Dual Fusion-Propagation Graph Neural Network for Multi-View Clustering

Shunxin Xiao, Shide Du, Zhaoliang Chen, Yunhe Zhang and Shiping Wang

***Abstract*—Deep multi-view representation learning focuses on training a unified low-dimensional representation for data with multiple sources or modalities. With the rapidly growing attention of graph neural networks, more and more researchers have introduced various graph models into multi-view learning. Although considerable achievements have been made, most existing methods usually propagate information in a single view and fuse multiple information only from the perspective of attributes or relationships. To solve the aforementioned problems, we propose an efficient model termed Dual Fusion-Propagation Graph Neural Network (DFP-GNN) and apply it to deep multi-view clustering tasks. The proposed method is designed with three sub-modules and has the following merits: a) The proposed view-specific and cross-view propagation modules can capture the consistency and complementarity information among multiple views. b) The designed fusion module performs multi-view information fusion with the attributes of nodes and the relationships among them simultaneously. Experiments on popular databases show that DFP-GNN achieves significant results compared with several state-of-the-art algorithms.**

***Index Terms*—Deep learning, graph neural network, unsupervised learning, multi-view clustering.**

I. Introduction

CLUSTERING analysis is a crucial task in the artificial intelligence era and has been successfully applied to numerous domains, including analyses on computer vision [1], documental texts [2] and social networks [3]. Most classical clustering methods are designed to deal with single attribute data that it is difficult to capture the characteristics of targets sufficiently. With the rapid growth of multimedia technology, real-world objects are represented in multiple manners, such as texts, images and voices. Taking object detection as an example, an object instance can be photoed from various rotation angles, thereby generating the multi-view data that depict the target with multiple descriptions [4]. Compared with single-view form, multi-view data can be utilized to train a more comprehensive representation and achieve desirable results for downstream tasks. Thus, it is highly expected to use multiple information to refine the clustering results.

Multi-view clustering algorithms aim to capture the consistency and complementarity among different views and employ them to group samples in an unsupervised way. Recently, numerous approaches have been proposed and can be roughly divided into graph-based approaches [5], [6], [7], subspace learning [8], [9], [10], matrix factorization [11], [12], [13], and deep neural networks [14], [15], [16]. Specifically, graph-based methods first build the relationship among nodes and then use spectral clustering to obtain the cluster assignments. The goal of subspace methods is to reconstruct the original indistinguishable space into an ideal subspace. In addition, considerable researchers pointed out that matrix factorization technology helps reduce redundant features and keep the most important attributes. Finally, deep model-based methods are usually used to learn a low-dimensional embedding. The core idea of these algorithms is to extract multiple information from different sources to learn a unified representation and they have been successfully applied in several domains. However, they usually either utilize the relationships among nodes or attributes of samples in the learning process. Consequently, these two important characteristics may not be made full use to refine the results of clustering simultaneously.

Graph Neural Network (GNN) has been proposed to deal with non-Euclidean data and has shown significant efficiency and powerful learning capability on various machine learning tasks [17], [18], [19], [20], [21]. As the most critical characteristic of GNN, message passing mechanism can aggregate information of neighborhoods through their relationships to learn a meaningful representation for each node. As a result, the learned node embedding captures the content information of samples and keeps the topological structure among nodes. Therefore, several works introduce GNN for multi-view learning to solve the aforementioned problem [22], [23], [24]. Although these GNN-based algorithms have achieved significant performance on many multi-view learning tasks, most of them suffer from the following limitations. In the perspective of message passing, on the one hand, they usually only conduct information propagation in view-specific input, but not in the fused representation obtained from all views [25], [26]. Therefore, the learned unified embedding may not sufficiently capture the consistency among heterogeneous views. On the other hand, in the perspective of information fusion, they either conduct fusion progress from the feature matrix level or the adjacency matrix level [27], [28]. Thus, it is difficult to adaptively fuse the content information and topological structure for representation learning simultaneously.

To tackle the discussed problems, we propose a graph convolution-based unsupervised learning framework termed Dual Fusion-Propagation Graph Neural Network (DFP-GNN) for multi-view clustering tasks. The overview of the pro-

This work is in part supported by the National Natural Science Foundation of China under Grant U21A20472 and 62276065, and the National Key Research and Development Plan of China under Grant 2021YFB3600503.

S. Xiao, S. Du, Z. Chen, Y. Zhang and S. Wang are with the College of Computer and Data Science, Fujian Provincial Key Laboratory of Network Computing and Intelligent Information Processing, Fuzhou University, Fuzhou 350116, China. E-mail: xiaoshunxin.tj@gmail.com, dushidems@gmail.com, chenzl23@outlook.com, zhangyhannie@163.com, shipingwangphd@163.com.

(Corresponding author: Shiping Wang.)

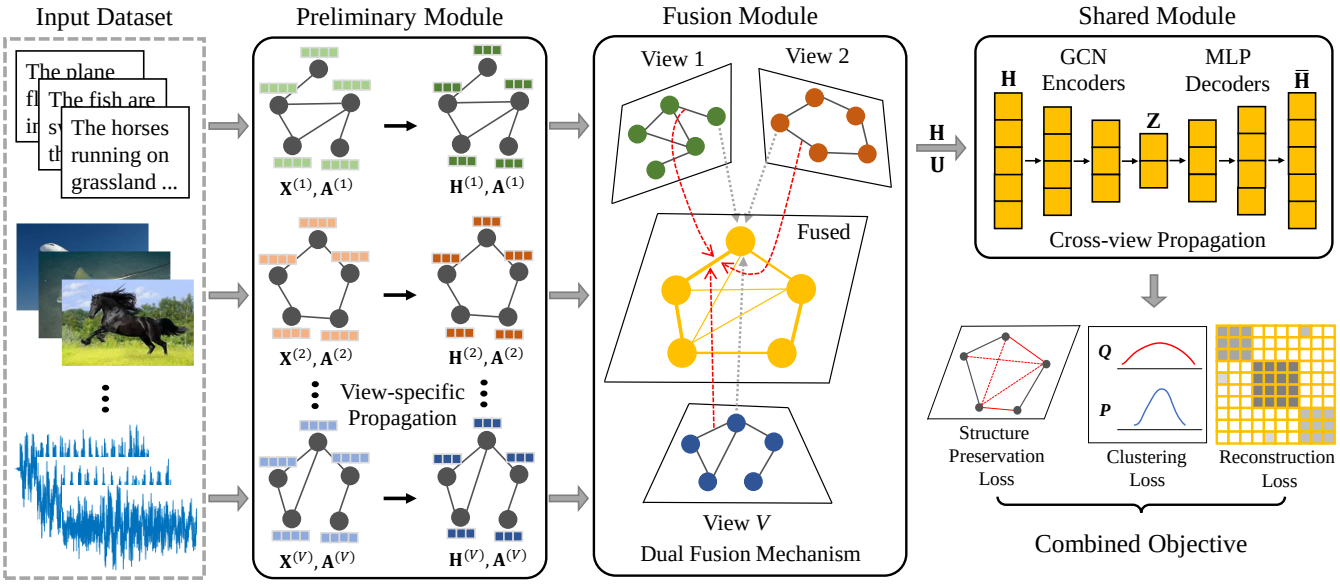


Fig. 1. The overview of the proposed DFP-GNN framework. Given a dataset with multiple sources, several view-specific graph convolution networks are used to generate a hidden representation for each view in the preliminary module. Based on these hidden features, an adaptive fusion mechanism is utilized for both attribute-level and structure-level fusion. Then, the fused matrices are exploited as the input of the shared module to apply information propagation across multifarious views. Finally, the whole framework is optimized by a combined objective.

posed method is shown in Fig. 1. Specifically, DFP-GNN contains three submodules, including the preliminary module, the fusion module, and the shared module, to capture the consistency and complementarity among multiple views. The preliminary part contains multiple encoders that each encoder is implemented by a multi-layer graph convolution network and employed to produce an initial embedding for each view. Then, the fusion module introduces an adaptive learning method to fuse the generated initial representations and topological relationships obtained from different views. Finally, the shared module is designed to learn a shared hidden representation by optimizing a combined loss. The training process of DFP-GNN can be divided into two stages: a) The proposed three submodules are trained separately where the output of a former module is used as the input of the next module, in the pretraining stage. b) In the finetuning stage, the whole DFP-GNN framework is optimized under the aforementioned combined objective.

To be specific, comprehensive experiments are designed to prove the effectiveness of DFP-GNN. The contributions of our work could be summed up as follows:

- We propose an efficient learning framework named Dual Fusion-Propagation Graph Neural Network (DFP-GNN) and apply it to multi-view clustering tasks.
- The dual fusion module and dual information propagation mechanism are designed to capture the multiple information among different views.
- Experimental results show that DFP-GNN achieves significant improvements against existing multi-view learning algorithms on ten benchmark databases.

The rest of this paper is organized as follows. In Section II, we introduce the related literature, including multi-view clustering algorithms and graph neural networks. The framework and training process of our method are presented in Section III. In Section IV, we evaluate and analyze the effectiveness of DFP-GNN on a variety of benchmark datasets. Finally, conclusions and possible directions are drawn in Section V.

II. Related Work

We first introduce several multi-view clustering methods based on traditional machine learning and deep learning. Furthermore, various graph neural networks used for single-view and multi-view representation learning are introduced.

A. Multi-view clustering

Different from single-view clustering, multi-view methods are designed to capture multiple information among different views and obtain better cluster assignments [29], [30], [31], [32]. Xia et al. [33] designed a spectral clustering method to construct a shared matrix for the Markov chain to solve the possible noises appearing in multi-view data. To release the requirement of parameters, an auto-weighted multiple graph learning framework was designed under the assumption that there exist consistency and complementarity information among multiple views [30]. To learn from data with large-scale samples, Zhang et al. [13] encoded the multiple input raws into a shared binary latent representation and then utilized a binary matrix factorization to generate the cluster assignments. In addition, Wang et al. [5] proposed a graph-based algorithm to adaptively balance the importance among different views by fusing all graph matrices into a unified matrix to obtain the cluster assignments. To efficiently capture complementary information, Tang et al. [34] proposed obtaining cluster assignments based on a joint affinity graph. For incomplete multi-view learning, Liu et al. [35] designed a late fusion strategy to solve the high computational complexity problem.

