Towards Adaptive Information Fusion in Graph Convolutional Networks

Meiqi Zhu, Xiao Wang, *Member, IEEE,* Chuan Shi*, *Member, IEEE*, Yibo Li, Junping Du, *Member, IEEE*

Abstract—Graph Convolutional Networks (GCNs) have gained great popularity in tackling various analytic tasks on graph and network data. However, some recent studies raise concerns about whether GCNs can optimally integrate node features and topological structures in a complex graph. In this paper, we first present an experimental investigation. Surprisingly, our experimental results clearly show that the capability of the state-of-the-art GCNs in fusing node features and topological structures is distant from optimal or even satisfactory. The weakness may severely hinder the capability of GCNs in some classification tasks, since GCNs may not be able to adaptively learn some deep correlation information between topological structures and node features. Can we remedy the weakness and design a new type of GCNs that can retain the advantages of the state-of-the-art GCNs and, at the same time, enhance the capability of fusing topological structures and node features substantially? We tackle the challenge and propose an **A**daptive **M**ulti-channel **G**raph **C**onvolutional **N**etwork for semi-supervised classification (**AM-GCN**). The central idea is that we extract the specific and common embeddings from node features, topological structures, and their combinations simultaneously, and use the attention mechanism to learn adaptive importance weights of the embeddings. However, considering that the input topology and feature structure in AM-GCN are still predefined and fixed, once the properties of graph structures are not consistent with tasks, the fusion performance of AM-GCN will be hindered from the beginning. Therefore, we need to adjust the structure and further propose the **L**abel **P**ropagation guided **M**ulti-channel **G**raph **C**onvolutional **N**etwork (**LPM-GCN**). LPM-GCN introduces edge weights learning on both topology and feature spaces to improve structural homophily, which can better promote the fusion process of graph convolutional networks. Our extensive experiments on benchmark data sets clearly show that our proposed models extract the most correlated information from both node features and topological structures substantially, and improves the classification accuracy with a clear margin.

Index Terms—Graph convolutional networks; Network representation learning; Deep learning

1 Introduction

Network data is ubiquitous, such as social networks, biology networks, complex networks and so on [1], [2], [3] . Recently, Graph Convolutional Networks (GCNs), a class of neural networks designed to learn graph data, have shown great popularity in tackling graph analytic problems, such as node classification [4], [5], graph classification [6], [7], link prediction [8], [9], recommendation [10], [11], [12].

The typical GCN [13] and its variants [8], [14], [15], [16], [17] usually follow a message-passing manner. A key step is feature aggregation, i.e., a node aggregates feature information from its topological neighbors in each convolutional layer. In this way, feature information propagates over network topology to node embedding, and then node embedding learned as such is used in classification tasks. The whole process is supervised partially by the node labels. The enormous success of GCN is partially thanks to that GCN provides a fusion strategy on topological structures and node features to learn node embedding, and the fusion process is supervised by an end-to-end learning framework.

Some recent studies, however, disclose certain weaknesses of the state-of-the-art GCNs in fusing node features and topological structures. For example, Li *et al.* [18] show that GCNs actually perform the Laplacian smoothing on node features, and make the node embedding in the whole network gradually converge. Nt *et al.* [19] and Wu *et al.* [17] prove that topological structures play the role of low-pass filtering on node features when the feature information propagates over network topological structure.

What information do GCNs really learn and fuse from topological structures and node features? This is a fundamental question since GCNs are often used as an end-to-end learning framework. A well-informed answer to this question can help us understand the capability and limitations of GCNs in a principled way. This motivates our study immediately.

As the first contribution of this study, we present experiments assessing the capability of GCNs in fusing topological structures and node features. Surprisingly, our experiments clearly show that the fusion capability of GCNs on network topological structures and node features is clearly distant from optimal or even satisfactory. Even under some simple situations that the correlation between node features/topology with node label is very clear, GCNs still cannot adequately fuse node feature and topological structure to extract the most correlated information (shown in Section 2). The weakness may severely hinder the capability of GCNs in some classification tasks, since GCNs may not be able to adaptively learn some correlation information between topological structures and node features.

Once the weakness of the state-of-the-art GCNs in the fusion mechanism is identified, a natural question is, "*Can we remedy the weakness and design a new type of GCNs that can retain the advantages of the state-of-the-art GCNs and, at the same time, enhance the capability of fusing topological structures and node features substantially?*" A good fusion capability

• *Meiqi Zhu, Xiao Wang, Chuan Shi (corresponding author), Yibo Li, Junping Du are with Beijing University of Posts and Telecommunications, China. E-mail: {zhumeiqi, xiaowang, shichuan}@bupt.edu.cn, liushiliushi0@gmail.com, junpingdu@126.com. Meiqi Zhu is also with Ant Group. E-mail: zhumeiqi.zmq@antgroup.com*

of GCNs should substantially extract and fuse the most correlated information for the classification task, however, one biggest obstacle in reality is that the correlation between network data and classification task is usually very complex and agnostic. The classification can be correlated with either the topology, or node features, or their combinations.

This paper tackles the challenge and first proposes an **A**daptive **M**ulti-channel **G**raph **C**onvolutional **N**etworks (**AM-GCN**) for semi-supervised classification. The central idea is that we learn the node embedding based on node features, topological structures, and their combinations simultaneously. The rationale is that the similarity between features and that inferred by topological structures are complementary to each other and can be fused adaptively to derive deeper correlation information for classification tasks. Technically, in order to fully exploit the information in feature space, we derive the *k*-nearest neighbor graph generated from node features as the feature structural graph. With the feature graph and the topology graph, we propagate node features over both topology space and feature space, so as to extract two specific embeddings in these two spaces with two specific convolution modules. Considering the common characteristics between two spaces, we design a common convolution module with a parameter sharing strategy to extract the common embedding shared by them. We further utilize the attention mechanism to automatically learn the importance weights for different embeddings, so as to adaptively fuse them. Moreover, we design the consistency and disparity constraints to ensure the consistency and disparity of the learned embeddings.

