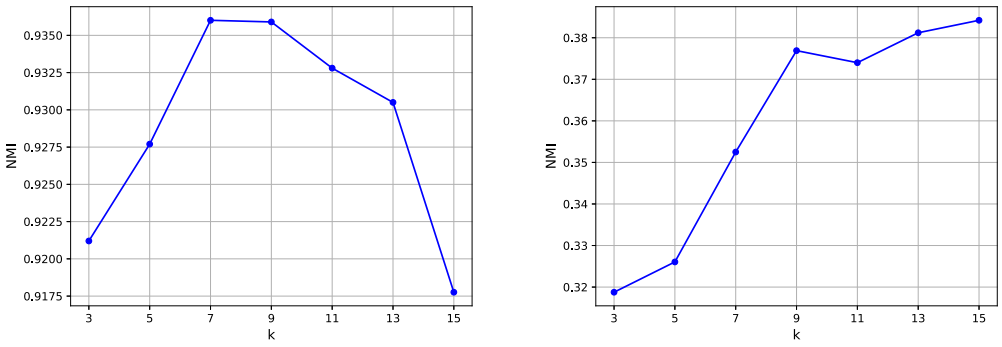


Fig. 6. *t*-SNE visualization for embedding features of DMVGC on the Mfeat dataset during training.



(a) Citeseer (b) Cora

Fig. 7. The parameter effect of τ and λ on Citeseer and Cora.



(a) Mfeat (b) Reuters

Fig. 8. The parameter effect of k on Mfeat and Reuters.

loss only, followed by visualizations for the embedding features in subsequent epoch 50 and 100, in which the self-training clustering module is included. It is obvious that the clusters are becoming increasingly well-separated as the training proceeds, which corresponds with the ascending trend of the NMI curve over epochs as shown in Fig. 4(d).

4.2.6. Parameter analysis

(1) Parameter effect of τ and λ

In our model, there are two hyper-parameters τ and λ that should be set properly. In our experiments, we tune τ and λ from $\{2, 4, 8, 16, 32\}$ and $\{0.01, 0.1, 1, 10, 100\}$ respectively. The NMI results under different values of τ and λ on the datasets Citeseer and Cora are shown in Fig. 7. For the parameter τ, larger τ value generally produces higher performance, and the performance becomes stable when $\tau \geq 16$. For the parameter λ, we can see that our model performs stably over a wide range of values. Therefore, in our experiments, we set $\tau = 16$ and $\lambda = 1$ for all datasets.

(2) Parameter effect of k

For the k-NN graph construction for the data with no pre-defined graph, the number of the nearest neighbors k is a critical parameter that may greatly influence the performance of our model. To see the effect of k, we tune k from 3 to 15 with step size 2 to construct graphs on the datasets Mfeat and Reuters, and report the NMI results under each value of k. From Fig. 8, we can see that when k is relatively small the performance rises with the increasing of k. Until reaching to a relatively stable level, the performance fluctuates around this level or even declines, since with k continues to increase, the separability of the graph may not improve but even deteriorate. And the larger the k, the more edges of the graph, which takes more time for the optimization. Therefore in our experiments, we set k to a moderate value as $k = 9$ for all datasets that have no pre-defined graphs.

5. Conclusion

In this paper, we propose a GCN-based deep multi-view graph clustering network with weighting mechanism and collaborative training. The forward propagation of multiple encoders and the back-propagation driven by the unified decoder incorporate both the diversity and unity of different views into the learned common embedding. The weighting mechanisms are employed in both encoder and decoder parts to measure the importance of different views, and a collaborative training objective for clustering is simultaneously optimized with the graph embedding to align the clustering results of different views. Experimental results on six datasets verifies the effectiveness and rationality of our method, and demonstrates the superiority of our method over the state-of-the-art methods. For future studies, we can focus on improving the collaborative training and weighting mechanism to make the model applicable to the problem of incomplete multi-view clustering.