In recent years, more and more researchers have introduced deep neural networks to multi-view clustering by learning a shared low-dimensional representation. As a nonlinear extension of canonical correlation analysis, the deep version was designed to capture the nonlinear relationship between two input raws to learn highly linearly correlated features [36]. To embed the multiple information into a shared intact representation for multi-view clustering, Zhang et al. [14] proposed a deep learning model named autoencoder in autoencoder networks by joint optimizing view-specific feature learning and multi-view information encoding. Furthermore, Huang et al. [15] designed a deep learning version of multi-view spectral clustering with a novel objective function to capture the view-specific local invariance and consistency among various views. Xie et al. [31] incorporated feature learning, view fusion and clustering into a unified framework for multi-view clustering. For incomplete multi-view learning, Lin et al. [37] designed a novel optimization object to learn a consistent embedding from multiple views and recover the missing views simultaneously. To reduce the impact of incomplete view, Yang et al. [38] proposed a unified model with a contrast learning paradigm by utilizing available nodes as positive samples and several cross-view patterns as the negative part. Besides, Peng et al. [39] projected the input multi-view data onto an embedding space where the constructed graph was used to generate the final clustering results. Different from [39], the proposed DFP-GNN framework uses the adjacency graph to perform information propagation instead of generating cluster assignments.

The difference between the DFP-GNN and the aforementioned methods is that our method has the ability to capture the topological structure of the graph and attributes of samples in the training process. Consequently, the learned embedding can be more discriminative rather than those learned by traditional clustering approaches.

B. Graph neural networks

Aa a non-Euclidean extension of traditional deep learning paradigms (e.g., convolution and pooling), various GNNs have been designed to deal with different learning tasks based on graph or manifold data [40], [41], [42].

In single-view clustering, for example, Wang et al. [43] transferred the face clustering problem as a link prediction task and then applied graph convolution network (GCN) to predict the probability of linkage between two samples, where there exists an edge if they belong to the same cluster. To mutually benefit from both graph embedding learning and clustering processing, a deep attentional embedded learning framework optimized in a self-training manner was proposed for graph clustering [44]. Furthermore, Bianchi et al. [45] combined spectral clustering with GNNs to release the requirement of spectral decomposition and extend for unseen samples by formulating a continuous relaxation.

Similarly, several works introduce graph learning models for multi-view learning tasks. In the task of urban region embedding, Zhang et al. [23] designed a graph attention mechanism-based joint training framework to learn informative region embeddings from urban data to assist urban planning. Cheng et al. [25] introduced a novel learning method for multi-view attribute graph clustering based on graph convolution operation. This work first utilized multiple graph attention networks for each view to learn an embedding feature and then designed consistent embedding encoders to find a shared clustering embedding. To improve the capacity of capturing multiple information, a graph autoencoder clustering framework was designed to learn feature representations by utilizing the most informative graph structure and node attributes to reconstruct all input views [27]. Furthermore, Li et al. [46] combined graph convolution network and co-training into a unified framework. In this method, the spectral information across different input views can be adaptively learned with the combined Laplacians.

Although the aforementioned GNN-based algorithms have achieved comparable performance on various multi-source data analysis tasks, there exist two differences between our method and them. On the one hand, most of them only propagate information in view-specific input raws. In contrast, DFP-GNN applies message passing on both all view-specific features and the cross-view representation. On the other hand, these approaches usually only employ the node contents among different views to fuse the multiple information. However, our method is able to sufficiently utilize multi-view information with an adaptively dual fusion module based on both node relationships and attributes.

III. Proposed Methodology

We introduce the detail of the proposed dual fusion-propagation graph neural network in this section. DFP-GNN contains three submodules: the preliminary module used for view-specific information propagation, the fusion module implemented by a dual-fusion mechanism and the shared module applying cross-view message passing on the fused inputs. The learning process of DFP-GNN consists of two stages: a) Module-wise pretraining. b) Finetuning the whole framework. Finally, the complexity analysis of DFP-GNN is demonstrated.

A. Multi-view clustering meets GNN

The goal of this work is to capture the consistency and complementarity information among multiple input data $\mathcal{X} = \left\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(V)}\right\}$ to improve the performance of clustering. Here, $\mathbf{X}^{(v)} \in \mathbb{R}^{n \times d_v}, v = 1, 2, \cdots, V$ represents the original features of the v-th view, n denotes the number of samples, V is the number of multiple views and d_v stands for the dimension of the v-th input feature space.

Besides, the message passing mechanism of GNN needs an adjacent graph to apply information propagation. However, existing multi-view datasets usually only provide the multiple input feature matrices and lack the relationships among samples. In order to introduce GNN for common multi-view clustering, it is necessary to simulate the real-world relationships between samples. To achieve this goal, a graph construction method is utilized to simulate adjacent graph $\mathcal{G}^{(v)} = (\mathcal{V}, \mathcal{E}^{(v)})$ for each input view $\mathbf{X}^{(v)}$ with $\mathcal{V}$ and $\mathcal{E}^{(v)}$ being the node set and edge set of the v-th view, respectively. In detail, we utilize an unsupervised method to build the initial

graph $\mathcal{G}^{(v)}$ by using Euclidean distance to find the nearest neighbors of each node to generate edge set $\mathcal{E}^{(v)}$. Then, two pruning strategies are used to improve the quality of the graph.

Based on the built graph $\mathcal{G}$, the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ can be generated under the rule $\mathbf{A}_{ij} = e_{ij}$, where $e_{ij} = 1$ denotes that there exists an edge between v_i and v_j in $\mathcal{G}$. Then, a renormalization trick form is utilized to alleviate the gradient vanishing/exploding problem occurred in the deep network. Accordingly, the refined form of $\mathbf{A}$ can be formulated as:

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}, \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n \in \mathbb{R}^{n\times n}$ is the self-connection variant of $\mathbf{A}$, $\mathbf{I}_n \in \mathbb{R}^{n\times n}$ denotes the identity matrix with n dimensions and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij} \in \mathbb{R}^{n\times n}$ is the degree matrix derived from the self-connection matrix $\tilde{\mathbf{A}}$. Thus, the final transition matrix $\hat{\mathbf{A}}$ can be calculated in the preprocessing stage.

B. Framework of DFP-GNN

The proposed DFP-GNN consists of three submodules: preliminary view-specific propagation module, dual fusion module and shared cross-view propagation module. In general, the preliminary module maps multi-view input features $\mathbf{X}^{(v)}$ with the simulated relationships into a low-dimensional initial embedding $\mathbf{H}^{(v)}$. Then, all learned initial representations and built relationships are fused simultaneously to generate the unified forms $\mathbf{H}$ and $\mathbf{U}$ in the fusion module. Finally, both the fused matrices are used as the inputs of the shared module and the whole framework is optimized under a combined objective.

Preliminary Module. Existing multi-view datasets are often represented from preprocessed features, which usually have a redundant feature space and are impacted by possible noisy samples. If the fusion process is carried out in this feature space, the learning algorithm can be affected by these redundant features, thereby failing to focus on capturing the complementarity among multiple views. Furthermore, multiple views usually have different input dimensions, which leads to difficulty in optimizing the downstream module. To solve these issues, we utilize a multi-layer graph convolution network for each input view to learn a low-dimensional and discriminative latent embedding space.

In this work, we denote the view-specific propagation network of the v-th view as $f_{pm}^{(v)}(\mathbf{X}^{(v)}, \hat{\mathbf{A}}^{(v)}; \Theta^{(v)})$, where $\Theta^{(v)} = \left\{\mathbf{W}^{(v,l)}, \mathbf{b}^{(v,l)}\right\}_{l=1}^{L}$ and L are the learnable parameters and number of layers of the view-specific network, respectively. In detail, $\hat{\mathbf{A}}^{(v)}$ is the transition matrix of the v-th view and can be obtained according to Eq. (1). Then, the computation of the l-th layer of $f_{pm}^{(v)}$ is formulated as follows:

$$\mathbf{H}^{(v,l)} = \sigma(\hat{\mathbf{A}}^{(v)}\mathbf{H}^{(v,l-1)}\mathbf{W}^{(v,l)} + \mathbf{b}^{(v,l)}). \tag{2}$$

Here, $\mathbf{W}^{(v,l)} \in \mathbb{R}^{d_{(v,l-1)}\times d_{(v,l)}}$ and $\mathbf{b}^{(v,l)} \in \mathbb{R}^{d_{(v,l)}}$ represent the weights and bias of the l-th layer, respectively. Then, $\mathbf{H}^{(v,0)} = \mathbf{X}^{(v)}$ is the initial input, $\mathbf{H}^{(v,l)} \in \mathbb{R}^{n\times d_{(v,l)}}$ denotes the output of the l-th layer, and $\mathbf{H}^{(v)} = \mathbf{H}^{(v,L)}$ represents the final output. Besides, σ denotes the non-linear activation function and is used for the first $L-1$ layers. As a result, the learned initial latent representations have the same dimension and keep the local context in the feature space.

Fusion Module. Fusing multiple information to capture the consistency and complementarity among various views is crucial for multi-view learning tasks. For those methods based on graph neural models, one can implement the fusion process through the node attributes or the topological structure. To our knowledge, only a few works carry out the fusion process from both attribute and structure information.

Therefore, we propose a dual fusion mechanism to adaptively fuse multiple node attribute information and multiple graph structure information simultaneously. Based on the learned view-specific latent embeddings, the fusion of feature-level can be formulated as:

$$\mathbf{H} = \sum_{v=1}^{V} \alpha_v \mathbf{H}^{(v)}. \tag{3}$$

Here, α_v is a learnable variable employed to adjust the importance of v-th preliminary embedding automatically and is initialized as $\alpha_v = 1/V$. Furthermore, α_v is normalized as:

$$\alpha_v = \frac{e^{\alpha_v}}{\sum_{i=1}^{V} e^{\alpha_i}}. \tag{4}$$

Similarly, the structure-level fusion can be computed to generate a unified transition matrix:

$$\mathbf{U} = \sum_{v=1}^{V} \beta_v \hat{\mathbf{A}}^{(v)}, \tag{5}$$

where β_v is a learnable weight for the v-th propagation matrix and is also normalized in a way similar to α_v. Benefiting from the dual fusion mechanism, the proposed method can flexibly fuse multiple information from all input views.