AM-GCN is able to flexibly propagate node features over topology and *k*NN graph structures. However, both of these two structures are also essentially defined independently from the final task, e.g., node classification. Once the properties of graph structures are not consistent with node label, it will fundamentally hinder the performance. It is well recognized that a graph with good homophily or community structure, i.e., nodes within the same class/label connect more strongly with each other, will be more optimal for GCN models [20], [21]. The challenge is that the input graph structure is usually predefined and then fixed, while we need to adjust the structure to further improve the homophily, so as to better adapt the property of GCNs. Thus, we further introduce label propagation guided edge weights learning on both topology and feature graphs. The label propagation algorithm can directly help adjust the edge weights using task labels as supervision, and further help improve the corresponding structural homophily and intra-class consistency. Thus the extended new model can better promote the fusion process with learned graph structures, referring to **L**abel **P**ropagation Guided **M**ulti-channel **G**raph **C**onvolutional **N**etworks (**LPM-GCN**). LPM-GCN is also in an end-to-end fashion that unifies the learning process of AM-GCN and LP for node classification, and LP serves as a regularization to assist AM-GCN in learning proper edge weights.

Our contributions are highlighted as follows:

- We present experiments assessing the capability of GCNs in fusing topological structures and node features and identify the weakness of GCN. We further study the important problem, i.e., how to substantially enhance the fusion capability of GCN for classification.
- We propose a novel adaptive multi-channel GCN framework, AM-GCN, which performs graph convolution operation over both topology and feature spaces. Combined with the attention mechanism, different information can be adequately fused.
- We further introduce the label propagation guided edge weights learning to improve the intra-class consistency as the extended model LPM-GCN. Both homophily of the topology and feature graphs will be improved, which further promotes the fusion process.
- Our extensive experiments on a series of benchmark data sets clearly show that both AM-GCN and LPM-GCN outperform the state-of-the-art GCNs and extract the most correlation information from both node features and topological structures for classification tasks.

Please note that the preliminary work has been accepted as a full paper at the research track of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining [22]. Based on the conference paper, we substantially extend the original work from the following aspects: 1) We propose a new LP guided graph structure learning framework named LPM-GCN in this maniscript, which adaptively learns the input edge weights for AM-GCN according to labels for tasks. By changing edge weights with LPM-GCN, the intra-class consistency will increase correspondingly, and is further beneficial to the fusion process of graph convolutional networks. 2) We further provide detailed justifications of the AM-GCN and the extended model LPM-GCN. We first conduct case studies on two cases in Section 2 using GAT and GraphSAGE, which confirms that the fusion problem also exists in other GNN models. We then design the LPM-GCN with a detailed algorithm. Furthermore, we analyze the computational complexity and number of parameters for AM-GCN and LPM-GCN in Section 4. 3) To further verify the effectiveness of the proposed models, we also significantly enrich experiments. Concretely, we first conduct experiments using LPM-GCN to analyze the learned edge weights. Then we compare the effectiveness of using cosine similarity or heat kernel to construct the feature graph. Besides, we conduct an ablation study to analyze the effectiveness of combining both the $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$ on both AM-GCN and LPM-GCN with three datasets. As for main node classification experiments, we add two new baselines (i.e., LP and MLP), one new dataset (i.e., ACM-PSP). It is worth noting that we rerun all the analysis tasks using LPM-GCN, which confirms the superiority of both LPM-GCN and AM-GCN. 4) We enrich our related work including graph neural networks, graph structural learning, attention mechanism. We also carefully polish our paper to improve the language quality.

2 Fusion Capability of GCNs: An Experimental Investigation

In this section, we use two simple yet intuitive cases to examine whether the state-of-the-art GCNs can adaptively learn from node features and topological structures in graphs and fuse them sufficiently for classification tasks. The main idea is that we will clearly establish the high

correlation between node label with network topology and node features, respectively, then we will check the performance of GCN on these two simple cases. A good fusion capability of GCN should adaptively extract the correlated information with the supervision of node label, providing a good result. However, if the performance drops sharply in comparison with baselines, this will demonstrate that GCN cannot adaptively extract information from node features and topological structures, even there is a high correlation between node features or topological structures with the node label.

2.1 Case 1: Random Topology and Correlated Node Features

We generate a random network consists of 900 nodes, where the probability of building an edge between any two nodes is 0.03. Each node has a feature vector of 50 dimensions. To generate node features, we randomly assign 3 labels to the 900 nodes, and for the nodes with the same label, we use one Gaussian distribution to generate the node features. The Gaussian distributions for the three classes of nodes have the same covariance matrix, but three different centers far away from each other. In this dataset, the node labels are highly correlated with the node features, but not the topological structures.

We apply GCN [13] to train this network. For each class, we randomly select 20 nodes for training and another 200 nodes for testing. We carefully tune the hyper-parameters to report the best performance and avoid over smoothing. Also, we apply MLP [23] to the node features only. The classification accuracies of GCN and MLP are 75.2% and 100%, respectively. Furthermore, we also conduct experiments on GAT [14] and GraphSAGE [15] with same settings, and the calssification accuracies of GAT and GraphSAGE are 72.3% and 86.7%.

These results meet the expectation. Since the node features are highly correlated with the node labels, MLP shows excellent performance. GCN extracts information from both the node features and the topological structures, but cannot adaptively fuse them to avoid interference from topological structures. It cannot match the high performance of MLP. As for GAT and GraphSAGE, the designed attention mechanism and sample strategies will increase the convolutional flexibility, but they also need to solve the fusion problem.

2.2 Case 2: Correlated Topology and Random Node Features

We generate another network with 900 nodes. This time, the node features, each of 50 dimensions, are randomly generated. For the topological structure, we employ the Stochastic Blockmodel (SBM) [24] to split nodes into 3 communities (nodes 0-299, 300-599, 600-899, respectively). Within each community, the probability of building an edge is set to 0.03, and the probability of building an edge between nodes in different communities is set to 0.0015. In this data set, the node labels are determined by the communities, i.e., nodes in the same community have the same label.

Again we apply GCN to this network. We also apply DeepWalk [25] to the topology of the network, that is, the features are ignored by DeepWalk. The classification accuracies of GCN and DeepWalk are 87% and 100%, respectively. And GAT and GraphSAGE in Case 2 are 78.5% and 83.5%,respectively DeepWalk performs well because it models network topological structures thoroughly. GCN extracts information from both the node features and the topological structures, but cannot adaptively fuse them to avoid interference from node features. Similarly with GAT and GraphSAGE, they all cannot match the high performance of DeepWalk.

Summary. These cases show that the current fusion mechanism of GCN [13] is distant from optimal or even satisfactory. Even the correlation between node label with network topology or node features is very high, the current GCN cannot make full use of the supervision by node label to adaptively extract the most correlated information. However, the situation is more complex in reality, because it is hard to know whether the topology or the node features are more correlated with the final task, which prompts us to rethink the current mechanism of GCN.

