Dropping Pathways Towards Deep Multi-View Graph Subspace Clustering Networks

Zihao Zhang
Xidian University
Xi'an, Shaanxi, China
zihaoz2021@foxmail.com

Qianqian Wang*
Xidian University
Xi'an, Shaanxi, China
qqwang@xidian.edu.cn

Zhiqiang Tao
Rochester Institute of Technology
Rochester, NY, USA
zhiqiang.tao@rit.edu

Quanxue Gao
Xidian University
Xi'an, Shaanxi, China
qxgao@xidian.edu.cn

Wei Feng
Xi'an Jiaotong University
Xi'an, Shaanxi, China
weifeng.ft@xjtu.edu.cn

ABSTRACT

Multi-view graph clustering aims to leverage different views to obtain consistent information and improve clustering performance by sharing the graph structure. Existing multi-view graph clustering algorithms generally adopt a single-pathway network reconstruction and consistent feature extraction, building on top of auto-encoders and graph convolutional networks (GCN). Despite their promising results, these single-pathway methods may ignore the significant complementary information between different layers and the rich multi-level context inside. On the other hand, GCN usually employs a shallow network structure (2-3 layers) due to the over-smoothing with the increase of network depth, while few multi-view graph clustering methods explore the performance of deep networks. In this work, we propose a novel Dropping Pathways strategy toward building a deep Multi-view Graph Subspace Clustering network, namely DPMGSC, to fully exploit the deep and multi-level graph network representations. The proposed method implements a multi-pathway self-expressive network to capture pairwise affinities of graph nodes among multiple views. Moreover, we empirically study the impact of a series of dropping methods on deep multi-pathway networks. Extensive experiments demonstrate the effectiveness of the proposed DPMGSC compared with its deep counterpart and state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Cluster analysis**; **Learning latent representations**.

KEYWORDS

Multi-view clustering, multi-pathway networks, graph convolutional networks.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '23, October 29–November 3, 2023, Ottawa, ON, Canada.
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0108-5/23/10…$15.00
https://doi.org/10.1145/3581783.3612332

ACM Reference Format:
Zihao Zhang, Qianqian Wang*, Zhiqiang Tao, Quanxue Gao, and Wei Feng. 2023. Dropping Pathways Towards Deep Multi-View Graph Subspace Clustering Networks. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3581783.3612332

1 INTRODUCTION

Deep learning has been widely applied in processing structured data but cannot handle the ubiquitous irregular data with a spatial topology such as molecular structure [8], paper citation network [2], and social network [12], because of their graph data structure. To tackle this challenge, graph representation learning becomes extremely necessary, which aims to convert graph data into low-dimensional and dense/compact embedding vectors. Early methods include Matrix Factorization [3, 5], DeepWalk [21], and Graph Neural Networks [10, 24, 34]. Among these methods, graph convolutional network (GCN) [15], which can process data with a generalized topological graph structure by aggregating local neighbour features, shows its effectiveness in graph embedding and has become one of the mainstream methods.

High cost of data annotation makes it significant to develop unsupervised graph learning methods. Among these, graph clustering that divides nodes into different clusters is an important unsupervised task, whose performance highly relies the effective representation. To this end, Graph Auto-encoder (GAE) [14] combining auto-encoders and GCNs was proposed to improve unsupervised graph embedding performance. Along this line of sight, numerous variant methodologies were proposed. For example, variational graph auto-encoder (V-GAE) [14] introduces variational auto-encoder (VAE) to leverage inner products of latent variables to reconstruct adjacency matrices and then learn node representations. Aiming at better robustness, adversarially regularized variational graph auto-encoder (ARVGA) [20] further introduces an adversarial learning module on the basis of VAGE and effectively improves representation capacity. However, the aforementioned methods adopts global shared weights for neighbour information of a node, resulting in poor generalization ability. To overcome the weakness, Graph Attention Auto-Encoders (GATE) [23] assigns different weights to different nodes to balance the impact of each neighbour on the node representation [27].

Despite the effectiveness of the above methods on single-view graph clustering, they are not applicable to multi-view graph clustering. Extensive works [29, 30, 37] have shown that multi-view data provide more abundant information than single-view data. Therefore, several GCN-based multi-view graph clustering methods were developed [7, 18, 35]. However, existing models are designed to learn a consistent representation for each view from single-pathway networks, resulting in that representations solely originate from the deepest layers and lose valuable complementary information. Since directly ignoring valuable information from shallow layers, these single-pathway methods somehow miss the comprehensive complementary and multi-level information between different layers. Thus, one of the challenges encountered in multi-view representation learning is how to achieve compatibility with consistency and complementarity. Another challenge is the *over-smoothing* [17] issue of GCN-based methods, i.e., the hidden layer representation of each node tends to converge to the same value with the increase of layers and iterations. Hence, GCN-based multi-view graph clustering methods usually exhibit poor expressiveness due to the difficulty in obtaining information from high-order neighbours, and these methods generally adopt relatively shallow structures (2-3 layers).

Inspired by the single-view deep GCN methods [11, 22, 39], we propose a novel Dropping Pathways strategy to realize a deep Multi-view Graph Subspace Clustering network, namely DPMGSC. Our method employs multiple pathways to explore more complementary information when ensuring consistency. Simultaneously, it deepens the network to learn high-order neighbour relationship and alleviates the over-smoothing problem with dropping-pathway strategy. Furthermore, by constraining the distribution in both the original space and latent subspace, our approach ensures the consistency of reliable low-dimensional node representations.

- We develop a novel multi-view graph clustering by utilizing multi-pathway to embed the complementary and multi-level information into a consistent subspace shared by different views. In addition, we impose consistency constraints on the distribution of the original data and embedding features.
- We introduce a dropping-pathway mechanism to alleviate the over-smoothing issue of multi-pathway GCNs for the clustering task, by systematically investigating three dropping strategies. To the best of our knowledge, this is the first study to explore the multi-view graph clustering performance with the increase of network depth (8 layers).
- Extensive experimental results on four graph datasets empirically demonstrate the superiority of the proposed dropping strategy for multi-pathway networks as the depth increases.