Shared Module. We further design a graph convolution-based shared module, which utilizes the fused representation and transition matrix as inputs, to train a comprehensive and compact embedding based on the cross-view propagation mechanism. The main architecture of the shared module is implemented by a stacked graph autoencoder (SGAE). Furthermore, a dot product decoder and a clustering layer are introduced to improve the robustness of DFP-GNN. The detailed information of the shared module is demonstrated in the following.

To be simplicity, the stacked graph autoencoder used in the shared module is denoted as $f_{sm}(\mathbf{H}, \mathbf{U}; \Omega)$, where $\Omega = \left\{\mathbf{W}^{(m)}, \mathbf{b}^{(m)}\right\}_{m=1}^{M}$ and M are the network parameters and the number of layers, respectively. On the one hand, the first $\frac{M}{2}$ layers are implemented by graph convolution encoders and the computation of the i-th encoder is

$$\begin{aligned} &\mathbf{Z}^{(i)} = \sigma\left(\mathbf{U}\mathbf{Z}^{(i-1)}\mathbf{W}^{(i)} + \mathbf{b}^{(i)}\right), \\ &i = 1, 2, \cdots, \tfrac{M}{2}, \end{aligned} \tag{6}$$

where $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ represent the learnable weights and bias, respectively. $\mathbf{Z}^{(0)} = \mathbf{H}$ is the initial input of encoder part and $\mathbf{Z} = \mathbf{Z}^{(\frac{M}{2})}$ is the final latent embeding. On the other hand, the last $\frac{M}{2}$ layers decode the learned representation $\mathbf{Z}$ to reconstruct the initial input $\mathbf{H}$ and the output of the j-th layer can be computed as:

$$\begin{aligned} &\overline{\mathbf{Z}}^{(j)} = \sigma\left(\overline{\mathbf{Z}}^{(j-1)}\mathbf{W}^{(j)} + \mathbf{b}^{(j)}\right), \\ &j = \tfrac{M}{2} + 1, \cdots, M. \end{aligned} \tag{7}$$

Here, $\overline{\mathbf{Z}}^{(0)} = \mathbf{Z}^{(\frac{M}{2})}$ is the input of decoder part and $\overline{\mathbf{H}} = \mathbf{Z}^{(M)}$ is the reconstruction form of $\mathbf{H}$. Besides, $\mathbf{W}^{(j)}$ and $\mathbf{b}^{(j)}$ are the network parameters and will be updated during the training process. Instead of using graph convolution, the decoder is implemented by a multi-layer perceptron to avoid the over-smoothing problem occurred in deep GCN models [47], [48]. Hence, the intrinsic information among different views would be captured by minimizing the difference between $\mathbf{H}$ and $\overline{\mathbf{H}}$ and more details will be introduced in the next subsection.

To alleviate the distortion degree of input space caused by deep network training, we introduce a dot product decoder to predict the relationships among samples in the learned space. In detail, we utilize the hidden embedding $\mathbf{Z}$ derived from Eq. (6) to reconstruct the graph by calculating the edge prediction probability. The computation is formulated as follows:

$$p(\overline{\mathbf{U}} \mid \mathbf{Z}) = \prod_{i=1}^{n} \prod_{j=1}^{n} p\left(\overline{\mathbf{U}}_{ij} \mid \mathbf{Z}_i, \mathbf{Z}_j\right), \tag{8}$$

where $p\left(\overline{\mathbf{U}}_{ij} = 1 \mid \mathbf{Z}_i, \mathbf{Z}_j\right) = \sigma\left(\mathbf{Z}_i \mathbf{Z}_j^T\right)$ represents the probability that there is an edge between two nodes and σ is implemented by a sigmoid function.

Furthermore, existing multi-view clustering algorithms usually first learn an embedding and apply k-means to obtain the final cluster assignments. However, such two-stage learning fails to make use of the clustering process to refine the proposed algorithm. Therefore, we introduce a clustering layer into the shared module in an unsupervised self-training manner. Specifically, the Student's t-distribution is adopted to generate soft assignments $\mathbf{Q}$ for all nodes based on the learned embedding $\mathbf{Z}$. Formally, the element q_{ij} of $\mathbf{Q}$ is computed as:

$$q_{ij} = \frac{\left(1 + \left\|\mathbf{Z}_i - \mu_j\right\|^2 / \gamma\right)^{-\frac{\gamma+1}{2}}}{\sum_{j'} \left(1 + \left\|\mathbf{Z}_i - \mu_{j'}\right\|^2 / \gamma\right)^{-\frac{\gamma+1}{2}}}, \tag{9}$$

where γ represents the degrees of freedom of the Student's t-distribution with $\{\mu_j\}_{j=1}^{k}$ being the cluster centroids initialized in the pretraining stage and learned during the finetuning stage. Accordingly, q_{ij} denotes the similarity between $\mathbf{Z}_i$ and μ_j and can be regarded as the label probability about assigning node v_i into cluster j. Then, a target distribution $\mathbf{P}$ is derived from $\mathbf{Q}$ and then used as supervised information to guide the training process. The computation of p_{ij} is shown as follows:

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

where $f_j = \sum_i q_{ij}$ is soft cluster frequency and employed to release the distortion appearing in the latent space. Finally, the clustering assignment c_i of node v_i can be formulated as:

$$\mathcal{C}_i = \arg\max_j q_{ij}. \tag{11}$$

Benefiting from the clustering layer, the proposed framework has the ability to learn a separate latent embedding space under the guidance of the clustering process. Furthermore, the highly confident pseudo-labels generated in the self-training process would make graph convolution more efficient in a self-supervised learning manner.

Overall, the proposed DFP-GNN can capture the consistency and complementarity among different views due to the view-specific propagation in the preliminary module, dual fusion mechanism in the fusion module and cross-view propagation in the shared module.

C. Training of DFP-GNN

In the pretraining stage, each view-specific network of the preliminary module is trained individually and then the fused matrices will be adopted to pretrain the share module. Finally, the pretrained weights of DFP-GNN will be finetuned by optimizing the proposed combined loss.

Pretraining Stage. In the preliminary module, specifically, each view-specific network will be extended to an autoencoder form to learn a low-dimensional embedding $\mathbf{H}^{(v)}$ in an unsupervised learning manner. The optimization objective contains two parts: a) Optimizing a mean square error between input feature $\mathbf{X}^{(v)}$ and the reconstruction form $\overline{\mathbf{X}}^{(v)}$. b) Optimizing a structure preservation loss between the predicted relationships $\overline{\mathbf{A}}^{(v)}$ and the target information $\tilde{\mathbf{A}}^{(v)}$.

Then, all learned initial embeddings $\left\{\mathbf{H}^{(v)}\right\}_{v=1}^{V}$ are fused to form a unified feature matrix $\mathbf{H}$ for the next module according to $\mathbf{H} = \frac{1}{V}\sum_v \mathbf{H}^{(v)}$ and all constructed transition matrices $\hat{\mathbf{A}}^{(v)}, v = 1, \cdots, V$ are used to form a fused transition matrix by computing $\mathbf{U} = \frac{1}{V}\sum_v \hat{\mathbf{A}}^{(v)}$. It is worth noting that a graph will be constructed from the fused feature $\mathbf{H}$ and a refined adjacency matrix $\tilde{\mathbf{U}}$ with a self-connection is then generated from the graph. In detail, $\tilde{\mathbf{U}}$ is employed as the supervised information to guide both the pretraining of the shared module and the finetuning of the whole framework.

The pretraining process of the shared module is optimized with two objectives. The first one is minimizing a mean square error between the fused representation $\mathbf{H}$ and the reconstructed feature $\overline{\mathbf{H}}$ derived from the shared module. The last one is the optimization of structure preservation loss between $\tilde{\mathbf{U}}$ and the predicted relationships $\overline{\mathbf{U}}$, which is computed by Eq. (8). After sufficient module-wise pretraining, the learned parameters are copied to the corresponding parts of DFP-GNN.

Finetuning Stage. The whole framework is trained to finetune the pretrained parameters under the optimization of a combined objective, which contains reconstruction loss, structure preservation loss and clustering loss. In detail, the reconstruction loss is implemented by a mean square error to optimize the difference between $\mathbf{H}$ and $\overline{\mathbf{H}}$. The optimization objective of reconstruction loss is formulated as follows:

$$\mathcal{L}_r(\mathbf{H}, \overline{\mathbf{H}}) = \frac{1}{n d_h} \left\|\mathbf{H} - \overline{\mathbf{H}}\right\|_F^2, \tag{12}$$

where d_h denotes the dimension of the fused embedding. Accordingly, the final intact representation $\mathbf{Z}$ of all views can be learned by optimizing (12).

Inspired by link prediction tasks, we propose a structure preservation loss for the dot product decoder by minimizing a binary cross-entropy and the computation is shown as follows:

$$\mathcal{L}_s = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\left[\tilde{\mathbf{U}}_{ij}\log\overline{\mathbf{U}}_{ij} + (1 - \tilde{\mathbf{U}}_{ij})\log\left(1 - \overline{\mathbf{U}}_{ij}\right)\right], \tag{13}$$

Algorithm 1 Dual Fusion-Propagation Graph Neural Network (DFP-GNN) for Multi-View Clustering

Input: Input features $\mathcal{X} = \{\mathbf{X}^{(i)}, \cdots, \mathbf{X}^{(V)}\}$, number of views V, maximum epoch K, update interval T.
Output: Cluster assignments $\mathcal{C} = \{\mathcal{C}_i\}_{i=1}^{n}$.

