RGLN: ROBUST RESIDUAL GRAPH LEARNING NETWORKS VIA SIMILARITY-PRESERVING MAPPING ON GRAPHS

Jiaxiang Tang, Xiang Gao, Wei Hu

Wangxuan Institute of Computer Technology, Peking University

ABSTRACT

Graph Convolutional Neural Networks (GCNNs) extend CNNs to irregular graph data domain, such as brain networks, citation networks and 3D point clouds. It is critical to identify an appropriate graph for basic operations in GCNNs. Existing methods often manually construct or learn one fixed graph based on known connectivities, which may be sub-optimal. To this end, we propose a residual graph learning paradigm to infer edge connectivities and weights in graphs, which is cast as distance metric learning under a low-rank assumption and a similarity-preserving regularization. In particular, we learn the underlying graph based on *similarity-preserving mapping* on graphs, which keeps similar nodes close and pushes dissimilar nodes away. Extensive experiments on semi-supervised learning of citation networks and 3D point clouds show that we achieve the state-of-the-art performance in terms of both accuracy and robustness.

***Index Terms*—** Graph Learning, Graph Convolutional Neural Networks, Semi-supervised Learning, Point Cloud Classification

1. INTRODUCTION

Graph Convolutional Neural Networks (GCNNs) [1] have attracted increasing attention as a powerful tool for learning irregularly structured data on graphs, such as citation networks, social networks and 3D point clouds. The construction of an appropriate graph plays a critical role in GCNNs to guide basic operators such as graph convolution for accurate feature learning. In many scenarios where the graph is incomplete, inaccurate or even unavailable, it is crucial to learn a graph topology that characterizes the intrinsic structure of data.

Some previous studies construct graphs from data empirically, such as the commonly used k-Nearest-Neighbor (k-NN) graphs [2, 3], where the choice of parameters such as k is often empirical. Few methods exploit *graph learning* for optimized representation learning, which either learn a fixed and shared graph for all instances [4, 5], or an individual graph for each instance [6]. However, they often only learn edge weights assuming the availability of graph connectivities, and the learned graph is sometimes not robust to missing or noisy labels for practical applications.

To this end, we propose a residual graph learning network (RGLN), which learns a residual graph with both *new connectivities* and edge weights. We propose to learn the underlying graph from the perspective of *similarity-preserving mapping* on graphs. Given an input graph data, the goal is to learn an edge weight function between each pair of nodes at each network layer, which not only captures the similarity between learned features, but also keeps such similarity consistent with the similarity within the input data. That is, nodes similar in the input data space are also close in the feature space, while nodes dissimilar in the input data space are pushed further away. This is achieved by minimizing our proposed *cross-space Graph Laplacian Regularizer*, which enforces smoothness of the input data with respect to the graph over the feature space and thus leads to a robust graph.

Further, we measure the similarity between nodes by a distance function, and cast graph learning as a distance metric learning problem [7]. We choose the Mahalanobis distance [8] that captures the correlations among features, and assume a *low-rank* model of graph features to decompose the distance metric into a lower-dimensional matrix for optimization, which is able to learn intrinsic representations. Finally, we unify the proposed graph learning and node feature learning in a joint learning framework. To validate the effectiveness of the proposed RGLN, we apply it to semi-supervised learning problems in citation networks and point clouds, where the graph is incomplete in citation networks and unavailable in point clouds. Experimental results demonstrate we outperform state-of-the-art methods on three citation network benchmarks and one point cloud classification benchmark.

2. RELATED WORK

2.1. Graph Convolution Neural Networks

Spectral methods. This class of graph convolution is defined on the spectral representation of graphs. [9] defines the convolution in graph Fourier transform domain, which however requires the eigen-decomposition of the graph Laplacian. [10] addresses this problem by leveraging Chebyshev polynomials to approximate spectral filters and achieves localized

Corresponding author: Wei Hu (forhuwei@pku.edu.cn). This work was supported in part by National Natural Science Foundation of China [61972009] and in part by Beijing Natural Science Foundation [19L2053].

filtering. GCN [11] further simplifies [10] by employing only first-order approximation of the filters. HGNN [12] exploits more flexible hypergraph structure on complex data.