3 AM-GCN: THE PROPOSED MODEL

Problem Settings: We focus on semi-supervised node classification in an attributed graph $G = (\mathbf{A}, \mathbf{X})$, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the symmetric adjacency matrix with n nodes and $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the node feature matrix, and d is the dimension of node features. Specifically, $A_{ij} = 1$ represents there is an edge between nodes i and j, otherwise, $A_{ij} = 0$. We suppose each node belongs to one out of C classes.

The overall framework of AM-GCN is shown in Figure 1. The key idea is that AM-GCN permits node features to propagate not only in topology space, but also in feature space, and the most correlated information with node label should be extracted from both of these two spaces. To this end, we construct a feature graph based on node features $\mathbf{X}$. Then with two specific convolution modules, $\mathbf{X}$ is able to propagate over both of feature graph and topology graph to learn two specific embeddings $\mathbf{Z}_F$ and $\mathbf{Z}_T$, respectively. Further, considering that the information in these two spaces has common characteristics, we design a common convolution module with parameter sharing strategy to learn the common embedding $\mathbf{Z}_{CF}$ and $\mathbf{Z}_{CT}$, also, a consistency constraint $\mathcal{L}_c$ is employed to enhance the "common" property of $\mathbf{Z}_{CF}$ and $\mathbf{Z}_{CT}$. Besides, a disparity constraint $\mathcal{L}_d$ is to ensure the independence between $\mathbf{Z}_F$ and $\mathbf{Z}_{CF}$, as well as $\mathbf{Z}_T$ and $\mathbf{Z}_{CT}$. Considering that node label may be correlated with topology or feature or both, AM-GCN utilizes an attention mechanism to adaptively fuse these embeddings with the learned weights, so as to extract the most correlated information $\mathbf{Z}$ for the final task.

3.1 Specific Convolution Module

Firstly, in order to capture the underlying structure of nodes in feature space, we construct a k-nearest neighbor (kNN) graph $G_f = (\mathbf{A}_f, \mathbf{X})$ based on node feature matrix $\mathbf{X}$, where $\mathbf{A}_f$ is the adjacency matrix of kNN graph. Specifically, we first calculate the similarity matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ among n nodes. Actually, there are many ways to obtain $\mathbf{S}$, and we list two popular ones here, in which $\mathbf{x}_i$ and $\mathbf{x}_j$ are feature vectors of nodes i and j:

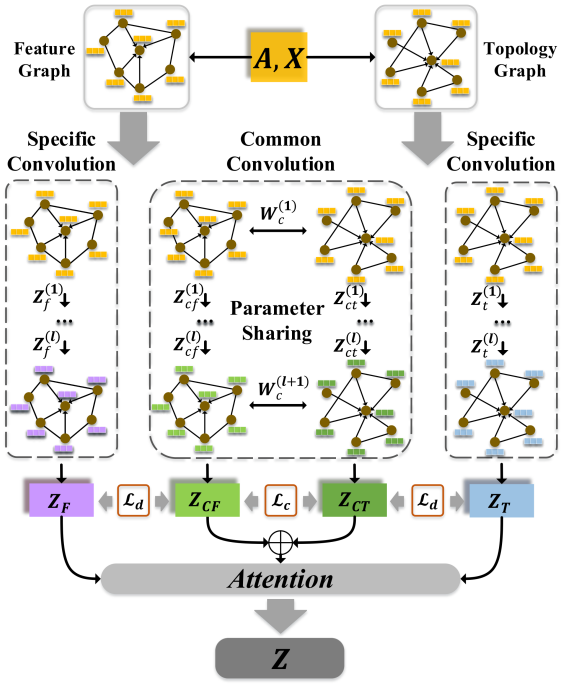


Fig. 1: The framework for AM-GCN, including two ***Specific Convolution Modules***, one ***Common Convolution Module*** and the ***Attention Mechanism*** for adaptively fuse most correlated information for classification.

1) **Cosine Similarity**: It uses the cosine value of the angle between two vectors to measure the similarity:

$$\mathbf{S}_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i||\mathbf{x}_j|}. \tag{1}$$

2) **Heat Kernel**: The similarity is calculated by the Eq. (2) where t is the time parameter in heat conduction equation and we set $t = 2$.

$$\mathbf{S}_{ij} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{t}}. \tag{2}$$

Here we uniformly choose the Cosine Similarity to obtain the similarity matrix $\mathbf{S}$, and then we choose top k similar node pairs for each node to set edges and finally get the adjacency matrix $\mathbf{A}_f$.

Then with the input graph $(\mathbf{A}_f, \mathbf{X})$ in feature space, the l-th layer output $\mathbf{Z}_f^{(l)}$ can be represented as:

$$\mathbf{Z}_f^{(l)} = ReLU(\tilde{\mathbf{D}}_f^{-\frac{1}{2}}\tilde{\mathbf{A}}_f\tilde{\mathbf{D}}_f^{-\frac{1}{2}}\mathbf{Z}_f^{(l\text{-}1)}\mathbf{W}_f^{(l)}), \tag{3}$$

where $\mathbf{W}_f^{(l)}$ is the weight matrix of the l-th layer in GCN, $ReLU$ is the Relu activation function and the initial $\mathbf{Z}_f^{(0)} = \mathbf{X}$. Specifically, we have $\tilde{\mathbf{A}}_f = \mathbf{A}_f + \mathbf{I}_f$ and $\tilde{\mathbf{D}}_f$ is the diagonal degree matrix of $\tilde{\mathbf{A}}_f$. We denote the last layer output embedding as $\mathbf{Z}_F$. In this way, we can learn the node embedding which captures the specific information $\mathbf{Z}_F$ in feature space.

As for the topology space, we have the original input graph $G_t = (\mathbf{A}_t, \mathbf{X}_t)$ where $\mathbf{A}_t = \mathbf{A}$ and $\mathbf{X}_t = \mathbf{X}$. Then the learned output embedding $\mathbf{Z}_T$ based on topology graph can be calculated in the same way as in feature space. Therefore, the specific information encoded in topology space can be extracted.

3.2 Common Convolution Module

In reality, the feature and topology spaces are not completely irrelevant. Basically, the node classification task may be correlated with the information either in feature space or in topology space or in both of them, which is difficult to know beforehand. Therefore, we not only need to extract the specific embedding in these two spaces, but also to extract the common information shared by the two spaces. In this way, it will become more flexible for the task to determine which part of information is the most correlated. To address this, we design a *Common*-GCN with parameter sharing strategy to get the embedding shared in two spaces.