Stage 1: Preprocessing
$\mathcal{G} = \{\mathcal{G}^{(1)}, \cdots, \mathcal{G}^{(V)}\}$ = GRAPHCONSTRUCTION($\mathcal{X}$)
Obtain $\mathcal{A} = \{\mathcal{A}^{(1)}, \cdots, \mathcal{A}^{(V)}\}$ based on $\mathcal{G}$
Stage 2: Pretraining
for $v \in \{0, 1, \cdots, V\}$ **do**
  for $epoch \in \{1, 2, \cdots, K\}$ **do**
    $\mathbf{H}^{(v)}, \overline{\mathbf{X}}^{(v)}, \overline{\mathbf{A}}^{(v)} = \mathrm{SGAE}_{pm}^{(v)}(\mathbf{X}^{(v)}, \hat{\mathbf{A}}^{(v)})$
    $loss = \mathcal{L}_r(\mathbf{X}^{(v)}, \overline{\mathbf{X}}^{(v)}) + \mathcal{L}_s(\tilde{\mathbf{A}}^{(v)}, \overline{\mathbf{A}}^{(v)})$
  end for
end for
Set $\mathcal{H} = \{\mathbf{H}^{(1)}, \cdots, \mathbf{H}^{(V)}\}$
$\mathbf{H}, \mathbf{U}$ = FUSIONMODULE($\mathcal{H}, \mathcal{A}$)
for $epoch \in \{1, 2, \cdots, K\}$ **do**
  $\overline{\mathbf{H}}, \overline{\mathbf{U}} = \mathrm{SGAE}_{sm}(\mathbf{H}, \mathbf{U})$
  $loss = \mathcal{L}_r(\mathbf{H}, \overline{\mathbf{H}}) + \mathcal{L}_s(\tilde{\mathbf{U}}, \overline{\mathbf{U}})$
end for
Stage 3: Finetuning
Initialize DFP-GNN with the pretrained weights.
for $epoch \in \{1, 2, \cdots, K\}$ **do**
  $\overline{\mathbf{H}}, \overline{\mathbf{U}}, \mathbf{Q} = \text{DFP-GNN}(\mathcal{X}, \mathcal{A})$
  Optimize the combined objective $\mathcal{L}_{rsc}$
  if $epoch/T = 0$ **then**
    Update $\mathbf{P}$ based on $\mathbf{Q}$
  end if
end for
Obtain cluster assignments $\mathcal{C}$
return $\mathcal{C}$

where $\tilde{\mathbf{U}}$ is derived from the fused embedding in the pre-training stage with $\overline{\mathbf{U}}$ being the predicted relationships. By optimizing (13), the learned space retains the topological structure in the input space and can release the over-smoothing problem appearing in multi-layer graph convolution networks.

We introduce a self-training clustering objective for the clustering layer to incorporate the assignment process into embedding learning. Formally, the optimization of clustering loss is computed as follows:

$$\mathcal{L}_c(\mathbf{P}, \mathbf{Q}) = KL(\mathbf{P} \| \mathbf{Q}) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{14}$$

where $KL(\cdot \mid \cdot)$ is Kullback-Leibler divergence between two distributions. Finally, the combined optimization objective is reformulated as follows:

$$\mathcal{L}_{rsc} = \mathcal{L}_r + \lambda_1 \mathcal{L}_s + \lambda_2 \mathcal{L}_c. \tag{15}$$

Here $\lambda_1 > 0$ and $\lambda_2 > 0$ is the coefficients of $\mathcal{L}_s$ and $\mathcal{L}_c$, respectively. Finally, the parameters of DFP-GNN can be updated during the back-propagation process. Although DFP-GNN is designed for clustering tasks, it is possible to extend our method to classification tasks. Furthermore, the procedure of DFP-GNN is summarized in Algorithm 1.

D. Complexity of DFP-GNN

We analyze the computational complexity of DFP-GNN in the finetuning stage. Specifically, the forward propagation process contains three parts: view-specific propagation, dual fusion process and cross-view propagation. The total computation cost of the preliminary module is $\mathcal{O}(VLnd_h(n+d_h))$, where d_h denotes the dimension of the preliminary module. Then, the fusion module is mainly based on matrix addition and its complexity is $\mathcal{O}(Vnd_h)$. Finally, the complexity of cross-view propagation is $\mathcal{O}(Mnd_z(n+d_z))$, where d_z represents the dimension of embedding in the shared module. As a result, the total computational complexity of DFP-GNN is summed as $\mathcal{O}(VLnd_h(n+d_h) + Mnd_z(n+d_z))$. Suppose the optimizations stop after t iterations, the final computational complexity can be simplified as $\mathcal{O}(tndV(n+d))$, since the values of L and M are usually set as two or three and d represents the dimensions of features.

IV. EXPERIMENTS

In this section, the superiority of DFP-GNN is verified with eleven clustering algorithms on multiple available real-world benchmark datasets. We first describe the experimental implementation in detail. Then, the clustering results of all compared methods are shown to demonstrate the performance of DFP-GNN. Finally, comprehensive additional experiments are conducted to verify the sensitivity of different hyperparameters or components.

A. Experiment setup

Datasets. Several famous benchmark datasets with multiple sources are adopted in our experiments.

- **BBCSport** is a document dataset with 544 samples and five topical areas. Two different kinds of views with 3,183-D and 3,203-D are adopted.
- **Caltech101** is an image dataset with six types of features: 48-D Gabor, 40-D wavelet moment (WM), 254-D CENTRIST, 1,984-D histogram of oriented gradient (HOG), 512-D GIST and 928-D local binary pattern (LBP).
- **Cifar-10K**: CIFAR consists of 60,000 images with ten classes and the first 10,000 samples are used. Six kinds of views are extracted: 768-D color histogram (CH), 512-D GIST and 324-D HOG.
- **Citeseer** is a scientific document benchmark dataset with 3,312 samples. It consists of two kinds of features: 3,703-D content feature and 3,312-D citation feature.
- **GRAZ02** is an object categorization dataset with high complexity and intra-class variability. Six visual views are adopted: 512-D SIFT, 32-D SURF, 256-D GIST, 500-D LBP, 500-D PHOG, and 680-D wavelet texture (WT).
- **MNIST-10K** contains 10,000 handwritten digit images with three types of visual features: 30-D IsoProjection, 9-D linear discriminant analysis, and 30-D neighborhood preserving embedding.
- **MSRCv1** is an object image dataset with five visual feature sources: 24-D color moment (CM), 576-D HOG, 512-D GIST, 256-D LBP, and 256-D CENTRIST.

TABLE I
A BRIEF DESCRIPTION OF THE TESTED DATASETS

Datasets	# Samples	# Views	# Distribution of dimensions						# Classes
BBCSport	544	2	3,183	3,203	-	-	-	-	5
Caltech101	9,144	6	48	40	254	1,984	512	928	101
Cifar-10K	10,000	3	768	512	324	-	-	-	10
Citeseer	3,312	2	3,703	3,312	-	-	-	-	6
GRAZ02	1,476	6	512	32	256	500	500	680	4
MNIST-10K	10,000	3	30	9	30	-	-	-	10
MSRCv1	210	5	24	576	512	256	254	-	7
NUS-WIDE	2,400	6	64	144	73	128	225	500	12
Out-Scene	2,688	4	512	432	256	48	-	-	8
Youtube	2,000	6	2,000	1,024	64	512	64	647	10

- **NUS-WIDE** is an object recognition dataset with six available features: 64-D CH, 144-D color correlogram, 73-D edge direction histogram, 225-D block-wise color moment, 128-D WT and 500-D SIFT.
- **Out-Scene** has 2,688 images with eight groups and four different kinds of features are extracted: 512-D GIST, 432-D CM, 256-D HOG, and 48-D LBP.
- **Youtube** is a video database with six views: 2,000-D cuboids histogram, 1,024-D hist motion estimate, 64-D HOG, 512-D MFCC, 64-D volume stream and 647-D spectrogram stream features.

A summary of dataset statistic information, including the number of samples, number of clusters, distribution of dimensions and so on, is shown in Table I.

Evaluation Metrics. Although the proposed method is used for clustering, we assume that the true class label information is available during the evaluation stage. Specifically, six mainstream evaluation metrics and protocols, including clustering Accuracy (ACC), Normalized Mutual Information (NMI), Purity, Adjusted Rand Index (ARI), Precision and F-score are adopted. The values of all metrics lie in [0, 1] except for ARI, which is ranged in [-1, 1]. The higher values they are, the better performance they achieve.

Compared Methods. We verify the efficiency of DFP-GNN with several multi-view clustering methods in our experiments.

- **k-means** [49]: Multi-view data are stacked into a high-dimensional single form and then grouped by k-means to provide a benchmark for single-view clustering.
- **RMSC** [33]: Robust multi-view spectral clustering guides multi-view clustering by constructing a shared probability transition matrix with low-rank constraints.
- **AMGL** [30]: The auto-weighted multiple graph learning method trains the weights of views adaptively and calculates the embedding matrix in the subspace.
- **SwMC** [50]: It constructs a Laplacian rank constrained graph model to build a private graph for each view with a shared nearest graph similarity matrix.
- **MVGL** [51]: Graph learning for multi-view clustering constrains the rank of the global graph to ensure that the clustering results are obtained directly.
- **AE^2-Nets** [14]: Autoencoder in autoencoder networks combines two autoencoders networks (autoencoders networks and degradation networks) into their framework to integrate a complete potential representation.
- **MSC-IAS** [52]: It adopts the Hilbert–Schmidt Independence to learn a latent embedding space by encoding complementary information.
- **MCGC** [53]: It learns a consensus graph by optimizing the imparity among different input views and constraining the rank of the Laplacian matrix.
- **BMVC** [13]: Binary multi-view clustering proposes a joint learning framework that addresses compact collaborative discrete representation and binary clustering structure learning simultaneously.
- **LMVSC** [10]: Large-scale multi-view subspace clustering algorithm obtains clustering results by integrating different anchor points graph into the latent subspace.
- **GMC** [5]: Graph-based multi-view clustering promotes mutual enhancement between the single view graph and learned unified view graph in a self-weighted way.

Implementation Details. For all compared approaches, most hyperparameters are set according to the statements in the original papers. Furthermore, we adjust the corresponding hyperparameters of BMVC, MSC-IAS, MCGC and LMVSC to achieve the best results.

For the proposed method, we set the number of common neighborhoods as $\tau = 2$ when constructing an adjacent graph for each view. For all data sources, each row data (an input data point) is normalized to be $\|\tilde{\mathbf{x}}_i\|_2^2 \approx 1$, where $\tilde{\mathbf{x}}_i$ is the normalized form of $\mathbf{x}_i$. The dimensions of each view-specific propagation encoder are set constant to d_v-512-2,048-256 for all benchmark datasets. For the shared module, the dimensional distribution of the stacked graph autoencoder is 256-64-16-64-256. In the module-wise pretraining stage, the learning rate is set to 10^{-5} and the coefficient of structure preservation loss is set as 0.001 constant. The pretraining processes of view-specific encoders and cross-view propagation network would be stopped when more than 100 epochs do not reduce the loss value compared to the current minimum value. In the finetuning stage, the learning rate and weight decay of the optimizer are set as 10^{-5} and 10^{-6}, respectively. The update interval T is set as 50 and the training process would be stopped when the cluster assignment rate is less than 10^{-5}. The values of λ_1 and λ_2 are varied over different datasets. The whole framework is optimized by RMSprop with a momentum value of 0.9. The maximum number of epochs is set as $K = 20,000$. We repeat each experiment of all methods five times and use the mean values as the results.