2 RELATED WORK

2.1 GCN for clustering

By stacking multiple graph convolutional layers, GCN aggregates its own features and those of its neighbours to extract high-level node representations, which is a pivotal part of many models for clustering tasks. GCN-based models usually try different approaches to obtain reliable latent node representations, while Graph Autoencoder (GAE) methods usually use various reconstruction constraints to learn network embeddings. For example, Deep Attentional Embedding Approach Graph Clustering (DAEGC) [28] jointly optimizes graph embedding and graph clustering and obtains impressive clustering results; Embedding Graph Auto-Encoder for Graph Clustering (EGAE) [38] introduces embedding relaxed k-means into GAE to induce the network to generate better embeddings. On this basis, some studies [23, 27, 32] pay attention to the neighbour information of nodes and adopt the self-attention mechanism.

Although the above methods have achieved effective clustering results, they are designed for single-view data but cannot handle the ubquitious multi-view data in practice. Thus, numerous multi-view graph clustering techniques were developed. For example, Cheng et al. proposed Multi-view Attribute Graph Convolutional Network (MAGCN) [7] with an attention mechanism and a consistency penalty on feature representations across multiple views. Co-training GCN (Co-GCN) [18] regards both the graph structure and node attributes as the description of the view, and constructs the adjacency matrix of each view and its combined graph accordingly. With the GCN encoder, Co-GCN obtains the respective feature representations and aggregates them into the final common representation for clustering; Self-supervised Graph Convolutional Network for Multi-view Clustering (SGCMC) [35] aims to address outliers in graph structures and utilizes clustering labels for guided training. The above methods are more inclined to mine the consistent information of multiple views while ignoring their complementary information that is also essential. Differently, our method proposes multi-pathway networks to exploit complementary information and multi-level information.

2.2 Deep GCNs

Although GCN is a glamorous graph representation learning method, it may suffer from the over-smoothing issue when deepening the network [15]. To solve the problem, Deep GCNs [16] apply residual/dense connections and dilated convolutions to GCN, which significantly improves performance in point cloud semantic segmentation tasks; GCNII [6] with initial residual and identity mapping, achieves advanced results in various semi-supervised and fully supervised node classification. Even with residual connections, GCN still gets unsatisfactory results after deepening the network. Hence, DropEdge [22] randomly discards some fixed-proportion edges during training, and by making node connections more sparse, it avoids the over-smoothing problem to a certain extent. Thus, we are motivated to deepen the network and explore the clustering performance of multi-pathway networks under different dropping strategies.

3 PROPOSED METHODOLOGY

3.1 Notation

Suppose an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, $\mathbf{V}$ denotes the node set and $\mathbf{E}$ denotes the edge set. We define node attributes as $\{\mathbf{X}_m\}_{m=1}^{M}$, where $\mathbf{X}_m = \{\boldsymbol{x}_m^1, \boldsymbol{x}_m^2, ..., \boldsymbol{x}_m^n\}^T \in \mathbb{R}^{n \times d_m}$; n denotes the number of nodes and d_m denotes the dimension of the m-th modality. For better clarity, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ is used to represent the topology of $\mathbf{G}$. A summary of the notations used throughout our paper can be found in Table 1. It should be noted that the vector will be represented in both *italic* and **bold** to avoid confusion.

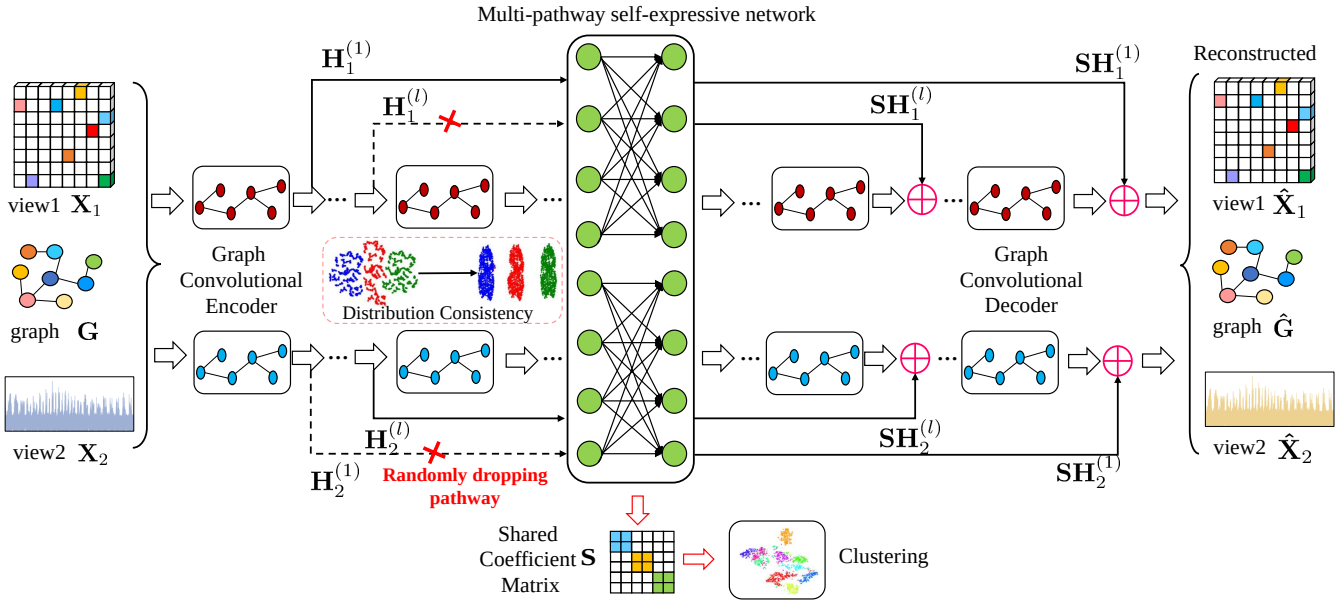


Figure 1: The framework of DPMGSC. It consists of three parts: a graph convolutional auto-encoders network, a multi-pathway self-expressive network, and a distribution consistency network. The model is dedicated to learning a consistent affinity matrix S with complementary information and multi-level information, which from node attributes X_1, X_2 and graph structure A.

3.2 The framework of DPMGSC

The model framework is composed of a graph convolutional auto-encoder network, a distribution consistency network, and a multi-pathway self-expressive network, as shown in Figure 1.