Spatial methods. This class of graph convolution is defined directly on each node and its neighbors for feature propagation. The mixture model network [13] provides a unified generalization of CNN architecture on graphs. Graph attention networks (GATs) [14] employ self-attention mechanism to solve the node classification problem. DGCNN [2] proposes to aggregate local features with the graph updated at each layer, which is applied to point cloud learning.

2.2. Graph Learning in GCNNs

Fixed-domain Graph Learning. This class of graph learning assumes there is a shared graph structure underlying all instances with fixed nodes. GAT [14] leverages masked self-attention layers to enable different weights for edges in the provided graph. GLCN [4] learns a non-negative function that represents the pairwise relationship between two nodes using attention mechanism via a single-layer neural network, and optimizes it by minimizing a graph learning loss. GLNN [15] proposes graph learning from several constraints via maximum a posteriori estimation. [16] statistically learns the underlying graph from multiple observations of spatio-temporal frames for skeleton-based action recognition.

Varying-domain Graph Learning. Instances corresponding to varying domains may require different graphs with possibly arbitrary number of nodes. [16] is proposed to learn a spatio-temporal graph on top of the provided skeleton graph for each skeleton instance. Adaptive graph convolution network (AGCN) [5] proposes to construct a unique residual Laplacian matrix by learning Mahalanobis distance metric for each instance. Different from the task-driven AGCN where the Mahalanobis distance metric $\mathbf{M}$ is learned by minimizing the cross-entropy only, we optimize $\mathbf{M}$ via the task-specific loss as well as a data-adaptive similarity-preserving objective. Further, instead of assuming a general $\mathbf{M}$ in AGCN, we assume $\mathbf{M}$ is low rank to capture intrinsic features and reduce number of parameters.

3. THE PROPOSED RESIDUAL GRAPH LEARNING

The problem of learning a graph is essentially learning each edge weight $w_{i,j}$ for nodes i and j.

The edge weight $w_{i,j}$ often measures the similarity between nodes i and j. Hence, we employ the commonly used Gaussian kernel over features to define an edge weight $w_{i,j}$ as

$$w_{i,j} = \exp\left\{-d^2(\mathbf{f}_i, \mathbf{f}_j)\right\}, \tag{1}$$

where $d(\mathbf{f}_i, \mathbf{f}_j)$ is a distance metric between $\mathbf{f}_i$ and $\mathbf{f}_j$. The smaller the distance is, the more similar $\mathbf{f}_i$ and $\mathbf{f}_j$ are, and the larger the edge weight is. We propose to optimize edge weights $w_{i,j}$'s from the perspective of similarity-preserving mapping on graphs, which will be discussed next.

3.1. Similarity-Preserving Mapping on Graphs

Given an input graph signal $\mathbf{X} \in \mathbb{R}^{N\times d}$ with N nodes and a d-dimensional feature vector on each node, a GCNN learns a function $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^K$, which maps the input graph signal $\mathbf{x}_i \in \mathbb{R}^d$ on node i to a new feature space $\mathbf{f}_i \in \mathbb{R}^K$. The learned features will be used for downstream tasks such as node classification.

A similarity-preserving mapping on graphs keeps the similarity between nodes when mapping nodes from the input space to a new feature space.

Definition 1. Given a graph signal $\mathbf{X} \in \mathbb{R}^{N\times d}$ defined on the vertices $\mathcal{V}$ of a graph $\mathcal{G}$, a mapping $\mathbf{f} : \mathbf{x}_i \in \mathbb{R}^d \to \mathbf{f}_i \in \mathbb{R}^K, i \in \mathcal{V}$ is *similarity-preserving* if it satisfies: 1) nodes with similar input data $\mathbf{x}$ (referred to as "similar nodes" in the sequel) still stay close in the new feature space $\mathbf{f}(\mathbf{x})$; 2) nodes with dissimilar input data $\mathbf{x}$ ("dissimilar nodes") are pushed further away from each other in the new feature space $\mathbf{f}(\mathbf{x})$.