TABLE II
MULTI-VIEW CLUSTERING PERFORMANCES OF ALL COMPARED METHODS
THE BEST AND SECOND BEST RESULTS ARE HIGHLIGHTED IN BOLD AND UNDERLINED, RESPECTIVELY

Datasets \ Methods		k-means	RMSC	AMGL	SwMC	MVGL	AE^2-Nets	MSC-IAS	MCGC	BMVC	LMVSC	GMC	DFP-GNN
BBCSport	ACC	0.4926	0.4434	0.3379	0.3621	0.3989	0.2982	0.8860	0.4099	0.4099	0.3474	0.8070	**0.9449**
	NMI	0.2519	0.1637	0.0148	0.0155	0.0623	0.0178	0.7129	0.0781	0.0816	0.1154	0.7226	**0.8521**
	Purity	0.5404	0.5077	0.3621	0.3658	0.4026	0.3790	0.8860	0.4154	0.4099	0.5165	0.8438	**0.9522**
	ARI	0.1733	0.1052	0.0011	0.0015	0.0209	0.0050	0.7170	0.0323	0.0627	0.0372	0.7218	**0.8593**
	Precision	0.3274	0.3124	0.2381	0.2391	0.2468	0.2481	0.7901	0.2513	0.2732	0.3979	0.7271	**0.8956**
	F-score	0.4360	0.3318	0.3354	0.3839	0.3902	0.2712	0.7840	0.3958	0.3456	0.3148	0.7943	**0.8974**
Caltech101	ACC	0.1370	0.1352	0.2040	0.2278	0.1293	0.1928	0.2175	0.2745	0.1995	0.1153	0.1950	**0.3187**
	NMI	0.3040	0.3064	0.3823	0.2235	0.1207	0.3779	**0.4285**	0.3335	0.3936	0.2503	0.2379	0.4239
	Purity	0.2914	0.2975	0.3978	0.3077	0.1960	0.3544	**0.4345**	0.3918	0.3748	0.1903	0.3012	0.3643
	ARI	0.0835	0.1000	0.0654	0.0058	0.0096	0.1464	0.1249	0.0148	0.1807	0.0275	0.0042	**0.3367**
	Precision	0.1456	0.2058	0.0723	0.0253	0.0235	0.2593	0.1990	0.0359	**0.3175**	0.0647	0.0261	0.2828
	F-score	0.1014	0.1141	0.1030	0.0481	0.0452	0.1771	0.1426	0.0652	0.1949	0.0578	0.0496	**0.3189**
Cifar-10K	ACC	0.1832	0.2315	0.1819	0.1055	0.1005	0.1346	0.2155	0.2673	0.2804	0.1727	0.2025	**0.3160**
	NMI	0.0548	0.1174	0.0698	0.0032	0.0041	0.0162	0.1142	0.1342	0.1576	0.0516	0.1141	**0.2302**
	Purity	0.1930	0.2458	0.1832	0.1062	0.1071	0.2038	0.2176	0.2680	0.2835	0.3492	0.2032	**0.3827**
	ARI	0.0324	0.0689	0.0238	0.0000	0.0000	0.0068	0.0725	0.0403	0.1112	0.0257	0.0689	**0.1442**
	Precision	0.1232	0.1554	0.1118	0.1000	0.1000	0.1065	0.1563	0.1227	0.1789	0.2068	0.1339	**0.2277**
	F-score	0.1509	0.1719	0.1884	0.1817	0.1817	0.1530	0.1783	0.1848	0.2211	0.1499	0.2263	**0.2342**
Citeseer	ACC	0.4758	0.4432	0.2393	0.2144	0.2080	0.2105	0.2656	0.2334	0.2171	0.3946	0.2174	**0.6153**
	NMI	0.2338	0.1999	0.0309	0.0066	0.0126	0.0037	0.0524	0.0239	0.0143	0.1349	0.0071	**0.3585**
	Purity	0.4855	0.4668	0.2650	0.2165	0.2168	0.3322	0.2766	0.2415	0.2310	0.5459	0.2180	**0.6790**
	ARI	0.2104	0.1797	0.0234	0.0003	0.0016	0.0006	0.0068	0.0010	0.0002	0.0619	0.0009	**0.3580**
	Precision	0.3267	0.3313	0.1970	0.1784	0.1778	0.1803	0.1815	0.1781	0.1786	0.4033	0.1789	**0.4782**
	F-score	0.3701	0.3220	0.2042	0.3016	0.3005	0.2684	0.2971	0.2976	0.2484	0.2804	0.3030	**0.4674**
GRAZ02	ACC	0.3598	0.3321	0.4744	0.3733	0.2785	0.3690	0.4188	0.4221	0.3232	0.3564	0.4722	**0.5589**
	NMI	0.0321	0.0355	0.1333	0.0636	0.0129	0.0583	0.0885	0.0820	0.0287	0.0362	0.1330	**0.1837**
	Purity	0.3598	0.3501	0.4817	0.3835	0.2879	0.5226	0.4217	0.4221	0.3428	0.5359	0.4885	**0.5813**
	ARI	0.0357	0.0272	0.1273	0.0532	0.0006	0.0491	0.0802	0.0726	0.0180	0.0359	0.1251	**0.1827**
	Precision	0.2721	0.2711	0.3354	0.2778	0.2521	0.2932	0.3109	0.3038	0.2623	0.3900	0.3263	**0.3915**
	F-score	0.3368	0.2919	0.3697	0.3792	**0.4000**	0.3080	0.3164	0.3162	0.3243	0.3217	0.3867	0.3950

All experiments are conducted on an Ubuntu-16.04 system, which contains a 2.1GHz Intel Xeon CPU and an Nvidia Tesla P100 GPU. The implementation of DFP-GNN is available at https://github.com/Xiaoshunxin/DFP-GNN.

B. Multi-view clustering

Quantitative Results. Comprehensive experiments are conducted to validate the superiority of our framework DFP-GNN in both quantitatively and qualitatively. In Table II and Table III, the results of all approaches on ten popular benchmark datasets are reported with six evaluation metrics. DFP-GNN achieves the best or second-best results, sometimes obtaining a significant margin, on all datasets over different evaluation protocols except for Caltech101 with the Purity metric. It shows that DFP-GNN obtains significant improvements compared to various multi-view clustering approaches.

Furthermore, several interesting phenomena can be observed from the clustering results. First, DFP-GNN achieves competitive performance on different datasets, demonstrating that it is suitable for various clustering applications in practice. As another deep learning-based algorithm, second, the performance of AE^2-Nets is worse than our method in all situations. A possible reason is that AE^2-Nets utilizes a common autoencoder instead of graph convolution and fails to capture the feature and structure information simultaneously. Another possible reason is that AE^2-Nets fails to utilize the clustering process to guide the training process. Third, most of the compared algorithms achieve worse performance on the Citeseer dataset, in which there exist real-world relationships between samples. In contrast, DPF-GNN obtains the best result since the build graph captures the relationship between nodes and makes the message passing mechanism play an important role. Furthermore, LMVSC is efficient for large-scale datasets, such as Caltech101, Cifar-10K and MNIST-10K, but it achieves general performance on those datasets. For example, LMVSC obtains the worst ACC result on the Caltech101 dataset. Compared with methods based on traditional graph learning, our framework always achieves a robust ARI performance on several benchmarks, including Caltech101, Cifar-10K and Citeseer. In addition, DFP-GNN does not achieve performance on Caltech101 as well as other datasets. One possible reason is that most of the hyper-parameters of DFP-GNN are decided from the datasets with only a few clusters. As a result, the adjusted hyper-parameters may be unsuitable for Caltech101, which has 101 different clusters.

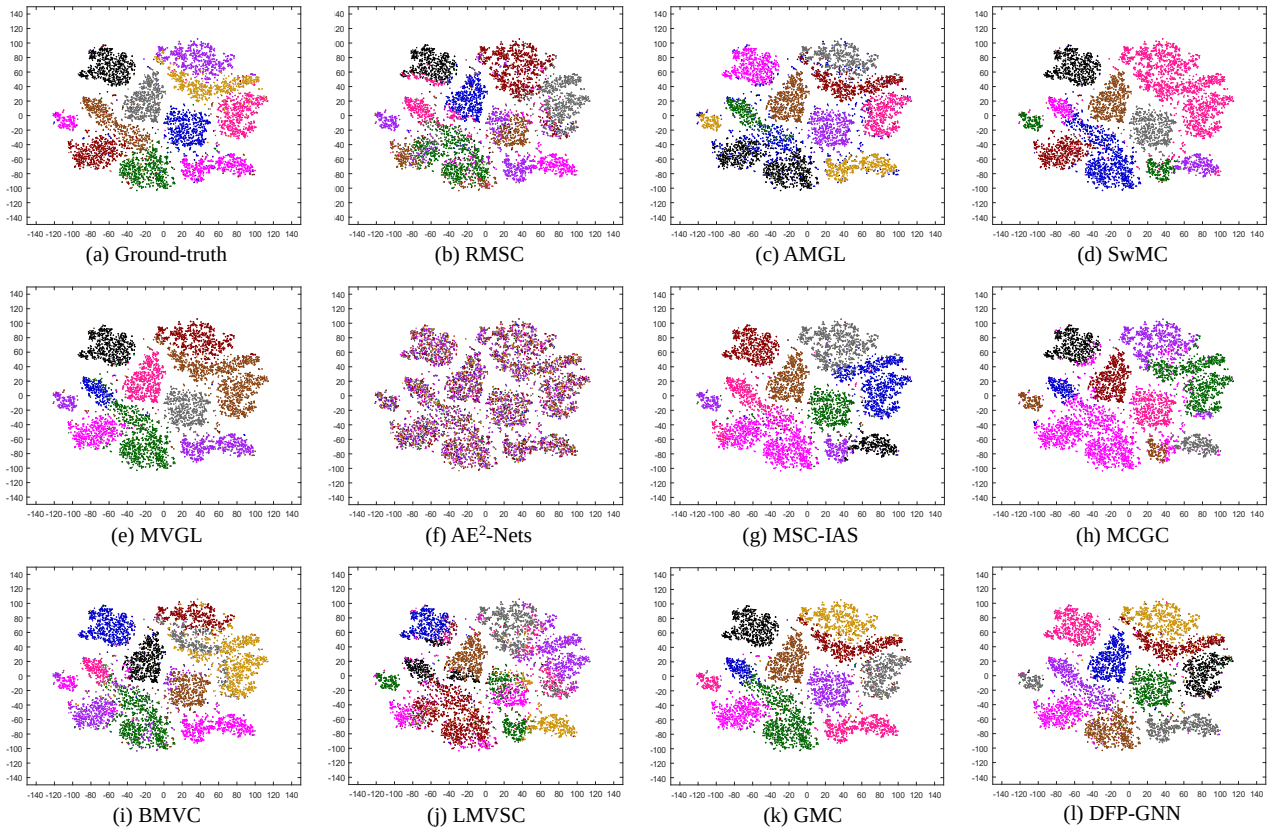
Results Visualization. To intuitively demonstrate the clustering results of different methods, we concatenate input multiple views and then utilize t-SNE to reduce the concatenated form into a two dimensions space. From the visualization shown in Fig. 2, we can observe that DFP-GNN is able to obtain a clustering assignment, which is more similar to the truth labels by comparing with other methods.