Graph Convolutional Auto-encoder: Through the ensemble learning of graph structure $\mathbf{A}$ and node attributes $\mathbf{X}_m$, the encoder maps the initial node attribute to the latent space of graph embedding to obtain the low-dimensional latent representation $\mathbf{H}_m$, i.e., $f(\mathbf{A}, \mathbf{X_m}; \theta) \rightarrow \mathbf{H}_m \in \mathbb{R}^{n \times d}$, where θ represents the trainable parameters of the encoder, and d represents the dimension after dimensionality reduction. For ease of understanding, we use $\mathbf{H}$ instead of $\mathbf{H}_m$, utilizing subscript only when needing to distinguish different views. Considering the l-th layer ($l \in \{0, 1, 2, ..., L\}$), the equation of the graph convolutional encoder can be written as

$$\mathbf{H}^{(l)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}^{*}\mathbf{D}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l)}\right), \tag{1}$$

where $\mathbf{A}^{*} = \mathbf{A} + \mathbf{I}_N$ means adding the identity matrix $\mathbf{I}_N$ to form a self-loop; $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ denotes the degree matrix of $\mathbf{A}$, and $\mathbf{W}^{(l)}$ represents the l-th layer parameter matrix composed of parameters θ; σ denotes activation function. Similarly, for the reconstruction of node attributes $\hat{\mathbf{X}}$ and graph structure $\hat{\mathbf{A}}$, its input $\hat{\mathbf{H}}^{(l)}$ and output $\hat{\mathbf{H}}^{(l-1)}$ follow the following equation:

$$\hat{\mathbf{H}}^{(l-1)} = \sigma\left(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}'\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{H}}^{(l)}\hat{\mathbf{W}}^{(l)}\right), \tag{2}$$

where $\hat{\mathbf{H}}^{(0)}$ is the reconstructed node attribute matrix $\hat{\mathbf{X}}$.

Applying symmetric normalization to $\mathbf{A}$ helps prevent information transfer distortion that may occur in GNNs and leads to more reasonable aggregation of neighbour information. However, node representations in GCN are learned by aggregating information from their neighbours, as shown in Eq. (1), which implies that the weight distribution is determined by the fixed graph structure $\mathbf{A}$. This approach may constrain the model's ability to learn desirable features.

To address this, an alternative solution is to learn the weights adaptively, enabling the model to have more flexibility in dealing with the over-smoothing problem. Therefore, the relevant matrix $\mathbf{R}$ will be introduced to capture the correlation between nodes and their neighbours and attempt to dynamically adjust $\mathbf{A}$:

Table 1: Main notations in this paper.

notations	Definition
n	Number of nodes
M	Number of views
k	Number of clusters
θ	Parameters of auto-encoders
ξ	Parameters of fully connected layer
$\{\boldsymbol{\mu}_j\}_{j=1}^{k}$	Cluster centroids
$\mathbf{X}_m \in \mathbb{R}^{n \times d_m}$	Node attribute matrix of m-th view
$\mathbf{H}^{(l)} \in \mathbb{R}^{n \times d}$	Latent features matrix of the l-th layer
$\mathbf{A} \in \mathbb{R}^{n \times n}$	Adjacency matrix of graph $\mathbf{G}$
$\mathbf{D} \in \mathbb{R}^{n \times n}$	Degree matrix of $\mathbf{A}$
$\mathbf{R}^{(l)} \in \mathbb{R}^{n \times n}$	Relevance matrix of the l-th layer
$\mathbf{S} \in \mathbb{R}^{n \times n}$	Consistent representation matrix
$\mathbf{C} \in \mathbb{R}^{n \times n}$	Affinity matrix $\mathbf{C} = \frac{1}{2}(\lvert\mathbf{S}\rvert + \lvert\mathbf{S}\rvert^{T})$

$$\mathbf{A}_{ij}^{(l)} = \frac{\exp\left(\mathbf{R}_{ij}{}^{(l)}\right)}{\sum_{k\in\mathbf{N}_i}\exp\left(\mathbf{R}_{ik}^{(l)}\right)}, \tag{3}$$

$$\mathbf{R}^{(l)} = \phi\left(\mathbf{A}\odot \boldsymbol{t}_s^{(l)}\mathbf{H}^{(l)}\mathbf{W}^{(l)} + \mathbf{A}\odot \boldsymbol{t}_r^{(l)}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{4}$$

where $\mathbf{R}_{ij}^{(l)}$ denotes the relevance matrix between nodes i and j at the l-th layer; $\mathbf{N}_i$ refers to the set of neighbours of node i; ϕ denotes a sigmoid function and $\odot$ is the Hadamard product with broadcasting capability; $\boldsymbol{t}_s^{(l)}$ and $\boldsymbol{t}_r^{(l)}$ denote trainable linear transformation vectors that are related to the own node and neighbour nodes. After the correction and scaling of the relevance matrix, the $\mathbf{A}$ obtained from the Eq. (3) helps to get a better node representation.

With the node attribute $\mathbf{X}$ and the reconstructed node attribute $\hat{\mathbf{X}}$, its reconstruction loss function can be defined as

$$\mathcal{L}_{reX} = \min_{\theta}\sum_{m=1}^{M}\left\|\mathbf{X}_m - \hat{\mathbf{X}}_m\right\|_F^2, \tag{5}$$

where $\|\cdot\|_F$ represents Frobenius norm.

For a node i from $\mathbf{H}^{(l)}$, it is expected to have a certain level of similarity with its neighbour nodes $j(j\in\mathbf{N}_i)$, which leads to the reconstruction of the graph. The reconstruction will be performed by an inner product decoder $\phi(\cdot)$, specifically $\hat{\mathbf{A}}_{ij} = \phi(-\boldsymbol{h}^{i\top}\boldsymbol{h}^j)$, and it adheres to the following constraints:

$$\mathcal{L}_{reA} = \min_{\theta}\left\|\mathbf{A} - \hat{\mathbf{A}}\right\|_F^2. \tag{6}$$

Multi-pathway Self-expressive Network: Assuming that a node can be linearly represented by other nodes in the same subspace, self-expressive learning can be described as $\mathbf{X}_m = \mathbf{S}\mathbf{X}_m$, where $\mathbf{S}\in\mathbb{R}^{n\times n}$ is the self-expressive coefficient matrix. To implement the idea of self-expression that is difficult to achieve, we formulate the optimization objective as

$$\min_{\mathbf{S}}\ \|\mathbf{S}\|_F + \|\mathbf{X}_m - \mathbf{S}\mathbf{X}_m\|_F^2\ \ s.t.\ diag(\mathbf{S}) = 0, \tag{7}$$

where $\|\cdot\|_F$ is Frobenius norm, $diag(\cdot)$ means to take the diagonal elements to avoid trivial solution $\mathbf{S} = \mathbf{I}$.