First, we utilize *Common*-GCN to extract the node embedding $\mathbf{Z}_{ct}^{(l)}$ from topology graph $(\mathbf{A}_t, \mathbf{X})$ as follows:

$$\mathbf{Z}_{ct}^{(l)} = ReLU(\tilde{\mathbf{D}}_t^{-\frac{1}{2}}\tilde{\mathbf{A}}_t\tilde{\mathbf{D}}_t^{-\frac{1}{2}}\mathbf{Z}_{ct}^{(l\text{-}1)}\mathbf{W}_c^{(l)}), \tag{4}$$

where $\mathbf{W}_c^{(l)}$ is the l-th layer weight matrix of *Common*-GCN and $\mathbf{Z}_{ct}^{(l\text{-}1)}$ is the node embedding in the $(l-1)$th layer and $\mathbf{Z}_{ct}^{(0)} = \mathbf{X}$. When utilizing *Common*-GCN to learn the node embedding from feature graph $(\mathbf{A}_f, \mathbf{X})$, in order to extract the shared information, we share the same weight matrix $\mathbf{W}_c^{(l)}$ for every layer of *Common*-GCN as follows:

$$\mathbf{Z}_{cf}^{(l)} = ReLU(\tilde{\mathbf{D}}_f^{-\frac{1}{2}}\tilde{\mathbf{A}}_f\tilde{\mathbf{D}}_f^{-\frac{1}{2}}\mathbf{Z}_{cf}^{(l\text{-}1)}\mathbf{W}_c^{(l)}), \tag{5}$$

where $\mathbf{Z}_{cf}^{(l)}$ is the l-layer output embedding and $\mathbf{Z}_{cf}^{(0)} = \mathbf{X}$. The shared weight matrix can filter out the shared characteristics from two spaces. According to different input graphs, we can get two output embedding $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$ and the common embedding $\mathbf{Z}_C$ of the two spaces is:

$$\mathbf{Z}_C = (\mathbf{Z}_{CT} + \mathbf{Z}_{CF})/2. \tag{6}$$

3.3 Attention Mechanism

Now we have two specific embeddings $\mathbf{Z}_T$ and $\mathbf{Z}_F$, and one common embedding $\mathbf{Z}_C$. Considering the node label can be correlated with one of them or even their combinations, we use the attention mechanism $att(\mathbf{Z}_T, \mathbf{Z}_C, \mathbf{Z}_F)$ to learn their corresponding importance $(\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_c, \boldsymbol{\alpha}_f)$ as follows:

$$(\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_c, \boldsymbol{\alpha}_f) = att(\mathbf{Z}_T, \mathbf{Z}_C, \mathbf{Z}_F), \tag{7}$$

here $\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_c, \boldsymbol{\alpha}_f \in \mathbb{R}^{n\times 1}$ indicate the attention values of n nodes with embeddings $\mathbf{Z}_T, \mathbf{Z}_C, \mathbf{Z}_F$, respectively.

Here we focus on node i, where its embedding in $\mathbf{Z}_T$ is $\mathbf{z}_T^i \in \mathbb{R}^{1\times h}$ (i.e., the i-th row of $\mathbf{Z}_T$). We firstly transform the embedding through a nonlinear transformation, and then use one shared attention vector $\mathbf{q} \in \mathbb{R}^{h'\times 1}$ to get the attention value ω_T^i as follows:

$$\omega_T^i = \mathbf{q}^T \cdot tanh(\mathbf{W}_T \cdot (\mathbf{z}_T^i)^T + \mathbf{b}_T). \tag{8}$$

Here $\mathbf{W}_T \in \mathbb{R}^{h'\times h}$ is the weight matrix and $\mathbf{b}_T \in \mathbb{R}^{h'\times 1}$ is the bias vector for embedding matrix $\mathbf{Z}_T$, respectively. Similarly, we can get the attention values ω_C^i and ω_F^i for node i in embedding matrices $\mathbf{Z}_C$ and $\mathbf{Z}_F$, respectively. We then normalize the attention values $\omega_T^i, \omega_C^i, \omega_F^i$ with softmax function to get the final weight:

$$\alpha_T^i = softmax(\omega_T^i) = \frac{exp(\omega_T^i)}{exp(\omega_T^i) + exp(\omega_C^i) + exp(\omega_F^i)}. \tag{9}$$

Larger α_T^i implies the corresponding embedding is more important. Similarly, $\alpha_C^i = softmax(\omega_C^i)$ and $\alpha_F^i = softmax(\omega_F^i)$. For all the n nodes, we have the learned weights $\boldsymbol{\alpha}_t = [\alpha_T^i], \boldsymbol{\alpha}_c = [\alpha_C^i], \boldsymbol{\alpha}_f = [\alpha_F^i] \in \mathbb{R}^{n\times 1}$, and denote $\boldsymbol{\alpha_T} = diag(\boldsymbol{\alpha}_t)$, $\boldsymbol{\alpha_C} = diag(\boldsymbol{\alpha}_c)$ and $\boldsymbol{\alpha_F} = diag(\boldsymbol{\alpha}_f)$. Then we combine these three embeddings to obtain the final embedding $\mathbf{Z}$:

$$\mathbf{Z} = \boldsymbol{\alpha_T} \cdot \mathbf{Z}_T + \boldsymbol{\alpha_C} \cdot \mathbf{Z}_C + \boldsymbol{\alpha_F} \cdot \mathbf{Z}_F. \tag{10}$$

Actually, the designed attention mechanism aims to learn adaptive importance weights for embeddings. Some prior works also perform adaptive selection for positive samples [26]. However, our designed adaptive fusion is more beneficial for extracting effective and useful information from both specific and common convolution process with feature graph and topology graph as input.

3.4 Consistency and Disparity Constraints

3.4.1 *Consistency Constraint*

For the two output embeddings $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$ of *Common*-GCN, despite the *Common*-GCN has the shared weight matrix, here we design a consistency constraint to further enhance their commonality.