C. Ablation study

Several additional experiments are conducted to verify the designed DFP-GNN framework. Most of the configurations of

TABLE III
MULTI-VIEW CLUSTERING PERFORMANCES OF ALL COMPARED METHODS
THE BEST AND SECOND BEST RESULTS ARE HIGHLIGHTED IN BOLD AND UNDERLINED, RESPECTIVELY

Datasets \ Methods		k-means	RMSC	AMGL	SwMC	MVGL	AE^2-Nets	MSC-IAS	MCGC	BMVC	LMVSC	GMC	DFP-GNN
MNIST-10K	ACC	0.5643	0.5872	0.8177	0.6712	0.8147	0.7185	0.6982	0.6103	0.5355	0.5177	0.9037	**0.9299**
	NMI	0.5219	0.5211	0.8002	0.7148	0.7886	0.6172	0.7175	0.6124	0.4838	0.4902	**0.8427**	0.8392
	Purity	0.5949	0.6124	0.8362	0.7171	0.8147	0.7377	0.7485	0.6570	0.5445	0.5450	0.9037	**0.9315**
	ARI	0.4206	0.4305	0.7446	0.5678	0.7318	0.5534	0.6283	0.4349	0.3577	0.3494	0.8244	**0.8544**
	Precision	0.4624	0.4593	0.7437	0.4911	0.6779	0.6040	0.5959	0.3992	0.4118	0.4264	0.8153	**0.8664**
	F-score	0.4811	0.4917	0.7715	0.6171	0.7621	0.6140	0.6703	0.5070	0.4241	0.4163	0.8426	**0.8667**
MSRCv1	ACC	0.4667	0.2343	0.6838	0.7048	0.6333	0.7314	0.7971	0.7810	0.6476	0.3429	0.7476	**0.9238**
	NMI	0.4053	0.0579	0.6299	0.6745	0.5576	0.6604	0.7403	0.6601	0.6083	0.2460	0.7421	**0.8429**
	Purity	0.4810	0.2419	0.7076	0.7429	0.6333	0.7790	0.8057	0.7810	0.6952	0.3810	0.7905	**0.9286**
	ARI	0.2569	0.0060	0.4980	0.5249	0.3949	0.5781	0.6492	0.5837	0.4785	0.1250	0.6400	**0.8291**
	Precision	0.3208	0.1437	0.5095	0.5051	0.3972	0.6404	0.6725	0.6147	0.5229	0.2496	0.6121	**0.8602**
	F-score	0.3785	0.1486	0.5796	0.6037	0.5007	0.6601	0.6998	0.6443	0.5554	0.2474	0.6968	**0.8595**
NUS-WIDE	ACC	0.2100	0.2433	0.2266	0.1329	0.1075	0.2202	0.2449	0.1879	0.2121	0.1812	0.1650	**0.2942**
	NMI	0.1095	0.1215	0.1381	0.0589	0.0360	0.1067	0.1270	0.1019	0.0970	0.0679	0.0788	**0.1612**
	Purity	0.2138	0.2629	0.2479	0.1442	0.1175	0.2637	0.2798	0.2117	0.2279	0.3404	0.1787	**0.3279**
	ARI	0.0420	0.0641	0.0514	0.0042	0.0019	0.0526	0.0637	0.0156	0.0610	0.0179	0.0124	**0.0941**
	Precision	0.1080	0.1410	0.1162	0.0849	0.0838	0.1415	0.1358	0.0907	0.1278	0.1815	0.0888	**0.1850**
	F-score	0.1592	0.1427	0.1596	0.1550	0.1539	0.1468	0.1489	0.1571	0.1561	0.1248	0.1592	**0.1824**
Out-Scene	ACC	0.3307	0.4477	0.5388	0.2731	0.3002	0.5655	0.4972	0.5182	0.5737	0.5874	0.3400	**0.7478**
	NMI	0.1773	0.3569	0.5105	0.1324	0.1903	0.4766	0.4290	0.3710	0.5288	0.4512	0.3142	**0.5961**
	Purity	0.3430	0.4670	0.5791	0.2812	0.3077	0.6305	0.5170	0.5186	0.6001	0.6272	0.3501	**0.7708**
	ARI	0.1222	0.2580	0.3832	0.0203	0.0824	0.3812	0.3107	0.1859	0.4021	0.3727	0.1925	**0.5199**
	Precision	0.2155	0.3494	0.4016	0.1362	0.1662	0.4714	0.3693	0.2358	0.4534	0.4572	0.2243	**0.6006**
	F-score	0.2531	0.3533	0.4762	0.2331	0.2754	0.4868	0.4072	0.3309	0.4829	0.4531	0.3546	**0.6033**
Youtube	ACC	0.2495	0.2564	0.1412	0.1175	0.1245	0.2535	0.2453	0.2250	0.1445	0.2005	0.1165	**0.3775**
	NMI	0.1577	0.1460	0.0335	0.0205	0.0255	0.1794	0.1517	0.1050	0.0198	0.0576	0.0204	**0.2394**
	Purity	0.2875	0.2875	0.1463	0.1205	0.1270	0.3430	0.2881	0.2355	0.1470	0.3440	0.1205	**0.4425**
	ARI	0.0876	0.0876	0.0076	0.0003	0.0009	0.0950	0.0750	0.0559	0.0130	0.0306	0.0004	**0.1496**
	Precision	0.1649	0.1746	0.1059	0.0997	0.1000	0.1948	0.1631	0.1317	0.1073	0.2192	0.0997	**0.2533**
	F-score	0.1951	0.1826	0.1138	0.1805	0.1805	0.2141	0.1723	0.1953	0.1543	0.1553	0.1806	**0.2529**



(a) Ground-truth (b) RMSC (c) AMGL (d) SwMC
(e) MVGL (f) AE^2-Nets (g) MSC-IAS (h) MCGC
(i) BMVC (j) LMVSC (k) GMC (l) DFP-GNN

Fig. 2. Visualization of the ground-truth labels and the clustering results of all compared multi-view clustering algorithms on the MNIST-10K dataset.

TABLE IV
IMPACT OF DUAL FUSION MEASURED BY ACC, NMI, PURITY AND PRECISION. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

Datasets	Caltech101				Citeseer				GRAZ02			
Metrics	ACC	NMI	Purity	Precision	ACC	NMI	Purity	Precision	ACC	NMI	Purity	Precision
OAF	0.3059	0.4137	0.3595	0.2798	0.4831	0.2287	0.5682	0.3595	0.4058	0.0857	0.4858	0.3106
OSF	0.2740	0.3767	0.3027	0.2362	0.5652	0.3127	0.6489	0.4363	0.5027	0.1928	0.5718	**0.3930**
Ours	**0.3187**	**0.4239**	**0.3643**	**0.2828**	**0.6153**	**0.3585**	**0.6790**	**0.4782**	**0.5589**	**0.1837**	**0.5813**	0.3915

TABLE V
IMPACT OF DUAL PROPAGATION MEASURED BY ACC, PURITY, PRECISION AND F-SCORE. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

Datasets	Caltech101				Citeseer				GRAZ02			
Metrics	ACC	Purity	Precision	F-score	ACC	Purity	Precision	F-score	ACC	Purity	Precision	F-score
MLP/MLP	0.0875	0.0875	0.0283	0.0550	0.2117	0.3315	0.1788	0.3033	0.3991	0.5467	0.3144	0.3126
MLP/GNN	0.2206	0.2207	0.1285	0.2000	0.3270	0.4411	0.2414	0.2370	0.2846	0.4580	0.2528	0.4036
GNN/MLP	0.1738	0.1974	0.0734	0.1158	0.4553	0.5178	0.3320	0.3362	0.2846	0.4580	0.2528	**0.4036**
DFP-GNN	**0.3187**	**0.3643**	**0.2828**	**0.3189**	**0.6153**	**0.6790**	**0.4782**	**0.4674**	**0.5589**	**0.5813**	**0.3915**	0.3950

TABLE VI
THE CLUSTERING RESULTS TRAINED BY DIFFERENT OBJECTIVES. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD

Datasets	Caltech101				Citeseer				GRAZ02			
Metrics	ACC	NMI	ARI	F-score	ACC	NMI	ARI	F-score	ACC	NMI	ARI	F-score
$\mathcal{L}_r$	0.2207	0.4382	0.1918	0.1935	0.5447	0.3212	0.2778	0.4207	0.4533	0.1165	0.1052	0.3385
$\mathcal{L}_s$	0.2489	0.4684	0.2943	0.2318	0.5864	0.3489	0.3057	0.4484	0.4885	0.1465	0.1395	0.3680
$\mathcal{L}_c$	0.2504	0.3493	0.1397	0.2758	0.4043	0.2023	0.1399	0.3125	0.5495	0.1834	0.1823	0.3941
$\mathcal{L}_{rs}$	0.2410	**0.4715**	0.2714	0.2299	0.6099	0.3579	0.3284	0.4538	0.4925	0.1311	0.1319	0.3576
$\mathcal{L}_{rc}$	0.2768	0.3838	0.1585	**0.3344**	0.4004	0.2006	0.1370	0.3125	0.5467	0.1758	0.1726	0.3877
$\mathcal{L}_{sc}$	0.3126	0.4151	0.3138	0.3122	0.6105	0.3545	0.3506	0.4630	0.5467	0.1763	0.1780	0.3901
$\mathcal{L}_{rsc}$	**0.3187**	0.4239	**0.3367**	0.3189	**0.6153**	**0.3585**	**0.3580**	**0.4674**	**0.5589**	**0.1837**	**0.1827**	**0.3950**