Through successive graph convolutions, node representations are able to perceive information from more distant neighbour nodes. However, restricting the self-expression layer to only incorporate features from the deepest layer ($\mathbf{H}^{(l)}$) may lead to the loss of some comprehensive complementary and multi-level information.

Assuming the encoder consists of l layers with M views, we adaptively adjust the weights of each layer using trainable parameters $\boldsymbol{\beta}$. The final loss function of the multi-pathway self-expressive network is defined as

$$\begin{aligned}\mathcal{L}_{se} = \min_{\mathbf{S}}\sum_{l=1}^{L}\sum_{m=1}^{M}\boldsymbol{\beta}_l\|\mathbf{H}_m^{(l)} - \mathbf{S}\mathbf{H}_m^{(l)}\|_F^2 + \|\mathbf{S}\|_F \\ s.t.\ diag(\mathbf{S}) = 0,\ \sum_{l=1}^{L}\boldsymbol{\beta}_l = 1.\end{aligned} \tag{8}$$

When initializing $\boldsymbol{\beta}$, we opt for a simple approach of initializing each pathway with equal weights to ensure the fairness of the pathways and the consistency of the experiment.

Distribution Consistency Network: Reconstruction constraint can ensure that its low-dimensional features are potentially reliable, but it cannot preserve the consistency of node distribution before and after dimensionality reduction. Thus, we add a constraint to enforce the consistency between the original space and the latent subspace. Given a nonlinear map $g(\mathbf{H}, \boldsymbol{\mu}; \xi)$ and an initial estimate of cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^k$ initialized by K-means, we use Student's t-distribution as a kernel to measure the difference between node embeddings $\boldsymbol{h}^i$ and centroid $\boldsymbol{\mu}_j$ similarity, i.e., satisfying the following equation:

$$\mathbf{Q}_{ij} = \frac{(1+\|\boldsymbol{h}^i - \boldsymbol{\mu}_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1+\|\boldsymbol{h}^i - \boldsymbol{\mu}_j\|^2/\alpha)^{-\frac{\alpha+1}{2}}}, \tag{9}$$

where α is the degree of freedom, assigned a value of 1. $\mathbf{Q}_{ij}$ can be considered as a soft assignment indicating the probability of assigning node i to cluster j.

To enforce its constraint consistency, we obtain the feature $\mathbf{Z}$ from the latent representation $\mathbf{H}^{(l)}$ through three fully connected (FC) layers. Consequently, employing Eq. (9) enables us to denote the node distributions of the original and potential space as $\mathbf{O}^{ij}$ and $\mathbf{P}^{ij}$, respectively. In this way, the final distributed consistency network loss is defined by the KL divergence:

$$\mathcal{L}_{dc} = \min_{\xi,\boldsymbol{\mu}}\sum_{m=1}^{M}\mathrm{KL}\left[\ g(\mathbf{X}_m, \boldsymbol{\mu}; \xi)\ ||\ g(\mathbf{Z}_m, \boldsymbol{\mu}; \xi)\ \right] \tag{10}$$

$$= \min_{\xi,\boldsymbol{\mu}}\sum_{m=1}^{M}\sum_{i}\sum_{j}\mathbf{O}_m^{ij}\log\frac{\mathbf{O}_m^{ij}}{\mathbf{P}_m^{ij}}, \tag{11}$$

where ξ denotes the trainable FC layer parameters.

Dropping Pathway Strategy: The potential problems (including over-smoothing) under deep network structures are unpredictable. Moreover, in this case, allowing the self-expressive layer to absorb the multi-level information from all pathways may result in invalid or redundant information. Thus, we design three dropping strategies from different perspectives.

1) DropEdge: GCN transmits the information via edges and making the edge sparser may help reduce its over-smoothing. Inspired by [22], we define the dropout rate as p to randomly select a fixed proportion of edge sets $\mathbf{V}_p$ from edges $\mathbf{V}$, and then Dropedge's formula can be written as

$$\mathbf{A}_{drop} = \mathbf{A} - \mathbf{A}', \tag{12}$$

where $\mathbf{A}'$ is an adjacency matrix composed of $\mathbf{V}_p$.

2) DropFeature: Although Dropout [26] has little effect on over-smoothing, it may be beneficial for other problems (such as over-fitting), especially under deep network structure. Therefore, we apply the Dropout operation at the feature level, such as $\mathbf{X}_m$, by deactivating some of the neurons for some generalization performance.

3) DropChannel: The multi-pathway network has the capacity to capture valuable multi-level information. However, it may also introduce some extraneous information that can lead to confusion. Thus, we propose a new dropping strategy called DropChannel. During the training process, it enforces the model to randomly drop a fixed proportion of all pathways while preserving the weighted

Algorithm 1 DPMGSC

Input:
Node attributes matrix $\{\mathbf{X}_m\}_{m=1}^{M}$,
Graph structure $\mathbf{A}$
Class number k
Initial:
Network parameters Θ, ξ
Self-expressive coefficient matrix $\mathbf{S}$
1: Perform K-means on $\mathbf{X}_1$
2: Initialize cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$
3: **while** not converge **do**
Train the entire networks and update parameters θ, $\mathbf{S}$, ξ and $\boldsymbol{\mu}$ by using Eq. (14)
end
4: Derive the self-expressive coefficient matrix $\mathbf{S}$ composed of network parameters
5: Obtain the affinity matrix $\mathbf{C} = \frac{1}{2}(|\mathbf{S}| + |\mathbf{S}|^T)$
6: Perform spectral clustering on matrix C

information from the previous iteration. Specifically, if the dropped channels form a set $\mathbf{Y}$ in sequence, the Eq. (8) becomes:

$$\mathcal{L}_{se} = \min_{\mathbf{S}} \sum_{l=1, l\notin\mathbf{Y}}^{L} \sum_{m=1}^{M} \boldsymbol{\beta}_l \|\mathbf{H}_m^{(l)} - \mathbf{S}\mathbf{H}_m^{(l)}\|_F^2 + \|\mathbf{S}\|_F \quad s.t.\ diag(\mathbf{S}) = 0,\ \sum_{l=1}^{L} \boldsymbol{\beta}_l = 1. \tag{13}$$