Firstly, we use L_2-normalization to normalize the embedding matrix as $\mathbf{Z}_{CTnor}$, $\mathbf{Z}_{CFnor}$. Then, the two normalized matrices can be used to capture the similarity of n nodes as $\mathbf{S}_T$ and $\mathbf{S}_F$ as follows:

$$\mathbf{S}_T = \mathbf{Z}_{CTnor} \cdot \mathbf{Z}_{CTnor}^T, \quad \mathbf{S}_F = \mathbf{Z}_{CFnor} \cdot \mathbf{Z}_{CFnor}^T. \tag{11}$$

The consistency implies that the two similarity matrices should be similar, which gives rise to the following constraint:

$$\mathcal{L}_c = \|\mathbf{S}_T - \mathbf{S}_F\|_F^2. \tag{12}$$

3.4.2 *Disparity Constraint*

Here because embeddings $\mathbf{Z}_T$ and $\mathbf{Z}_{CT}$ are learned from the same graph $G_t = (\mathbf{A}_t, \mathbf{X}_t)$, to ensure they can capture different information, we employ the Hilbert-Schmidt Independence Criterion (HSIC) [27], a simple but effective measure of independence, to enhance the disparity of these two embeddings. Due to its simplicity and neat theoretical properties, HSIC has been applied to several machine learning tasks [28], [29]. Formally, the HSIC constraint of $\mathbf{Z}_T$ and $\mathbf{Z}_{CT}$ is defined as:

$$HSIC(\mathbf{Z}_T, \mathbf{Z}_{CT}) = (n-1)^{-2} tr(\mathbf{R}\mathbf{K}_T\mathbf{R}\mathbf{K}_{CT}), \tag{13}$$

where $\mathbf{K}_T$ and $\mathbf{K}_{CT}$ are the Gram matrices with $k_{T,ij} = k_T(\mathbf{z}_T^i, \mathbf{z}_T^j)$ and $k_{CT,ij} = k_{CT}(\mathbf{z}_{CT}^i, \mathbf{z}_{CT}^j)$. And $\mathbf{R} = \mathbf{I} - \frac{1}{n}ee^T$, where $\mathbf{I}$ is an identity matrix and e is an all-one column vector. In our implementation, we use the inner product kernel function for $\mathbf{K}_T$ and $\mathbf{K}_{CT}$.

Similarly, considering the embeddings $\mathbf{Z}_F$ and $\mathbf{Z}_{CF}$ are also learned from the same graph $(\mathbf{A}_f, \mathbf{X})$, their disparity should also be enhanced by HSIC:

$$HSIC(\mathbf{Z}_F, \mathbf{Z}_{CF}) = (n-1)^{-2} tr(\mathbf{R}\mathbf{K}_F\mathbf{R}\mathbf{K}_{CF}). \tag{14}$$

Then we set the disparity constraint as $\mathcal{L}_d$ where:

$$\mathcal{L}_d = HSIC(\mathbf{Z}_T, \mathbf{Z}_{CT}) + HSIC(\mathbf{Z}_F, \mathbf{Z}_{CF}). \tag{15}$$

3.5 Optimization Objective for AM-GCN

We use the output embedding $\mathbf{Z}$ in Eq. (10) for semi-supervised multi-class classification with a linear transformation and a softmax function. Denote the class predictions for n nodes as $\hat{\mathbf{Y}} = [\hat{y}_{ic}] \in \mathbb{R}^{n\times C}$ where $\hat{y}_{ic}$ is the probability of node i belonging to class c. Then the $\hat{\mathbf{Y}}$ can be calculated in the following way:

$$\hat{\mathbf{Y}} = softmax(\mathbf{W} \cdot \mathbf{Z} + \mathbf{b}), \tag{16}$$

where $softmax(x) = \frac{\exp(x)}{\Sigma_{c=1}^C exp(x_c)}$ is actually a normalizer across all classes.

Suppose the labeled training set is L, for each $l \in L$ the real label is $\mathbf{Y}_l$ and the predicted label is $\hat{\mathbf{Y}}_l$. Then the cross-entropy loss for node classification over all training nodes is represented as $\mathcal{L}_t$ where:

$$\mathcal{L}_t = -\sum_{l\in L}\sum_{i=1}^{C} \mathbf{Y}_{li} \ln \hat{\mathbf{Y}}_{li}. \tag{17}$$

Combining the node classification task and constraints, we have the following overall objective function:

$$\mathcal{L}_{AM-GCN} = \mathcal{L}_t + \gamma\mathcal{L}_c + \beta\mathcal{L}_d, \tag{18}$$

where γ and β are parameters of the consistency and disparity constraint terms. With the guide of labeled data, we can optimize the proposed model via back propagation and learn the embedding of nodes for classification.

4 LPM-GCN: The Extended Model

The proposed AM-GCN is actually a task-oriented framework, where the corresponding fusing mechanism is driven by the consistency between feature/topology space with labels. However, AM-GCN still works on the predefined graph. While, such fixed graph structure inevitably contains incorrect or noisy edges, which are not beneficial to the downstream task. Therefore, a more practical GCN model should be able to not only learn good node representations, but also adjust the graph structure, so as to make the graph more optimal for the node classification task. Therefore, we further use an edge weights learning module to introduce learnable and soft graphs on both topology and feature spaces. Using the label propagation process as supervision, we tend to learn graphs whose edge weights are consistent with the distribution of labels and beneficial for tasks. Then we propose the extended **L**abel **P**ropagation guided **M**ulti-channel **G**raph **C**onvolutional **N**etworks (**LPM-GCN**) model in this section, where the framework of LPM-GCN is in Figure 2.

4.1 Label Guided Edge Weights Learning

Here, we aim to learn edge weights based on label supervision, without adding additional edges into original structure. Specifically, we use learnable mask matrices $\mathbf{M}_t$ and $\mathbf{M}_f$ to adjust the edge weights of topology graph $\tilde{\mathbf{A}}_t$ and feature graph $\tilde{\mathbf{A}}_f$. With the adjacency matrices $\tilde{\mathbf{A}}_*, * \in \{t, f\}$ as input, we have the non-negative masked adjacency matrices $\tilde{\mathbf{A}'}_*$ in the following ways:

$$\tilde{\mathbf{A}'}_* = \mathbf{M}_* \circ \mathbf{M}_* \circ \tilde{\mathbf{A}}_*, * \in \{t, f\} \tag{19}$$

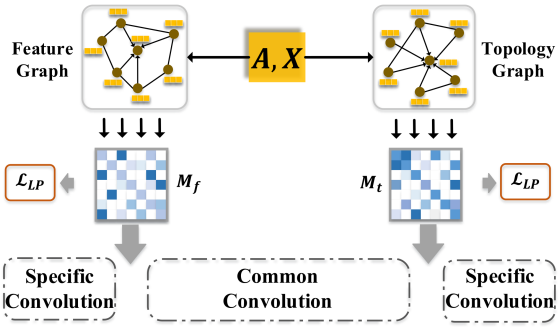


Fig. 2: The label propagation guided edge weights learning module for LPM-GCN. $\mathbf{M}_F$ and $\mathbf{M}_T$ are two learnable mask matrices under the supervision of label propagation.

where $\circ$ is the Hadamard product. Note that $\mathbf{M}_t$ and $\mathbf{M}_f$ are initialized as $\tilde{\mathbf{A}}_t$ and $\tilde{\mathbf{A}}_f$, respectively.