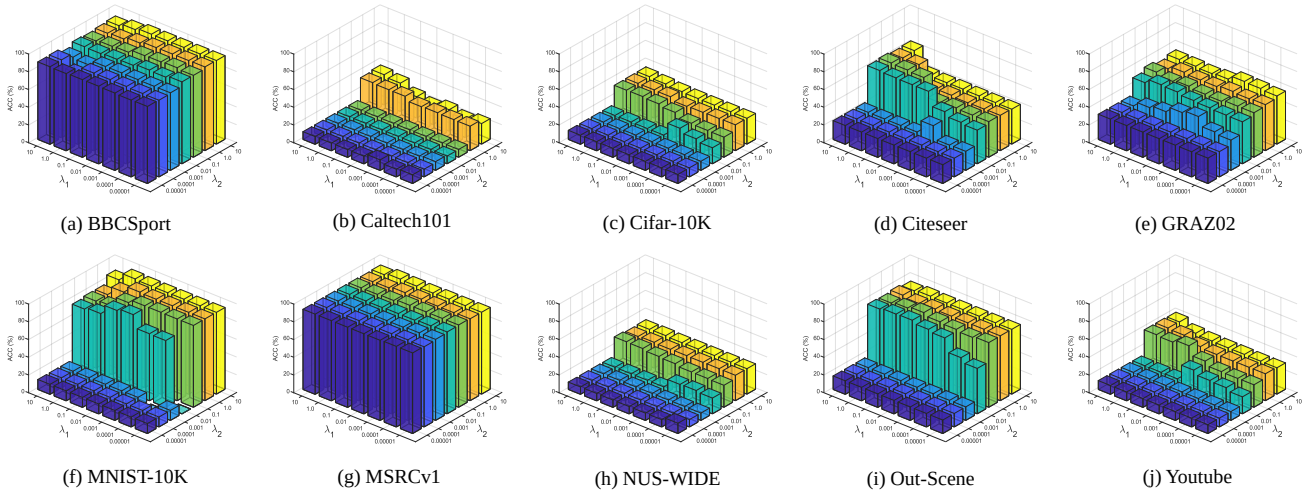
all ablation studies are set according to the implementation details except for the factors of the current analysis.

Impact of Dual Fusion. To validate the influence of the adaptively dual fusion mechanism, we conduct an ablation study to compare two single fusions with DFP-GNN. Specifically, the first single fusion is attribute fusion (OAF) and the second is structure fusion (OSF). For OAF, the transition matrix used for the shared module is generated from the fused feature matrix during the pretraining stage. For OSF, we concatenate all learned preliminary embeddings to formulate the fused representation. As shown in Table IV, the dual fusion mechanism achieves the best result than two single fusion strategies except for Precision on GRAZ02. In addition, structure fusion is important for multi-view learning. However, most existing GNN-based methods usually only perform feature-level fusion.

Influence of Dual Propagation. We conduct an additional study to prove helpful of the designed dual propagation mechanism for multi-view clustering. Specifically, we replace the graph convolution operation with a multilayer perceptron for both the preliminary and the shared modules. As a result, there exist three different variants of DFP-GNN and termed MLP/MLP, MLP/GNN and GNN/MLP, respectively. For example, GNN/MLP means that the graph convolution used in the shared module is replaced with a multilayer perceptron. Several observations can be obtained from Table V. First, dual propagation achieves significant improvements over other variants. Second, DFP-GNN usually has a better result than GNN/MLP and it demonstrates that applying message passing over different views is important. However, most existing methods only apply information propagation for each view individually. Finally, MLP/GNN and GNN/MLP achieve a worse performance than MLP/MLP on the GRAZ02 dataset. A possible reason is that the value of $\mathcal{L}_c$ is too small to guide the clustering assigning.

Effectiveness of Combined Loss. We compare all single objectives $\mathcal{L}_r$, $\mathcal{L}_s$ and $\mathcal{L}_c$, all dual combined objectives $\mathcal{L}_{rs}$, $\mathcal{L}_{rc}$ and $\mathcal{L}_{sr}$ with the final optimization target $\mathcal{L}_{rsc}$ to validate the effectiveness of the designed optimization objective. For those objectives without clustering loss $\mathcal{L}_c$, the training would be stopped when more than 100 epochs do not reduce the minimum loss value and the learned embedding Z will be used as the input of k-means to obtain the cluster assignments. Table VI reports the corresponding results of various objectives in the finetuning stage. Observation from Table VI shows that $\mathcal{L}_{rsc}$ achieves competitive results in most situations, except NMI and F-score on Caltech101. In addition, $\mathcal{L}_s$ and $\mathcal{L}_c$ have different effects on different datasets. For example, $\mathcal{L}_c$ is important for Caltech101 and GRAZ02. This phenomenon demonstrates that the proposed combined objective is suitable for various clustering tasks.

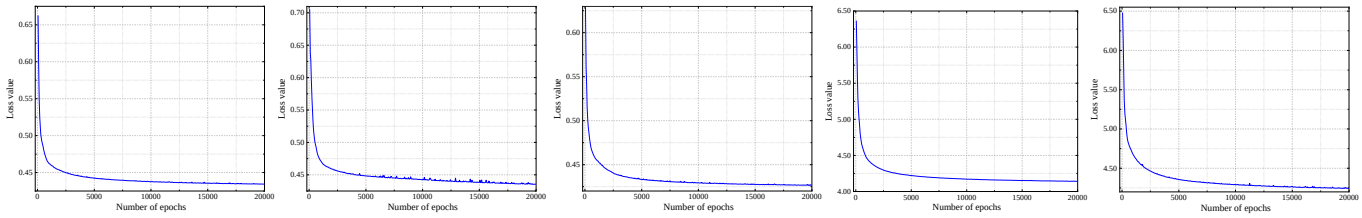
Impact of Coefficients λ_1 and λ_2. We design an experiment to illustrate the sensitivity of λ_1 and λ_2 with different values and explore the relation among them measured by the



(a) BBCSport (b) Caltech101 (c) Cifar-10K (d) Citeseer (e) GRAZ02

(f) MNIST-10K (g) MSRCv1 (h) NUS-WIDE (i) Out-Scene (j) Youtube

Fig. 3. Sensitivity analysis of parameters λ_1 and λ_2 for the proposed DFP-GNN framework on all benchmark datasets measured by the ACC protocol.



(a) Cifar-10K (b) Citeseer (c) NUS-WIDE (d) Out-Scene (e) Youtube

Fig. 4. The convergence curve of the proposed multi-view clustering method. Results on five benchmark datasets with 20,000 training epochs are reported.

ACC metric. The values of λ_1 and λ_2 are taken from 0.00001 to 10. As depicted in Fig. 3, DFP-GNN usually achieves better results when λ_1 and λ_2 have larger values. It demonstrates that the used structure preservation loss $\mathcal{L}_s$ and clustering objective $\mathcal{L}_c$ are both important for multi-view clustering. Furthermore, the performance would drop greatly when the value of $\mathcal{L}_c$ is too small on most datasets except BBCSport and MSRCv1. A possible reason is that the cluster assignments are derived from the clustering layer, which is optimized by the $\mathcal{L}_c$. The small weight of $\mathcal{L}_c$ can result in the situation that the clustering layer does not play the role of assigning.

Convergence Analysis. We plot the value of the combined objective $\mathcal{L}_{rsc}$ during the training process to validate the convergence of our method. After the pretraining stage, DFP-GNN only finetunes parameters of the whole framework and the value of $\mathcal{L}_{rsc}$ would not change too much. Thus, we record the loss value of DFP-GNN without pretrained weights in the finetuning stage. In detail, we only record the value of $\mathcal{L}_{rsc}$ when updating the auxiliary target distribution $\mathbf{P}$. Since the training of other datasets would be stopped early, we report the results on five benchmarks, including Cifar-10K, Citeseer, NUS-WIDE, Out-Scene and Youtube. As shown in Fig. 4, our method can rapidly converge in the first 2,500 epochs and then gradually converge in the remaining epochs.

Complexity Analysis. Furthermore, we compare DFP-GNN with other multi-view clustering algorithms in terms of computational complexity in Table VII. In detail, t, c, n, d and V denote the number of iterations, groups, feature dimensions, samples and views, respectively. Observation from Table VII shows that most of the multi-view learning methods approximately cost time in $\mathcal{O}(n^2)$. Note that the computational complexity of the graph convolution operation used in DFP-GNN can be reduced by efficiently being implemented as a product of a sparse matrix with a dense matrix.

TABLE VII
COMPARISON OF COMPUTATIONAL COMPLEXITIES OF ALL MULTI-VIEW CLUSTERING ALGORITHMS

Methods	Computational complexity
RMSC	$\mathcal{O}(tn^2)$
AMGL	$\mathcal{O}(tcn^2)$
SwMC	$\mathcal{O}(tcn^2)$
MVGL	$\mathcal{O}(t(n^2V^2+nV^3))$
AE^2-Nets	$\mathcal{O}(tnd^2V)$
MSC-IAS	$\mathcal{O}(tn^2)$
MCGC	$\mathcal{O}(tn^2V)$
BMVC	$\mathcal{O}(tn(c+V))$
LMVSC	$\mathcal{O}(tn(d+c^2)+(n+V^2)V))$
GMC	$\mathcal{O}(tn^2(c+V)+ndV))$
DFP-GNN	$\mathcal{O}(tndV(n+d))$

V. CONCLUSION

We proposed a deep learning framework termed dual fusion-propagation graph neural network to capture multiple information among different views and then utilize them to refine the results of multi-view clustering. The view-specific propagation used in the preliminary module is efficient to capture the

complementarity and learn a discriminative representation for each view. Then, the dual fusion mechanism has been proved that it is able to adaptively combine both attribute and structure information among different views simultaneously. Finally, the consistent information can be captured by the shared module and the whole framework is optimized under a combined objective. Comprehensive experiments demonstrate that our method has achieved competitive results on ten popular benchmark datasets concurrently. Our future work includes generalizing the proposed DFP-GNN to multi-view semi-supervised learning tasks. Specifically, DFP-GNN will be used to generate high-confidence pseudo labels under the guide of a few supervised information. It would be efficient to propagate the supervised information to unlabeled samples via the adjacency matrix.