The final loss is given by combining the above losses:

$$\mathcal{L} = \min_{\theta, \mathbf{S}, \xi, \boldsymbol{\mu}} \mathcal{L}_{reX} + \lambda_1 \mathcal{L}_{reA} + \lambda_2 \mathcal{L}_{se} + \lambda_3 \mathcal{L}_{dc}, \tag{14}$$

where λ_1, λ_2, and λ_3 are trade-off parameters to balance the effects of graph structure loss, multi-pathway self-expressive loss, and distribution consistency loss, respectively.

Training Procedure: In our experiments, The k-means will be implemented on the node attribute $\mathbf{X}_1$ to get the initialized cluster centroids $\{\boldsymbol{\mu}_j\}_{j=1}^{k}$ and thus speed up the convergence. The entire network model will be trained by Eq. (14), and the network parameters ξ and θ will be updated continuously. During the training, the self-expressive coefficient matrix $\mathbf{S}$ is iteratively updated, and we obtain an affinity matrix $\mathbf{C}$ that is suitable for spectral clustering. Detailed algorithm flow can be seen in Algorithm 1.

4 EXPERIMENTS

This section describes the dataset, evaluation metrics, implementation details, and comparison methods for our experiments. We also conduct some follow-up experiments, *i.e.*, ablation analysis and parametric analysis, to show the advantages of DPMGSC. Lastly, we investigate its performance with different numbers of GCN layers, from 2 to 8, to examine the hazards of network deepening and the effectiveness of multi-pathway networks and dropping strategy.

4.1 Experimental Settings

Datasets and Metrics: We perform experiments on four classic graph datasets containing graph structure and node attributes, whose statistics information is summarized in Table 2. For all datasets, we choose the Fast Fourier Transform (FFT) of their node attributes as the second view. Meanwhile, we adopt three widely-used metrics as our performance evaluation basis, namely, clustering accuracy (ACC), normalized mutual information (NMI), and average rand index (ARI). Larger values imply better performance.

Table 2: Information of four datasets

Dataset	Nodes	Edges	Dimension	Classes
Cora [25]	2708	5429	1433	7
Citeseer [25]	3327	4723	3703	6
Wiki [36]	2405	8261	4973	17
DBLP [4]	4057	3528	334	4

Implementation Details: Similar to other mainstream multi-view graph clustering methods, a two-layer graph convolutional autoencoder is considered for comparison with other algorithms. Specifically, we set the layer dimension to [512, 512] for the Cora, Wiki, and DBLP datasets and set that to [2000, 500] for the Citeseer dataset. Meanwhile, to investigate the GCN with a deep network structure (i.e., 8-layer GCN), we use [512,512,512,512,512,512,512,512] as its dimension. We conduct all the experiments on a desktop with an NVIDIA 2080TI GPU and 64 GB memory and implement our network model using the TensorFlow toolbox with the ADAM optimizer [13]. Furthermore, the same dataset is used for comparing algorithms and conducting performance testing. In the case of single-view algorithms, we record the best results obtained from each view. For multi-view algorithms (e.g., SGCMC), the second view adopts the FFT form. Based on the corresponding paper settings of the algorithms, we conducted several experiments to obtain the best performance.

4.2 Comparison with Other Algorithms

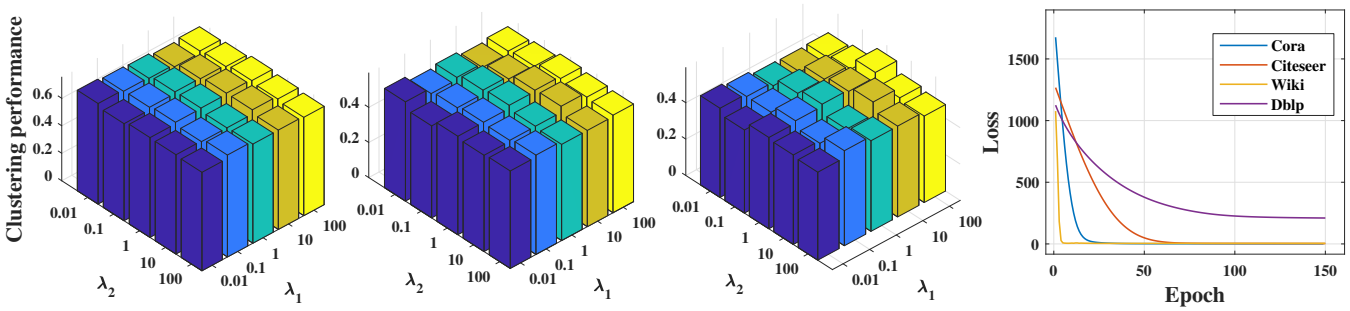
Existing Methods: We select different types of graph clustering algorithms as comparison targets, which can be roughly categorized into the following groups:

- Node attributes $\mathbf{X}$: K-means [19].
- Graph structure $\mathbf{A}$: Deep Walk [21].
- Both node attributes $\mathbf{X}$ and graph structure $\mathbf{A}$: GAE and VGAE [14], Adversarially Regularized Graph Autoencoder (ARGAE) and Adversarially Regularized Variational Graph Autoencoder (ARVGAE) [20], Deep Attentional Embedding Graph Clustering (DAEGC) [28], Graph Attention Auto-Encoders (GATE) [23], Structural deep clustering network (SDCN) [4], Embedding Graph Auto-Encoder for Graph Clustering (EGAE) [38].
- Node attributes $\mathbf{X_1}$ and $\mathbf{X_2}$: Deep canonical correlation analysis [1], Deep canonically correlated autoencoders [31].
- Node attributes $\mathbf{X_1}$, $\mathbf{X_2}$ and graph structure $\mathbf{A}$: CO-GCN [18], Self-supervised graph convolutional network for multi-view clustering SGCMC [35], Consistent Multiple Graph Embedding for Multi-View Clustering (CMGEC) [33].