Using the masked adjacency matrices $\tilde{\mathbf{A}'}_*$, we then introduce the label as supervision to guide the edge weights learning using label propagation (LP) [30], [31], [32]. Label propagation algorithms propagate node labels along edges while assume that two connected nodes are likely to have same labels. For both topology graph and feature graph with partial labeled nodes, we have the same label matrix $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_n]^T$ where $\mathbf{y}_n \in \mathbb{R}^{1\times C}$ is the one-hot label vector (i.e., nodes in labeled set $\mathbf{L}$) or zero vector otherwise (i.e., nodes in unlabeled set $\mathbf{U}$). Then label propagation process at the kth ($k > 0$) iteration with the masked graph $\tilde{\mathbf{A}'}_*$ is formulated as the following two steps:

$$\mathbf{Y}_*^{(k)} = {\tilde{\mathbf{D}}_*}^{-1}\tilde{\mathbf{A}'}_*\mathbf{Y}_*^{(k-1)}, * \in \{t, f\} \tag{20}$$

$${\mathbf{Y}_*^{(k)}}_m = {\mathbf{Y}_*^{(0)}}_m, \forall m \in \mathbf{L}, \tag{21}$$

where $\mathbf{Y}_*^{(0)} = \mathbf{Y}$. Firstly, the label information is propagated along the normalized adjacency in Eq. (20). Then the labeled nodes should be reset to persist the initial label information in Eq. (21) and avoid the overpowering influence of unlabeled nodes. After K iterations, the final predicted label matrices $\mathbf{Y}_*^{(K)}$ are generated based on the corresponding graph.

Then the masked adjacency matrices $\tilde{\mathbf{A}'}_t$ and $\tilde{\mathbf{A}'}_f$ can be optimized through minimizing the difference between ground-truth labels and labels reconstructed from multi-hop neighbors:

$$\mathcal{L}_{LP}(\tilde{\mathbf{A}'}_*) = -\sum_{l\in\mathbf{L}}\sum_{i=1}^{C}{\mathbf{Y}_*^{(0)}}_{li}\ln{\mathbf{Y}_*^{(K)}}_{li}, * \in \{t, f\} \tag{22}$$

where $\mathbf{Y}_*^{(0)} = \mathbf{Y}$, L is the labeled set, and $\mathbf{M}_*$ are the learnable variables. With the above optimization objective, the model will increase the weight of edges between nodes of the same label. In this way, the potential intra-class edges will be identified and the intra-class consistency of topology and feature graph can be improved correspondingly.

After using Eq. (22) to optimal both the $\tilde{\mathbf{A}}_t$ and $\tilde{\mathbf{A}}_f$, we can get the new adjacency matrices as $\tilde{\mathbf{A}''}_*, * \in \{t, f\}$:

$$\tilde{\mathbf{A}''}_* = \underset{\tilde{\mathbf{A}'}_*}{argmin}\,\mathcal{L}_{LP}(\tilde{\mathbf{A}'}_*), * \in \{t, f\}. \tag{23}$$

Algorithm 1 LPM-GCN

Input: Node features $\mathbf{X}$; Topological adjacency with self-loop $\tilde{\mathbf{A}}_t$; Label matrix $\mathbf{Y}$;
Parameter: k, γ, β, λ, iteration number K for LP;
Output: Predicted label matrix $\hat{\mathbf{Y}}$;

```
//Initialization for feature graph
Ã_f ← kNN(X, k) + I using Eq. (1);
//Initialization for mask matrices
M_f ← Ã_f; M_t ← Ã_t
while Stopping condition is not meet do
    Ã'_* ← M_* ∘ M_* ∘ Ã_*, * ∈ {t, f};
    // Using label propagation algorithm in Eq. (20)-(21)
    Y_*^(K) ← LabelPropagation(Ã'_*, Y, K), * ∈ {t, f};
    // Using proposed AM-GCN in Section 3
    Ŷ, Z_emb ← AM-GCN(Ã'_*, X), * ∈ {t, f};
    //Z_emb ∈ {Z_T, Z_F, Z_CF, Z_CT};
    L_t ← {Ŷ, Y} using Eq. (17);
    L_c ← {Z_CF, Z_CT} using Eq. (12);
    L_d ← {Z_T, Z_F} using Eq. (15);
    L_LP ← {Y_*^(K), Y} using Eq. (22);
    L_LPM−GCN ← L_t + γL_c + βL_d + λL_LP;
    Back-propagate L_LPM−GCN to update M_* and W'_set, * ∈ {t, f} ;
end while
return Ŷ.
```

Then we can apply the learned $\tilde{\mathbf{A}''}_*$ to the original AM-GCN framework in Section 3 and further adaptively learn label consistency and fuse the most useful information from topology and feature spaces. The updated graphs with larger intra-class label consistency will be beneficial to AM-GCN. Specifically, the l-th layer GCN output of specific convolution module in Eq. (3) change to:

$$\mathbf{Z}_*^{(l)} = ReLU({\tilde{\mathbf{D}''}_*}^{-\frac{1}{2}}\tilde{\mathbf{A}''}_*{\tilde{\mathbf{D}''}_*}^{-\frac{1}{2}}\mathbf{Z}_*^{(l\text{-}1)}\mathbf{W}_*^{(l)}), * \in \{t, f\}. \tag{24}$$

And the l-th layer GCN output of common convolution module in Eq.(4) and Eq.(5) change to:

$$\mathbf{Z}_{c*}^{(l)} = ReLU({\tilde{\mathbf{D}''}_*}^{-\frac{1}{2}}\tilde{\mathbf{A}''}_*{\tilde{\mathbf{D}''}_*}^{-\frac{1}{2}}\mathbf{Z}_{c*}^{(l\text{-}1)}\mathbf{W}_c^{(l)}), * \in \{t, f\}. \tag{25}$$