REFERENCES

[1] B. Wang, Y. Hu, J. Gao, Y. Sun, F. Ju, and B. Yin, "Learning adaptive neighborhood graph on grassmann manifolds for video/image-set subspace clustering," *IEEE Transactions on Multimedia*, vol. 23, pp. 216–227, 2021.

[2] S. Yoon, F. Dernoncourt, D. S. Kim, T. Bui, and K. Jung, "A compare-aggregate model with latent clustering for answer selection," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 2093–2096.

[3] Z. Zhou and A. A. Amini, "Analysis of spectral clustering algorithms for community detection: the general bipartite setting," *Journal of Machine Learning Research*, vol. 20, no. 47, pp. 1–47, 2019.

[4] A. Kanezaki, Y. Matsushita, and Y. Nishida, "Rotationnet for joint object categorization and unsupervised pose estimation from multi-view images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 269–283, 2019.

[5] H. Wang, Y. Yang, and B. Liu, "Gmc: graph-based multi-view clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 6, pp. 1116–1129, 2019.

[6] C. Tang, X. Liu, X. Zhu, E. Zhu, Z. Luo, L. Wang, and W. Gao, "Cgd: multi-view clustering via cross-view graph diffusion," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020, pp. 5924–5931.

[7] X. Li, H. Zhang, R. Wang, and F. Nie, "Multi-view clustering: a scalable and parameter-free bipartite graph fusion method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, doi: 10.1109/TPAMI.2020.3011148.

[8] X. Cao, C. Zhang, H. Fu, S. Liu, and H. Zhang, "Diversity-induced multi-view subspace clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 586–594.

[9] C. Zhang, Q. Hu, H. Fu, P. Zhu, and X. Cao, "Latent multi-view subspace clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4279–4287.

[10] Z. Kang, W. Zhou, Z. Zhao, J. Shao, M. Han, and Z. Xu, "Large-scale multi-view subspace clustering in linear time," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020, pp. 4412–4419.

[11] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-view clustering via joint nonnegative matrix factorization," in *Proceedings of the SIAM International Conference on Data Mining*, 2013, pp. 252–260.

[12] H. Zhao, Z. Ding, and Y. Fu, "Multi-view clustering via deep matrix factorization," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2017, pp. 2921–2927.

[13] Z. Zhang, L. Liu, F. Shen, H. T. Shen, and L. Shao, "Binary multi-view clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1774–1782, 2018.

[14] C. Zhang, Y. Liu, and H. Fu, "Ae2-nets: autoencoder in autoencoder networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2577–2585.

[15] Z. Huang, J. T. Zhou, X. Peng, C. Zhang, H. Zhu, and J. Lv, "Multi-view spectral clustering network," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 2563–2569.

[16] R. Zhou and Y.-D. Shen, "End-to-end adversarial-attention network for multi-modal clustering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 619–14 628.

[17] L. Liang, L. Jin, and Y. Xu, "Adaptive gnn for image analysis and editing," *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems*, pp. 3638–3649, 2019.

[18] H. Gao, Y. Chen, and S. Ji, "Learning graph pooling and hybrid convolutional operations for text representations," in *Proceedings of the 28th International Conference on World Wide Web*, 2019, pp. 2743–2749.

[19] L.-P. Xhonneux, M. Qu, and J. Tang, "Continuous graph neural networks," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 10 432–10 441.

[20] K. Liu, L. Gao, N. M. Khan, L. Qi, and L. Guan, "A multi-stream graph convolutional networks-hidden conditional random field model for skeleton-based action recognition," *IEEE Transactions on Multimedia*, vol. 23, pp. 64–76, 2021.

[21] D. Valsesia, G. Fracastoro, and E. Magli, "Learning localized representations of point clouds with graph-convolutional generative adversarial networks," *IEEE Transactions on Multimedia*, vol. 23, pp. 402–414, 2021.

[22] M. R. Khan and J. E. Blumenstock, "Multi-gcn: graph convolutional networks for multi-view networks, with applications to global poverty," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019, pp. 606–613.

[23] M. Zhang, T. Li, Y. Li, and P. Hui, "Multi-view joint graph representation learning for urban region embedding," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 4431–4437.

[24] F. Xue, X. Wu, S. Cai, and J. Wang, "Learning multi-view camera relocalization with graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 372–11 381.

[25] J. Cheng, Q. Wang, Z. Tao, D. Xie, and Q. Gao, "Multi-view attribute graph convolution networks for clustering," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 2973–2979.

[26] N. Xu, P. Wang, L. Chen, J. Tao, and J. Zhao, "Mr-gnn: multi-resolution and dual graph neural network for predicting structured entity interactions," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3968–3974.

[27] S. Fan, X. Wang, C. Shi, E. Lu, K. Lin, and B. Wang, "One2multi graph autoencoder for multi-view graph clustering," in *Proceedings of the 29th International Conference on World Wide Web*, 2020, pp. 3070–3076.

[28] S. Mai, H. Hu, and S. Xing, "Modality to modality translation: an adversarial representation learning and graph fusion network for multimodal fusion," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020, pp. 164–172.

[29] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 478–487.

[30] F. Nie, J. Li, X. Li *et al.*, "Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification." in *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016, pp. 1881–1887.

[31] Y. Xie, B. Lin, Y. Qu, C. Li, W. Zhang, L. Ma, Y. Wen, and D. Tao, "Joint deep multi-view learning for image clustering," *IEEE Transactions on Knowledge and Data Engineering*, 2020, doi: 10.1109/TKDE.2020.2973981.

[32] S. Du, Z. Liu, Z. Chen, W. Yang, and S. Wang, "Differentiable bi-sparse multi-view co-clustering," *IEEE Transactions on Signal Processing*, vol. 69, pp. 4623–4636, 2021.

[33] R. Xia, Y. Pan, L. Du, and J. Yin, "Robust multi-view spectral clustering via low-rank and sparse decomposition," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014, pp. 2149–2155.

[34] C. Tang, X. Zhu, X. Liu, M. Li, P. Wang, C. Zhang, and L. Wang, "Learning a joint affinity graph for multiview subspace clustering," *IEEE Transactions on Multimedia*, vol. 21, no. 7, pp. 1724–1736, 2018.

[35] X. Liu, X. Zhu, M. Li, L. Wang, C. Tang, J. Yin, D. Shen, H. Wang, and W. Gao, "Late fusion incomplete multi-view clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2410–2423, 2018.

[36] G. Andrew, R. Arora, J. Bilmes, and K. Livescu, "Deep canonical correlation analysis," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1247–1255.

[37] Y. Lin, Y. Gou, X. Liu, J. Bai, J. Lv, and X. Peng, "Dual contrastive prediction for incomplete multi-view representation learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022, doi: 10.1109/TPAMI.2022.3197238.

[38] M. Yang, Y. Li, P. Hu, J. Bai, J. C. Lv, and X. Peng, "Robust multi-view clustering with incomplete information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 1055–1069, 2023.
[39] X. Peng, Z. Huang, J. Lv, H. Zhu, and J. T. Zhou, "Comic: Multi-view clustering without parameter selection," in *Proceedings of the 36th International Conference on Machine Learning*, 2019, pp. 5092–5101.
[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the 4th International Conference on Learning Representations*, 2016.
[41] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.
[42] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of 31st Annual Conference on Neural Information Processing Systems*, 2017, pp. 1024–1034.
[43] Z. Wang, L. Zheng, Y. Li, and S. Wang, "Linkage based face clustering via graph convolution network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1117–1125.
[44] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: a deep attentional embedding approach," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3670–3676.
[45] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 874–883.
[46] S. Li, W.-T. Li, and W. Wang, "Co-gcn for multi-view semi-supervised learning," in *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, 2020, pp. 4691–4698.
[47] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 5453–5462.
[48] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 3538–3545.
[49] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.
[50] F. Nie, J. Li, X. Li *et al.*, "Self-weighted multiview clustering with multiple graphs," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 2564–2570.
[51] K. Zhan, C. Zhang, J. Guan, and J. Wang, "Graph learning for multiview clustering," *IEEE Transactions on Cybernetics*, vol. 48, no. 10, pp. 2887–2895, 2017.
[52] X. Wang, Z. Lei, X. Guo, C. Zhang, H. Shi, and S. Z. Li, "Multi-view subspace clustering with intactness-aware similarity," *Pattern Recognition*, vol. 88, pp. 50–63, 2019.
[53] K. Zhan, F. Nie, J. Wang, and Y. Yang, "Multiview consensus graph clustering," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1261–1270, 2018.

Shunxin Xiao received the B.S. degree from Guilin University of Electronic Technology, Guilin, China, in 2016, and the M.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China, in 2019. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. He was a visiting Ph.D. student at the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2022. His research interests include machine learning, graph neural networks, and computer vision.

Shide Du received his M.S. degree from the College of Computer and Data Science, Fuzhou University, Fuzhou, China in 2022. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His current research interests include machine learning, differentiable programming, and deep learning.

Zhaoliang Chen received his B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China in 2019. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. He is also currently visiting Faculty of Computer Science, University of Vienna, Vienna, Austria. His current research interests include machine learning, deep learning, graph neural networks and recommender systems.

Yunhe Zhang received her B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China in 2019. She is currently pursuing the M.S. degree with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. Her current research interests include feature selection, matrix factorization and deep learning.

Shiping Wang received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2014. He was a Research Fellow with Nanyang Technological University from August 2015 to August 2016. He is currently a Full Professor and Qishan Scholar with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. His research interests include machine learning, data mining, computer vision, optimization theory, and granular computing.