Experimental Results: We record the experimental results of each algorithm on the four datasets in Table 3. Through the overall analysis and comparison, we have drawn the following conclusions: 1)

Table 3: Clustering results of various methods on four datasets. The best results are highlighted in Bold.

Models	Cora			Citeseer			Wiki			DBLP		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
K-means [19]	0.500	0.317	0.239	0.544	0.312	0.285	0.439	0.384	0.152	0.384	0.110	0.069
Deep Walk [21]	0.529	0.384	0.291	0.390	0.131	0.137	0.399	0.366	0.201	0.472	0.147	0.108
GAE [14]	0.530	0.397	0.293	0.380	0.174	0.141	0.332	0.320	0.151	0.556	0.304	0.185
VGAE [14]	0.592	0.408	0.347	0.392	0.163	0.101	0.493	0.461	0.299	0.532	0.293	0.168
ARGAE [20]	0.640	0.449	0.352	0.573	0.350	0.341	0.391	0.357	0.122	0.624	0.267	0.247
ARVGAE [20]	0.638	0.450	0.374	0.544	0.261	0.245	0.396	0.346	0.115	0.670	0.308	0.261
DAEGC [28]	0.704	0.528	0.496	0.672	0.397	0.410	0.521	0.432	0.337	0.625	0.329	0.216
GATE [23]	0.658	0.527	0.451	0.616	0.401	0.381	0.482	0.343	0.188	0.680	0.337	0.346
SDCN [4]	0.495	0.275	0.215	0.663	0.390	0.406	0.432	0.396	0.227	0.690	0.401	0.404
EGAE [38]	0.724	0.540	0.472	0.674	0.412	0.432	0.515	0.480	0.331	0.672	0.331	0.336
DCCA [1]	0.436	0.214	0.160	0.450	0.221	0.204	0.358	0.366	0.170	0.412	0.182	0.063
DCCAE [31]	0.472	0.289	0.221	0.503	0.240	0.211	0.321	0.343	0.156	0.437	0.232	0.103
COGCN [18]	0.735	0.567	0.512	0.655	0.432	0.423	0.515	0.450	0.330	0.648	0.341	0.305
SGCMC [35]	0.658	0.502	0.382	0.612	0.336	0.366	0.478	0.436	0.275	0.683	0.391	0.374
CMGEC [33]	0.737	0.504	0.438	**0.728**	0.403	0.443	0.557	0.496	0.316	0.699	0.386	0.392
Ours (2 layers)	**0.749**	**0.590**	**0.546**	0.696	**0.435**	**0.445**	**0.603**	**0.531**	**0.403**	**0.720**	**0.402**	**0.417**



(a) ACC **(b) NMI** **(c) ARI** **(d) Convergence**

Figure 2: (a)-(c): Parameter sensitivity of the λ_1 and λ_2 on Cora dataset with fixed $\lambda_3 = 0.01$; (d): Convergence analysis on four datasets.

Compared with other algorithms, DPMGSC achieves outstanding results on all datasets. Specifically, our method leads by a large margin on the wiki dataset, and it outperforms others by about 2% on the DBLP dataset in terms of ACC. 2) Compared with methods that only use node attributes or graph structures, GCN-based graph clustering methods can achieve better clustering performance, such as DAEGC and EGAE, indicating the superior representation ability of that GCN. However, these single-view methods are generally inferior to multi-view graph clustering methods. The probable reason lies in the lack of complementary information provided by different views. 3) Although Co-GCN and CMGEC can also learn consistent information in multiple views, their clustering performance is slightly inferior to ours. We think the single-pathway method misses the multi-level and comprehensive complementary information, which hinders the improvement of clustering performance.

Parameters and Convergence Analysis: DPMGSC employs three trade-off parameters to balance the effects of various losses. With a fixed parameter λ_3 of the distribution consistency loss, we plot the variation of the clustering performance on the Cora dataset against different values of the parameter λ_1 of the graph structure loss and the parameter λ_2 of the multi-pathway self-expressive loss, as shown in Figure 2. (a)-(c). By changing the parameters in the range of $\{0.01, 0.1, 1, 10, 100\}$, DPMGSC achieves the best clustering performance when $\lambda_1 = 10$, $\lambda_2 = 10$ and $\lambda_3 = 0.01$, indicating that appropriate parameter changes are beneficial to the improvement of clustering performance. Furthermore, the convergence curve on the four datasets is depicted in Figure 2. (d). The epoch number

Table 4: Ablation Study on four datasets. Best results are highlighted in Bold.

Models	Cora			Citeseer			Wiki			DBLP		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
Single-Pathway	0.743	0.567	0.503	0.687	0.419	0.433	0.586	0.509	0.391	0.703	0.381	0.379
Without L_{ReX}	0.724	0.523	0.483	0.673	0.396	0.417	0.582	0.501	0.375	0.712	0.382	0.402
Without L_{ReA}	0.708	0,515	0.466	0.683	0.424	0.434	0.571	0.492	0.369	0.668	0.359	0.366
Without L_{Se}	0.718	0.529	0.485	0.672	0.413	0.418	0.576	0.487	0.376	0.679	0.367	0.372
Without L_{Dc}	0.737	0.534	0.523	0.690	0.423	0.426	0.591	0.518	0.389	0.709	0.381	0.390
DPMGSC	**0.749**	**0.590**	**0.546**	**0.696**	**0.435**	**0.445**	**0.603**	**0.531**	**0.403**	**0.720**	**0.402**	**0.417**

Table 5: Clustering performance of the proposed DPMGSC model with an increasing view of HW dataset.

HW(k=5)	1 view	2 views	3 views	4 views	5 views	6 views
ACC	0.812	0.821	0.834	0.830	0.831	**0.838**
NMI	0.788	0.795	0.817	**0.840**	0.836	0.836
ARI	0.716	0.737	0.764	0.775	0.772	**0.778**

of each dataset is set to 150. During experiments, the loss values on the four data sets can drop rapidly and tend to a fixed value after dozens of epochs. It can be seen that DPMGSC can converge quickly.