To enforce these conditions in **Definition 1**, we formulate the similarity-preserving objective as

$$\min_{\mathbf{f},d} \sum_{i=1}^{N}\sum_{j=1}^{N} \underbrace{\exp\left\{-d^2(\mathbf{f}_i, \mathbf{f}_j)\right\}}_{w_{i,j}} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2. \tag{2}$$

During the minimization, on one hand, when nodes i and j are similar in the input space, *i.e.*, $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ is small, $w_{i,j}$ could be large, *i.e.*, the distance in the new feature space $d(\mathbf{f}_i, \mathbf{f}_j)$ tends to be small as well. On the other hand, when nodes i and j are dissimilar in the input space, *i.e.*, $\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$ is large, $w_{i,j}$ has to be a small number for minimizing the objective, *i.e.*, the distance in the new feature space $d(\mathbf{f}_i, \mathbf{f}_j)$ will be enforced to be large.

Further, the similarity-preserving objective in (2) can be rewritten as

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{i,j}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 = tr\left(\mathbf{X}^\top \mathbf{L}(\mathbf{f})\mathbf{X}\right), \tag{3}$$

which is a *graph Laplacian regularizer* (GLR) [17] reweighted by the similarity in the feature space $\mathbf{f}$ instead of the input data $\mathbf{X}$. We thus refer to it as the *cross-space GLR*, which will be adopted in the loss function.

3.2. The Proposed Graph Learning

In order to acquire the edge weights in $\mathcal{G}$, we need to learn the distance function $d(\mathbf{f}_i, \mathbf{f}_j)$. We cast the graph learning problem as *distance metric learning* given a feature vector $\mathbf{f}_i \in \mathbb{R}^K$ per node i, *i.e.*, learning a distance function for each pair of features $\{\mathbf{f}_i, \mathbf{f}_j\}$ for similarity calculation.

While there exist various definitions of distance metrics, such as Euclidean distance and bilateral filtering distance, we choose the *Mahalanobis distance* that captures correlations among features. It is defined as

$$d_{\mathbf{M}}(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{(\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{M}(\mathbf{f}_i - \mathbf{f}_j)}, \tag{4}$$

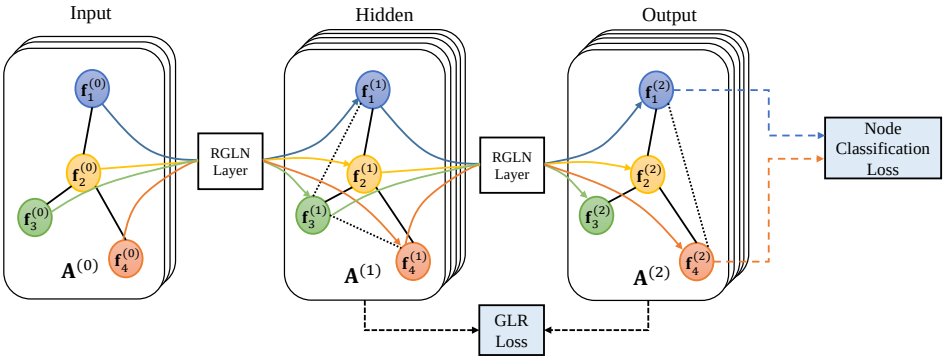

Fig. 1: The proposed RGLN network architecture for node classification and graph classification tasks.

where $\mathbf{M} \in \mathbb{R}^{K\times K}$ is a symmetric and positive semi-definite (PSD) matrix that encodes the feature correlations. Since $\mathbf{M}$ is PSD and symmetric, we can decompose it into $\mathbf{M} = \mathbf{R}\mathbf{R}^\top$, where $\mathbf{R} \in \mathbb{R}^{K\times S}$. Since many real-world graphs naturally have low-rank features, we assume $\mathbf{M}$ is low rank to capture intrinsic representations, and thus $S \ll K$. The number of parameters to learn is also significantly reduced.