CRediT authorship contribution statement

Jing Liu: Investigation, Conceptualization, Methodology, Experiment, Writing. **Fuyuan Cao:** Guidance, Writing, Resources, Supervision. **Xuechun Jing:** Writing. **Jiye Liang:** Guidance, Resources.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61976128, 62376145, 62072293, 62022052), the Science and Technology Innovation Talent Team of Shanxi Province, China (Grant No. 202204051002016), the Shanxi Province Basic Research Program, China (Grant No. 202203021212416), the Open Project Foundation of Intelligent Information Processing Key Laboratory of Shanxi Province, China (Grant No. CICIP2021005), and the 1331 Engineering Project of Shanxi Province, China.

References

Bisson, G., & Grimal, C. (2012). Co-clustering of multi-view datasets: A parallelizable approach. In *2012 IEEE 12th international conference on data mining* (pp. 828–833). http://dx.doi.org/10.1109/ICDM.2012.93.

Cai, X., Nie, F., & Huang, H. (2013). Multi-view K-means clustering on big data. In *Proceedings of the twenty-third international joint conference on artificial intelligence* (pp. 2598–2604).

Chaudhuri, K., Kakade, S. M., Livescu, K., & Sridharan, K. (2009). Multi-view clustering via canonical correlation analysis. *Vol. 382*, In *Proceedings of the 26th annual international conference on machine learning* (pp. 129–136). http://dx.doi.org/10.1145/1553374.1553391.

Chen, M.-S., Huang, L., Wang, C.-D., & Huang, D. (2020). Multi-view clustering in latent embedding space. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (04), (pp. 3513–3520). http://dx.doi.org/10.1609/aaai.v34i04.5756.

Cheng, J., Wang, Q., Tao, Z., Xie, D., & Gao, Q. (2020). Multi-view attribute graph convolution networks for clustering. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence* (pp. 2973–2979). http://dx.doi.org/10.24963/ijcai.2020/411.

Du, G., Zhou, L., Yang, Y., Lü, K., & Wang, L. (2021). Deep multiple auto-encoder-based multi-view clustering. *Data Science and Engineering*, *6*(3), 323–338. http://dx.doi.org/10.1007/s41019-021-00159-z.

Fan, S., Wang, X., Shi, C., Lu, E., Lin, K., & Wang, B. (2020). One2Multi graph autoencoder for multi-view graph clustering. In *WWW '20: The web conference 2020* (pp. 3070–3076). http://dx.doi.org/10.1145/3366423.3380079.

Fang, S.-G., Huang, D., Cai, X.-S., Wang, C.-D., He, C., & Tang, Y. (2023). Efficient multi-view clustering via unified and discrete bipartite graph learning. *IEEE Transactions on Neural Networks and Learning Systems*, 1–12. http://dx.doi.org/10.1109/TNNLS.2023.3261460.

Fang, U., Li, M., Li, J., Gao, L., Jia, T., & Zhang, Y. (2023). A comprehensive survey on multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, 1–20. http://dx.doi.org/10.1109/TKDE.2023.3270311.

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, *57*(3), 238–247.

Fu, L., Lin, P., Vasilakos, A. V., & Wang, S. (2020). An overview of recent multi-view clustering. *Neurocomputing*, *402*, 148–161. http://dx.doi.org/10.1016/j.neucom.2020.02.104.

Gao, H., Nie, F., Li, X., & Huang, H. (2015). Multi-view subspace clustering. In *2015 IEEE international conference on computer vision* (pp. 4238–4246). http://dx.doi.org/10.1109/ICCV.2015.482.

Huang, Z., Ren, Y., Pu, X., Pan, L., Yao, D., & Yu, G. (2021). Dual self-paced multi-view clustering. *Neural Networks*, *140*, 184–192. http://dx.doi.org/10.1016/j.neunet.2021.02.022.

Huang, D., Wang, C.-D., & Lai, J.-H. (2023). Fast multi-view clustering via ensembles: Towards scalability, superiority, and simplicity. *IEEE Transactions on Knowledge and Data Engineering*, http://dx.doi.org/10.1109/TKDE.2023.3236698.

Kang, Z., Lin, Z., Zhu, X., & Xu, W. (2022). Structured graph learning for scalable subspace clustering: From single view to multiview. *IEEE Transactions on Cybernetics*, *52*(9), 8976–8986. http://dx.doi.org/10.1109/TCYB.2021.3061660.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations*.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. In *NIPS workshop on bayesian deep learning*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th international conference on learning representations*.