Ablation Study: As shown in Table 4, a series of works is set up to verify the role of the components in the network model. On the four datasets, we test the performance of the model in the absence of reconstruction loss, self-expression layer learning loss, and distribution-consistent learning loss, respectively. In addition, the model's performance under a single pathway is additionally considered to verify the effectiveness of multiple pathways. It can be found through experiments that: 1) All loss functions have a positive effect on their clustering performance. When all of them exist, DPMGSC achieves optimal performance. 2) Constraints on graph structure reconstruction make a greater contribution to clustering performance, possibly because it is a GCN-based method that relies on graph structures. 3) The implementation of multi-pathway networks can obtain multi-level information and improve clustering accuracy, for example by improving ACC by 1% to 2% on the Wiki and DBLP datasets.

Evaluation on dataset with more than two views: It is appreciated to focus on the clustering performance of DPMGSC on datasets with more than two views, and we try to validate it on the Handwritten Numerals (HW) dataset [9]. It contains 2000 samples from 10 classes and has 6 views with feature dimensions of $\{216, 76, 64, 6, 240, 47\}$. HW dataset does not involve the graph structure, which hinders the graph representation learning of GCN. Thus, we build the graph structure $\mathbf{A}$ via KNN (k = 5) and evaluate the performance, as shown in the table 5. There is no significant performance improvement with increasing the number of views ($M >= 3$), and one speculation is that the noise interference and redundant information between the views hinder improving its performance.

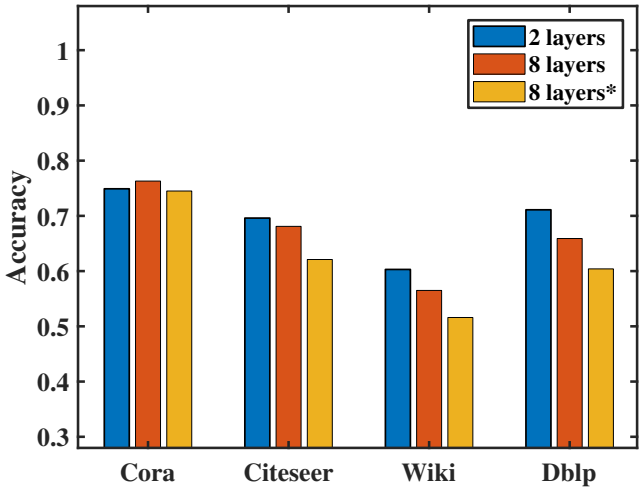


Figure 3: Comparison of the accuracy of each dataset in shallow and deep networks. * Represents that the strategy adopted is single-pathway.

4.3 Impact of Deep GCN on Representation Learning

To explore whether deep GCN is conducive to representation learning, we deepen the network to 8 layers based on the model DPMGSC, and the performance is shown in Figure 3. We have the following interesting observations: 1) With the increase in network depth, DPMGSC exhibits poor clustering performance on the Citeseer, Wiki, and DBLP datasets, indicating that the deep network structure has caused problems such as over-smoothing; 2) The performance of DPMGSC on the Cora dataset is relatively awesome. When combined with our multi-pathway network, it even outperforms the shallow network ; 3) A comparison between the results in the second and third columns reveals that the negative effect caused by over-smoothing is more severe when only a single pathway is adopted. However, this problem can be effectively alleviated when the multi-pathway approach is considered.

4.4 Discussion of dropping strategies

(1) DPMGSC with DropEdge: Due to the high computational complexity that layer-wise DropEdge may bring, we compromised

Table 6: Clustering results of the proposed DPMGSC model w and w/o different dropping strategies. * Represents that the strategy adopted is single-pathway. Best results for different cases are highlighted in Bold while sub-optimal results are underlined.

Dataset	Metric	2 layers		8 layers*		8 layers			
		Orignal	DropEdge	Orignal	DropEdge	Orignal	DropEdge	DropFeature	DropChannel
Cora	ACC	0.749	**0.752**	0.745	**0.751**	0.763	**0.776**	0.769	0.769
	NMI	**0.590**	0.584	**0.563**	**0.563**	0.594	**0.603**	0.590	0.597
	ARI	**0.546**	0.542	0.522	**0.543**	0.555	**0.591**	0.567	0.567
Citeseer	ACC	0.696	**0.699**	**0.621**	0.613	0.681	0.677	0.676	**0.683**
	NMI	0.435	**0.439**	**0.327**	0.320	0.407	**0.411**	0.404	0.409
	ARI	**0.445**	**0.445**	**0.329**	0.317	0.420	**0.424**	0.417	0.423
Wiki	ACC	0.603	**0.607**	0.516	**0.521**	0.565	**0.576**	0.571	0.575
	NMI	0.531	**0.536**	0.440	**0.458**	0.514	0.516	0.517	**0.521**
	ARI	0.403	**0.414**	0.267	**0.292**	0.363	**0.382**	0.346	0.375
DBLP	ACC	**0.720**	0.709	0.604	**0.611**	0.659	**0.668**	0.656	0.662
	NMI	**0.402**	0.385	0.270	**0.280**	0.318	**0.350**	0.341	0.331
	ARI	**0.417**	0.402	**0.258**	0.228	0.321	**0.354**	0.324	0.335

by only perturbing that all layers share the same adjacency matrix. The final comparison result is summarized in Table 6 (second column). It can be seen that our multi-pathway network combined with DropEdge can perform better clustering performance. When the network has an 8-layer GCN structure, the results on the four datasets are improved. For example, the ACC on the Cora and Wiki datasets has increased. Even in a two-layer network or a single pathway, the performance when introducing the DropEdge strategy is basically the same or slightly improved.

(2) DPMGSC with DropFeature: Strategies for layer-wise dropout and dropout on the original node attributes $\mathbf{X}_m$ are studied in 8-layer GCN separately, and the results are recorded in Table 6. Experimental results show the performance of adding dropout varies on different datasets. Besides, introducing the dropout strategy shows little impact on clustering performance and also varies on different datasets. Furthermore, when considering layer-by-layer dropout, the help to performance tends to be detrimental, which may be caused by over-generalization.

(3) DPMGSC with DropChannel: We tested the performance of DropChannel in the case of 8-layer GCN and recorded it in Table 6. As expected, after randomly dropping multiple pathways, the performance on the four datasets has improved. There is a reason to believe that the network model can obtain a more affinity shared coefficient matrix at each iteration through adaptive learning of each channel, thereby achieving better clustering results.