Then the Mahalanobis distance in (4) becomes

$$\begin{aligned} d_{\mathbf{M}}(\mathbf{f}_i, \mathbf{f}_j) &= \sqrt{(\mathbf{f}_i - \mathbf{f}_j)^\top \mathbf{R}\mathbf{R}^\top (\mathbf{f}_i - \mathbf{f}_j)} \\ &= \|\mathbf{R}^\top(\mathbf{f}_i - \mathbf{f}_j)\|_2, \end{aligned} \tag{5}$$

which is essentially a linear transformation of the Euclidean distance by $\mathbf{R}$. So each edge weight can be computed as

$$w_{i,j} = \exp\left\{-\|\mathbf{R}^\top(\mathbf{f}_i - \mathbf{f}_j)\|_2^2\right\}. \tag{6}$$

Hence, we convert the graph learning problem to the optimization of a low-rank $\mathbf{R}$, which leads to the following graph learning loss function:

$$\mathcal{L}_{\text{GL}} = \sum_{l=1}^{L} \sum_{\{i,j\}} \exp\left\{-\|\mathbf{R}^{(l)\top}(\mathbf{f}_i^{(l)} - \mathbf{f}_j^{(l)})\|_2^2\right\} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2, \tag{7}$$

where L is the number of graph learning layers and also the number of learned graphs. The superscript (l) denotes a variable at the l-th layer. (7) is convex and differentiable, which is thus a tractable loss in GCNNs. By optimizing the distance metric $\mathbf{R}^{(l)}$ at each layer, we are able to compute the edge weights from (5), leading to the adjacency matrix of a learned residual graph $\mathbf{A}^{*(l)}$.

3.3. Joint Learning of Node Feature and Graph

Among a variety of GCNNs, we choose the GCN [11] as an instance to discuss our node feature learning. The layer-wise graph convolution in the GCN is defined as:

$$\mathbf{F}^{(l+1)} = \sigma\left(\mathbf{\Lambda}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{F}^{(l)}\mathbf{\Theta}^{(l)}\right), \tag{8}$$

where $\mathbf{F}^{(l)} \in \mathbb{R}^{N\times K}$ is the node feature matrix at the l-th layer, $\sigma(\cdot)$ is the activation function, $\mathbf{A}$ is the adjacency matrix of the provided graph, $\mathbf{I}$ is an identity matrix that adds self-loop to the graph, $\mathbf{\Lambda}$ is the degree matrix to normalize $\mathbf{A}+\mathbf{I}$ and $\mathbf{\Theta}^{(l)}$ is the trainable weight matrix at the l-th layer.

In our setting, given the learned residual graph $\mathbf{A}^{*(l)}$ and the available graph $\mathbf{A}^{(l)}$ at the l-th layer, we compute the final graph as

$$\mathbf{A}^{(l+1)} = \mathbf{A}^{(l)} + \mathbf{A}^{*(l)}. \tag{9}$$

Then, we define the convolution as:

$$\mathbf{F}^{(l+1)} = \sigma\left(\tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{A}^{(l+1)}\tilde{\mathbf{\Lambda}}^{-\frac{1}{2}}\mathbf{F}^{(l)}\mathbf{\Theta}^{(l)}\right), \tag{10}$$

where $\tilde{\mathbf{\Lambda}}$ is the updated degree matrix with the i-th diagonal entry computed from $\mathbf{A}^{(l+1)}$ as $\tilde{\mathbf{\Lambda}}_{i,i} = \sum_j a_{i,j}^{(l+1)}$. In cases where there is no initial graph, we assume $\mathbf{A}^{(0)} = \mathbf{I}$ is an identity matrix.