Kumar, A., Rai, P., & Daumé, H. (2011). Co-regularized multi-view spectral clustering. In *Proceedings of the 24th international conference on neural information processing systems* (pp. 1413–1421).

Li, X., Zhang, H., & Zhang, R. (2022). Adaptive graph auto-encoder for general data clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *44*(12), 9725–9732. http://dx.doi.org/10.1109/TPAMI.2021.3125687.

Liang, N., Yang, Z., Li, Z., Sun, W., & Xie, S. (2020). Multi-view clustering by non-negative matrix factorization with co-orthogonal constraints. *Knowledge-Based Systems*, *194*, Article 105582. http://dx.doi.org/10.1016/j.knosys.2020.105582.

Lin, J.-Q., Chen, M.-S., Zhu, X.-R., Wang, C.-D., & Zhang, H. (2022). Dual information enhanced multi-view attributed graph clustering. arXiv preprint arXiv:2211.14987.

Lin, Z., Kang, Z., Zhang, L., & Tian, L. (2023). Multi-view attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering*, *35*(2), 1872–1880. http://dx.doi.org/10.1109/TKDE.2021.3101227.

Liu, J., Cao, F., Gao, X.-Z., Yu, L., & Liang, J. (2020). A cluster-weighted kernel K-means method for multi-view clustering. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (04), (pp. 4860–4867). http://dx.doi.org/10.1609/aaai.v34i04.5922, URL https://ojs.aaai.org/index.php/AAAI/article/view/5922.

Liu, J., Cao, F., & Liang, J. (2022). Centroids-guided deep multi-view K-means clustering. *Information Sciences*, *609*, 876–896. http://dx.doi.org/10.1016/j.ins.2022.07.093.

Liu, X., Dou, Y., Yin, J., Wang, L., & Zhu, E. (2016). Multiple kernel *k*-means clustering with matrix-induced regularization. In *Proceedings of the thirtieth AAAI conference on artificial intelligence* (pp. 1888–1894). http://dx.doi.org/10.1609/aaai.v30i1.10249.

Liu, J., Wang, C., Gao, J., & Han, J. (2013). Multi-view clustering via joint nonnegative matrix factorization. In *Proceedings of the 2013 SIAM international conference on data mining* (pp. 252–260). http://dx.doi.org/10.1137/1.9781611972832.28.

van der Maaten, L., & Hinton, G. (2008). Viualizing data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. *Vol. 1*, In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (14), (pp. 281–297).

Monadjemi, A., Thomas, B. T., & Mirmehdi, M. (2002). Experiments on high resolution images towards outdoor scene classification. In *Computer vision winter workshop*.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, *5*(1), 32–38.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning* (pp. 807–814).

Nie, F., Li, J., & Li, X. (2016). 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* (pp. 1881–1887).

Nie, F., Li, J., & Li, X. (2017). Self-weighted multiview clustering with multiple graphs. In *Proceedings of the twenty-sixth international joint conference on artificial intelligence* (pp. 2564–2570). http://dx.doi.org/10.24963/ijcai.2017/357.

Pan, E., & Kang, Z. (2021). Multi-view contrastive graph clustering. *Vol. 34*, In *Advances in neural information processing systems* (pp. 2148–2159).

Rai, N., Negi, S., Chaudhury, S., & Deshmukh, O. (2016). Partial multi-view clustering using graph regularized NMF. In *23rd international conference on pattern recognition* (pp. 2192–2197). http://dx.doi.org/10.1109/ICPR.2016.7899961.

Rasiwasia, N., Mahajan, D., Mahadevan, V., & Aggarwal, G. (2014). Cluster canonical correlation analysis. *Vol. 33*, In *Proceedings of the seventeenth international conference on artificial intelligence and statistics* (pp. 823–831).

Tang, C., Liu, X., Zhu, X., Zhu, E., Luo, Z., Wang, L., et al. (2020). CGD: Multi-view clustering via cross-view graph diffusion. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (04), (pp. 5924–5931). http://dx.doi.org/10.1609/aaai.v34i04.6052.