Using the same optimization objective with AM-GCN in Eq. (18), we can get the optimal parameter sets for AM-GCN including all corresponding parameters, and here we mark them as $\mathbf{W}'_{set}$:

$$\mathbf{W}'_{set} = \underset{\mathbf{W}_{set}}{argmin}\,\mathcal{L}_{AM-GCN}(\mathbf{W}_{set}, \tilde{\mathbf{A}''}_t, \tilde{\mathbf{A}''}_f). \tag{26}$$

Actually, the above two main steps including edge weights learning and parameters optimization are separate. However, instead of training and adjusting parameters separately [33], it is empirically better and more elegant to combine these two steps together and train the model in an end-to-end mode in practice:

$$\begin{aligned}\mathbf{W}'_{set}, \tilde{\mathbf{A}''}_t, \tilde{\mathbf{A}''}_f = \underset{\mathbf{W}_{set}, \tilde{\mathbf{A}'}_t, \tilde{\mathbf{A}'}_f}{argmin} \Big\{&\mathcal{L}_{AM-GCN}(\mathbf{W}_{set}, \tilde{\mathbf{A}'}_t, \tilde{\mathbf{A}'}_f) \\ &+ \lambda(\mathcal{L}_{LP}(\tilde{\mathbf{A}'}_t) + \mathcal{L}_{LP}(\tilde{\mathbf{A}'}_f))\Big\},\end{aligned} \tag{27}$$

where λ is the hyper-parameter for balance. In this way, label propagation on $\tilde{\mathbf{A}}_t$ and $\tilde{\mathbf{A}}_f$ serve as a regularization

term that assist the learning of edge weights. Then in Eq.(27) we get the complete optimization object for LPM-GCN.

4.2 Complexity Analysis

Firstly, we list the used notations for complexity analysis here: C is the number of class for node classification, N is the number of nodes, E is the number of edges, and M is the number of parameters for GCN. We then analyze the complexity for both AM-GCN and LPM-GCN.

As for AM-GCN, we use the learned embedding $\mathbf{Z}$ as outputs, where $\mathbf{Z}$ is calculated with Eq.(10). Since all the $\mathbf{Z}_T$, $\mathbf{Z}_F$, $\mathbf{Z}_C$ are calculated with four GCNs, and the computational complexity for GCN is $\mathcal{O}(ECD)$, then the corresponding computational complexity for AM-GCN is $\mathcal{O}(4 * ECD)$. With the parameter sharing strategies, the number of parameters for AM-GCN is three times of GCN as $\mathcal{O}(3 * M)$.

As for LPM-GCN, the additional computations compared with AM-GCN include the calculation for the adjustable adjacency matrix in Eq.(19) and the label propagation process in Eq.(20). Actually, $\mathbf{M}_*$ is the parameter matrix initialized with the adjacency matrices with E edges, and $\circ$ is the Hadamard product. Then computational cost is $\mathcal{O}(E)$ for process in Eq. (19) and $\mathcal{O}(EC)$ for process in Eq. (20). Considering that both feature graph and topology graph are updated, the total time complexity for LPM-GCN is $\mathcal{O}(4*ECD+2*E+2*EC)$. And the number of parameters for LPM-GCN are $\mathcal{O}(3 * M + 2 * E)$.

4.3 Discussion

In this part, we make some further discussions on the proposed LPM-GCN and AM-GCN.

First, we compare the multi-channel convolution process with the Multiple Knowledge Representation [34]. Multiple Knowledge Representation (MKR) aims to acquire represent and manipulate knowledge at multiple abstraction levels, from different sources or derived by different approaches. MKR reinforces the strengths of different presentations, and helps to improve the generalization, explainability of models, and reduce the data bias. Although the proposed AM-GCN/LPM-GCN somehow introduce the multi-source features, it focuses on introducing different graph structures (i.e., topology or feature graph) than different representation approaches. AM-GCN and LPM-GCN pay more attention on the multiple structures to propagate feature and are more flexible for improving the fusion process of GCN.

Then, we discuss why using the label propagation algorithm as supervision is helpful for increasing the fusion capability based on intra-class consistency. Firstly, GCN and LP are closely related to each other. Both of them are under the assumption of homophily, i.e., the connected data samples tend to be similar or have same labels. And the inherent difference between them can be considered as the object (features/labels) to be propagated before the predictions [35]. In this way, the label smoothing process on topology using LP can also bring inspiration for studying the feature and topology fusing process of GCN. Secondly, the fusion process using GCN is always interpreted as the feature smoothing process on the network topology, where the intra-class feature influence (i.e., influence on features between nodes with the same class) plays a key role in final fusion results. Furthermore, improving the intra-class feature influence of GCN can be turned into improving the intra-class label influence with LP by enabling nodes within the same class/label to connect more strongly [33]. Therefore, we here use LP to improve the intra-class label influence and further implicitly improve the intra-class feature influence of GCNs. Specifically, we here make edge weights trainable under the supervision of LP, and then adaptively fuse node features and network topology on the learned graphs to further improve the fusion capability.

TABLE 1: The statistics of the datasets.

Dataset	Nodes	Edges	Classes	Features	Training	Test
Citeseer	3327	4732	6	3703	120/240/360	1000
UAI2010	3067	28311	19	4973	380/760/1140	1000
ACM-PAP	3025	13128	3	1870	60/120/180	1000
ACM-PSP	3025	1103868	3	1870	60/120/180	1000
BlogCatalog	5196	171743	6	8189	120/240/360	1000
Flickr	7575	239738	9	12047	180/360/540	1000
CoraFull	19793	65311	70	8710	1400/2800/4200	1000

5 Experiments

5.1 Experimental setup

Datasets. To evaluate the effectiveness of our proposed AM-GCN and LPM-GCN, we conduct experiments on seven benchmark datasets **Citeseer** [13], **UAI2010** [36], **ACM-PAP** [37], **ACM-PSP** [37], **BlogCatalog** [38], **Flickr** [38], **CoraFull** [39] in Table 1. Compared with the previous work [22], ACM-PSP is a new dataset extracted from the ACM dataset and there is an edge between two paper if they have same subjects.

Baselines. We compare AM-GCN against the following baselines in our experiments: (1) **Network Embedding methods**: DeepWalk [25], LINE [40]. (2) **Traditional graph learning methods**: Label Propagation (LP) [32], Multi-layer Perceptron (MLP) [23]. (3) **Graph Neural Networks based methods**: ChebNet [41], GCN [13], GAT [14], DEMO-Net [5], MixHop [4]. And for comparison, instead of traditional topology graph, we use the sparse *k*-nearest neighbor graph calculated from feature matrix as the input graph of GCN and represent it as *k*NN-GCN. Actually, DeepWalk, LINE and LP only utilize network topology information, while MLP only utilizes node features information, and for the rest of baselines are GNN-based method using both node features and network topology as inputs.