The objective of node feature learning is to minimize the cross entropy between predicted labels and ground truth labels:

$$\mathcal{L}_{\text{GCNN}} = -\sum_{i=1}^{N}\sum_{c=1}^{C} \mathbf{Y}_{ic}\log(\hat{\mathbf{Y}}_{ic}), \tag{11}$$

where N is the number of input instances, C is the number of classes, $\mathbf{Y}$ denotes the ground truth label matrix and $\hat{\mathbf{Y}}$ denotes the predicted probability matrix. Along with the graph learning loss in (7), our network is trained end-to-end by minimizing the following loss function:

$$\mathcal{L} = \mathcal{L}_{\text{GCNN}} + \lambda\mathcal{L}_{\text{GL}}, \tag{12}$$

where λ is a balancing hyper-parameter.

To implement our network, we design a RGLN layer, which consists of two modules: the graph learning module and the node feature learning module. The graph learning module learns a residual graph encoded by an adjacency matrix $\mathbf{A}^*$ from input node features. $\mathbf{A}^*$ is then employed by the node feature learning module to propagate node features via our proposed graph convolution in (10). The entire network architecture is illustrated in Fig. 1.

4. EXPERIMENTAL RESULTS

In order to evaluate the performance of the RGLN, we apply it to semi-supervised node classification on citation networks and point clouds, which are discussed in order as follows.

4.1. Node Classification on Citation Networks

Experimental settings. We test our method on three citation network datasets, *i.e.*, Citeseer, Cora and Pubmed [18]. We follow the experimental setup of previous works [19, 11, 14], and employ the same data partition as in [19]. The features are L_2-normalized before fed into the network. We apply a RGLN layer followed by a regular GCN layer. The number of hidden units in each layer is 16. We set $S = 16$ to reduce the number of parameters. We train our RGLN for a maximum of 500 epochs using ADAM optimizer with a learning rate of 0.1 and weight decay of 5e-4.

Baselines. We compare our RGLN with the baseline of GCN [11], Planetoid [19], and other semi-supervised learning methods based on graph neural networks, including GAT [14], GLCN [4], AGCN [5] and GMNN [20].

Experimental Results. Tab. 1 shows the comparison results on three citation network datasets. Overall, we note that 1) RGLN outperforms the GCN baseline on all datasets significantly by adding a graph learning module. This demonstrates the superiority of learning new edge connectivities and accurate weights. 2) RGLN also outperforms all of the other methods on three datasets, which demonstrates the effectiveness of RGLN in learning graphs.

Algorithm	Citeseer	Cora	Pubmed
Planetoid [19]	64.7	75.7	77.2
GCN [11]	70.3	81.5	79.0
GMNN [20]	73.1	83.7	81.8
GLCN [4]	72.2 ± 0.8	82.1 ± 0.3	77.8 ± 0.3
AGCN [5]	71.1 ± 0.8	83.1 ± 0.4	81.2 ± 0.6
GAT [14]	72.5 ± 0.7	83.0 ± 0.7	79.0 ± 0.3
RGLN	$\mathbf{74.6 \pm 0.5}$	$\mathbf{84.7 \pm 0.3}$	$\mathbf{82.0 \pm 0.4}$

Table 1: Results of semi-supervised node classification.

4.2. Point cloud classification

Furthermore, we test our model on point cloud classification. We follow the experimental setting in [12], using extracted features to represent each point cloud.

Datasets. We test on ModelNet40 dataset [21] for point cloud classification. We extract features of each point cloud using Multi-View Convolutional Neural Network (MVCNN) [22] and Group-View Convolutional Neural Network (GVCNN) [23]. Features from two networks are concatenated together, generating features shaped 12311×7044 as input to GCNNs.

Experimental settings. Since there is no initial graph available in this task, we initialize it as an identity matrix. We stack 3 RGLN layers to learn the residual graphs. The dimension of hidden layers and S are set to 64. We train the network with a learning rate of 0.001 for 150 epochs and decay it by 0.1 at the 100th epoch with ADAM optimizer.

Baselines. We use GCN as the baseline, and empirically construct a k-NN graph as the underlying graph. We mainly compare our RGLN model against the baseline GCN, and also against other state-of-the-art methods, including Hyper Graph Neural Networks (HGNN) [12] and AGCN. We also compare with fully supervised methods for point cloud learning such as PointNet [24], PointNet++ [25] and DGCNN [2].