To sum up, the three strategies improve clustering performance from different perspectives. It can be observed that the most significant issue encountered under the deep network structure is likely over-smoothing, which can be effectively mitigated by employing the DropEdge strategy. Notably, DropEdge outperforms the other two dropping strategies. The representation learning capability of the multi-pathway method is further enhanced with the incorporation of DropChannel. Introducing the concept of randomly dropping specific channels and enabling the network to adaptively learn yields promising results. However, when we try to apply layer-by-layer dropout on the features, there was no significant improvement in clustering performance. One conjecture is that the layer-by-layer dropout may lead to a more severe loss of the propagated feature information.

5 CONCLUSION

This paper has proposed a novel multi-view graph subspace clustering method entitled as DPMGSC, which utilizes multi-pathway to capture complementary and multi-level information. Besides, we have explored its performance under deep network structures and alleviated the over-smoothing issue in deep GCNs by developing a dropping pathway (DropPathway) strategy. Extensive experimental results have demonstrated that the proposed multi-pathway network can well retain the complementary information and thus improve the clustering performance. Plus, we have empirically tested the DropPathway strategy by adding network depth, exhibiting its good potential to mitigate the negative impact of the over-smoothing issue in deep graph neural networks, especially for the clustering problem. In the future, we will further investigate the intrinsic reason of the over-smoothing issue and seek a more effective solution.

6 ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under Grant 62176203, the Open Projects Program of State Key Laboratory of Multimodal Artificial Intelligence Systems, the Fundamental Research Funds for the Central Universities, the Natural Science Basic Research Program of Shaanxi Province (Grant 2023-JC-YB-534), and the Science and technology project of Xi'an (Grant 2022JH-JSYF-0009).

REFERENCES

[1] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *ICML*. PMLR, 1247–1255.

[2] Vladimir Batagelj. 2003. Efficient Algorithms for Citation Network Analysis. *CoRR* cs.DL/0309023 (2003).

[3] Mikhail Belkin and Partha Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS*. MIT Press, 585–591.

[4] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural deep clustering network. In *Web Conference*. 1400–1410.

[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *ACM CIKM*. 891–900.

[6] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*. PMLR, 1725–1735.

[7] Jiafeng Cheng, Qianqian Wang, Zhiqiang Tao, Deyan Xie, and Quanxue Gao. 2021. Multi-view attribute graph convolution networks for clustering. In *IJCAI*. 2973–2979.

[8] Peter Csermely, Tamás Korcsmáros, Huba JM Kiss, Gábor London, and Ruth Nussinov. 2013. Structure and dynamics of molecular networks: a novel paradigm of drug discovery: a comprehensive review. *Pharmacology & therapeutics* 138, 3 (2013), 333–408.

[9] Robert Duin. [n. d.]. Multiple Features. UCI Machine Learning Repository.

[10] Claudio Gallicchio and Alessio Micheli. 2020. Fast and Deep Graph Neural Networks. In *AAAI*. AAAI Press, 3898–3905.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE CVPR*. 770–778.

[12] Ruiqi Hu, Shirui Pan, Guodong Long, Xingquan Zhu, Jing Jiang, and Chengqi Zhang. 2016. Co-clustering enterprise social networks. In *IJCNN*. IEEE, 107–114.

[13] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[14] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*. OpenReview.net.

[16] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *IEEE ICCV*. 9267–9276.

[17] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*. AAAI Press, 3538–3545.

[18] Shu Li, Wen-Tao Li, and Wei Wang. 2020. Co-gcn for multi-view semi-supervised learning. In *AAAI*, Vol. 34. 4691–4698.

[19] James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Mathematical Statistics and Probability*. 281–297.

[20] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI*. ijcai.org, 2609–2615.

[21] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *ACM SIGKDD*. 701–710.

[22] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

[23] Amin Salehi and Hasan Davulcu. 2020. Graph Attention Auto-Encoders. In *IEEE ICTAI*. IEEE, 989–996.

[24] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE TNN* 20, 1 (2008), 61–80.

[25] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[27] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*. OpenReview.net.

[28] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *IJCAI*. ijcai.org, 3670–3676.

[29] Qianqian Wang, Zhengming Ding, Zhiqiang Tao, Quanxue Gao, and Yun Fu. 2018. Partial multi-view clustering via consistent GAN. In *IEEE ICDM*. 1290–1295.

[30] Qianqian Wang, Zhengming Ding, Zhiqiang Tao, Quanxue Gao, and Yun Fu. 2021. Generative partial multi-view clustering with adaptive fusion and cycle consistency. *IEEE TIP* 30 (2021), 1771–1783.

[31] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. 2015. On deep multi-view representation learning. In *ICML*. PMLR, 1083–1092.

[32] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW*. ACM, 2022–2032.

[33] Yiming Wang, Dongxia Chang, Zhiqiang Fu, and Yao Zhao. 2023. Consistent Multiple Graph Embedding for Multi-View Clustering. *IEEE TMM* 25 (2023), 1008–1018.

[34] Bo Wu, Yang Liu, Bo Lang, and Lei Huang. 2018. DGCNN: Disordered graph convolutional neural network based on the Gaussian mixture model. *Neurocomputing* 321 (2018), 346–356.

[35] Wei Xia, Qianqian Wang, Quanxue Gao, Xiangdong Zhang, and Xinbo Gao. 2021. Self-supervised graph convolutional network for multi-view clustering. *IEEE TMM* 24 (2021), 3182–3192.

[36] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information.. In *IJCAI*, Vol. 2015. 2111–2117.

[37] Changqing Zhang, Qinghua Hu, Huazhu Fu, Pengfei Zhu, and Xiaochun Cao. 2017. Latent multi-view subspace clustering. In *IEEE CVPR*. 4279–4287.

[38] Hongyuan Zhang, Pei Li, Rui Zhang, and Xuelong Li. 2022. Embedding graph auto-encoder for graph clustering. *IEEE TNNLS* (2022), 1–11.

[39] Ronghang Zhu, Zhiqiang Tao, Yaliang Li, and Sheng Li. 2021. Automated graph learning via population based self-tuning GCN. In *ACM SIGIR*. 2096–2100.