Wang, Y., Chang, D., Fu, Z., & Zhao, Y. (2023). Consistent multiple graph embedding for multi-view clustering. *IEEE Transactions on Multimedia*, *25*, 1008–1018. http://dx.doi.org/10.1109/TMM.2021.3136098.

Wang, C., Pan, S., Hu, R., Long, G., Jiang, J., & Zhang, C. (2019). Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of the twenty-eighth international joint conference on artificial intelligence* (pp. 3670–3676). http://dx.doi.org/10.24963/ijcai.2019/509.

Wang, H., Yang, Y., Liu, B., & Fujita, H. (2019). A study of graph-based system for multi-view clustering. *Knowledge-Based Systems*, *163*, 1009–1019. http://dx.doi.org/10.1016/j.knosys.2018.10.022.

Wang, Y., Zhang, W., Wu, L., Lin, X., Fang, M., & Pan, S. (2016). Iterative views agreement: An iterative low-rank based structured optimization method to multi-view spectral clustering. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 2153–2159).

Xia, W., Wang, Q., Gao, Q., Yang, M., & Gao, X. (2022). Self-consistent contrastive attributed graph clustering with pseudo-label prompt. *IEEE Transactions on Multimedia*, 1–13. http://dx.doi.org/10.1109/TMM.2022.3213208.

Xia, W., Wang, T., Gao, Q., Yang, M., & Gao, X. (2023). Graph embedding contrastive multi-modal representation learning for clustering. *IEEE Transactions on Image Processing*, *32*, 1170–1183. http://dx.doi.org/10.1109/TIP.2023.3240863.

Xia, W., Wang, S., Yang, M., Gao, Q., Han, J., & Gao, X. (2022). Multi-view graph embedding clustering network: Joint self-supervision and block diagonal representation. *Neural Networks*, *145*, 1–9. http://dx.doi.org/10.1016/j.neunet.2021.10.006.

Xie, J., Girshick, R. B., & Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. *vol. 48*, In *Proceedings of the 33nd international conference on machine learning* (pp. 478–487).

Xie, Y., Lin, B., Qu, Y., Li, C., Zhang, W., Ma, L., et al. (2021). Joint deep multi-view learning for image clustering. *IEEE Transactions on Knowledge and Data Engineering*, *33*(11), 3594–3606. http://dx.doi.org/10.1109/TKDE.2020.2973981.

Xu, J., Ren, Y., Li, G., Pan, L., Zhu, C., & Xu, Z. (2021). Deep embedded multi-view clustering with collaborative training. *Information Sciences*, *573*, 279–290. http://dx.doi.org/10.1016/j.ins.2020.12.073.

Xu, J., Ren, Y., Tang, H., Yang, Z., Pan, L., Yang, Y., et al. (2023). Self-supervised discriminative feature learning for deep multi-view clustering. *IEEE Transactions on Knowledge and Data Engineering*, *35*(7), 7470–7482. http://dx.doi.org/10.1109/TKDE.2022.3193569.

Xu, Y.-M., Wang, C.-D., & Lai, J.-H. (2016). Weighted multi-view clustering with feature selection. *Pattern Recognition*, *53*, 25–35. http://dx.doi.org/10.1016/j.patcog.2015.12.007.

Yin, Q., Wu, S., He, R., & Wang, L. (2015). Multi-view clustering via pairwise sparse subspace representation. *Neurocomputing*, *156*, 12–21. http://dx.doi.org/10.1016/j.neucom.2015.01.017.

Zhang, C., Fu, H., Hu, Q., Cao, X., Xie, Y., Tao, D., et al. (2020). Generalized latent multi-view subspace clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(1), 86–99. http://dx.doi.org/10.1109/TPAMI.2018.2877660.

Zhang, H., Li, P., Zhang, R., & Li, X. (2022). Embedding graph auto-encoder for graph clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 1–11. http://dx.doi.org/10.1109/TNNLS.2022.3158654.

Zhao, J., Kang, F., Zou, Q., & Wang, X. (2023). Multi-view clustering with orthogonal mapping and binary graph. *Expert Systems with Applications*, *213*, Article 118911. http://dx.doi.org/10.1016/j.eswa.2022.118911.