Experimental Results. Tab. 2 presents results for ModelNet40 classification. We see that, 1) our method outperforms state-of-the-art supervised and semi-supervised methods in the ModelNet40 dataset; 2) compared with GCN which is highly influenced by the hyperparameter k, our RGLN does not need to determine k to build the underlying graph. Instead, RGLN learns an effective graph from node features and achieves improved performance.

Algorithm	Setting	Accuracy
PointNet [24]	Supervised	89.2
PointNet++ [25]	Supervised	90.7
DGCNN [2]	Supervised	93.5
HGNN [12]	Semi-Supervised	96.7
AGCN [5]	Semi-Supervised	95.6
GCN ($k = 1$) [11]	Semi-Supervised	96.5
GCN ($k = 5$) [11]	Semi-Supervised	95.3
GCN ($k = 10$) [11]	Semi-Supervised	94.3
RGLN	Semi-Supervised	**97.1**

Table 2: Results of point cloud classification on ModelNet40.

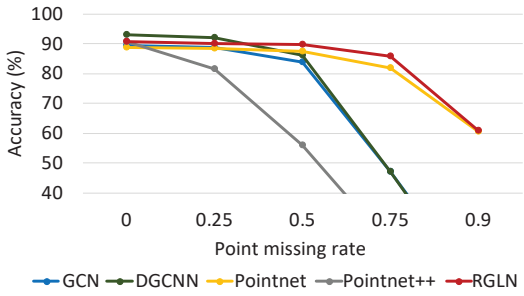

Fig. 2: Robustness test for point clouds.

Robustness Study. Further, we test the robustness of our model when the point cloud is of low density. We randomly drop out points with missing ratios $\{0, 0.25, 0.5, 0.75, 0.9\}$. As shown in Fig. 2, our model outperforms the GCN baseline significantly, and keeps high accuracy even when the point cloud density is quite low.

5. CONCLUSION

We propose a robust residual graph learning network (RGLN), which learns both new edge connectivities and edge weights in the underlying graph. We cast graph learning as distance metric learning under a low-rank assumption. Further, we learn the graph structure by enforcing similarity-preserving mapping via a cross-space graph Laplacian regularizer, which keeps the similarity of nodes consistent between the input data space and the feature space. Extensive experiments demonstrate the superiority and robustness of our method in semi-supervised learning tasks.

6. REFERENCES

[1] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, "Geometric deep learning: going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[2] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, 2019.

[3] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo, "RGCNN: Regularized graph cnn for point cloud segmentation," in *ACM Multimedia Conference on Multimedia Conference*. ACM, 2018, pp. 746–754.

[4] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo, "Semi-supervised learning with graph learning-convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11313–11320.

[5] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang, "Adaptive graph convolutional neural networks," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[6] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu, "Non-local graph convolutional networks for skeleton-based action recognition," *arXiv preprint arXiv:1805.07694*, 2018.

[7] Eric P Xing, Michael I Jordan, Stuart J Russell, and Andrew Y Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[8] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[9] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[11] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[12] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao, "Hypergraph neural networks," *AAAI 2019*, 2018.

[13] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[15] Xiang Gao, Wei Hu, and Zongming Guo, "Exploring structure-adaptive graph learning for robust semi-supervised classification," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[16] Xiang Gao, Wei Hu, Jiaxiang Tang, Jiaying Liu, and Zongming Guo, "Optimized skeleton-based action recognition via sparsified graph regression," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 601–610.

[17] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[18] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[19] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," *arXiv preprint arXiv:1603.08861*, 2016.

[20] Meng Qu, Yoshua Bengio, and Jian Tang, "GMNN: graph markov neural networks," in *International Conference on Machine Learning (ICML)*, 2019, pp. 5241–5250.

[21] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[22] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[23] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao, "GVCNN: Group-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 264–272.

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 652–660.

[25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5099–5108.