Parameters Setting. To more comprehensively evaluate our model, we select three label rates for training set (i.e., 20, 40, 60 labeled nodes per class) and choose 1000 nodes as the test set. All baselines are initialized with same parameters suggested by their papers and we also further carefully turn parameters to get optimal performance. For our model, we train three 2-layer GCNs with the same hidden layer dimension (*nhid1*) and the same output dimension (*nhid2*) simultaneously, where *nhid1* $\in \{512, 768\}$ and *nhid2* $\in \{32, 128, 256\}$. We use $0.0001 - 0.0005$ learning rate with Adam optimizer. In addition, the dropout rate is 0.5, weight decay $\in \{5e-3, 5e-4\}$ and $k \in \{2 \dots 10\}$ for *k*-nearest neighbor graph. The coefficient of consistency constraint and disparity constraints are searched in

TABLE 2: Node classification results(%). (We use **D.W.**, **Cheb**, **DEMO**, **M.H.**, **AMGCN**$_{w/o}$ to represent DeepWalk, ChebNet, DEMO-Net, MixHop and the variants of AMGCN without any constraints. We use Bold and Underline to show the best and the runner-up results. OOM means out-of-memory.)

Datasets	Metrics	L/C	D.W.	LINE	LP	MLP	Cheb	GCN	*k*-GCN	GAT	DEMO	M.H.	AMGCN$_{w/o}$	AM-GCN	LPM-GCN
Citeseer	**ACC**	**20**	43.47	32.71	57.70	54.90	69.80	70.30	61.35	72.50	69.50	71.40	72.36	73.10	**73.36**
		40	45.15	33.32	56.50	62.26	71.64	73.10	61.54	73.04	70.44	71.48	73.04	74.70	**74.90**
		60	48.86	35.39	56.00	66.48	73.26	74.48	62.38	74.76	71.86	72.16	74.22	75.56	**75.94**
	F1	**20**	38.09	31.75	57.34	52.82	65.92	67.50	58.86	68.14	67.84	66.96	67.76	68.42	**69.28**
		40	43.18	32.42	56.94	58.66	68.31	69.70	59.33	69.58	66.97	67.40	68.55	69.81	**69.98**
		60	48.01	34.37	56.59	63.61	70.31	71.24	60.07	**71.60**	68.22	69.31	69.24	70.92	71.38
UAI2010	**ACC**	**20**	42.02	43.47	42.00	69.48	50.02	49.88	66.06	56.92	23.45	61.56	68.80	70.10	**70.48**
		40	51.26	45.37	50.60	71.32	58.18	51.80	68.74	63.74	30.29	65.05	72.48	73.14	**73.62**
		60	54.37	51.05	54.20	73.14	59.82	54.40	71.64	68.44	34.11	67.66	73.62	74.40	**74.82**
	F1	**20**	32.93	37.01	36.01	54.85	33.65	32.86	52.43	39.61	16.82	49.19	**56.90**	55.61	56.40
		40	46.01	39.62	40.49	56.40	38.80	33.80	54.45	45.08	26.36	53.86	64.04	**64.88**	64.02
		60	44.43	43.76	43.53	57.84	40.60	34.12	54.78	48.97	29.05	56.31	**68.23**	65.99	66.27
ACM-PAP	**ACC**	**20**	62.69	41.28	67.90	77.40	75.24	87.80	78.52	87.36	84.48	81.08	89.68	90.40	**90.62**
		40	63.00	45.83	67.90	80.92	81.64	89.06	81.66	88.60	85.70	82.34	89.70	90.76	**90.82**
		60	67.03	50.41	68.90	84.52	85.43	90.54	82.00	90.40	86.55	83.09	90.28	91.42	**91.58**
	F1	**20**	62.11	40.12	68.31	77.17	74.86	87.82	78.14	87.44	84.16	81.40	89.57	90.43	**90.53**
		40	61.88	45.79	69.25	80.73	81.26	89.00	81.53	88.55	84.83	81.13	89.61	90.66	**90.77**
		60	66.99	49.92	69.24	84.38	85.26	90.49	81.95	90.39	84.05	82.24	90.24	91.36	**91.54**
BlogCatalog	**ACC**	**20**	38.67	58.75	40.90	75.16	38.08	69.84	75.49	64.08	54.19	65.46	79.56	81.98	**82.58**
		40	50.80	61.12	47.70	81.24	56.28	71.28	80.84	67.40	63.47	71.66	83.30	84.94	**85.26**
		60	55.02	64.53	48.30	85.30	70.06	72.66	82.46	69.95	76.81	77.44	85.94	87.30	**87.48**
	F1	**20**	34.96	57.75	38.85	73.94	33.39	68.73	72.53	63.38	52.79	64.89	79.11	81.36	**82.08**
		40	48.61	60.72	45.30	80.40	53.86	70.71	80.16	66.39	63.09	70.84	82.83	84.32	**84.91**
		60	53.56	63.81	45.75	84.38	68.37	71.80	81.90	69.08	76.73	76.38	85.53	86.94	**87.07**
Flickr	**ACC**	**20**	24.33	33.25	19.10	56.06	23.26	41.42	69.28	38.52	34.89	39.56	74.58	75.26	**75.62**
		40	28.79	37.67	20.30	68.58	35.10	45.48	75.08	38.44	46.57	55.19	76.00	80.06	**80.36**
		60	30.10	38.54	25.20	75.58	41.70	47.96	77.94	38.96	57.30	64.96	80.56	82.10	**82.30**
	F1	**20**	21.33	31.19	12.46	55.76	21.27	39.95	70.33	37.00	33.53	40.13	75.21	74.63	**75.34**
		40	26.90	37.12	14.39	67.65	33.53	43.27	75.40	36.94	45.23	56.25	76.81	79.36	**80.34**
		60	27.28	37.77	19.29	75.02	40.17	46.58	77.97	37.35	56.49	65.73	80.37	81.81	**81.82**
CoraFull	**ACC**	**20**	29.33	17.78	51.80	48.12	53.38	56.68	41.68	58.44	54.50	47.74	56.24	58.90	**58.90**
		40	36.23	25.01	55.90	53.68	58.22	60.60	44.80	62.98	60.28	57.20	62.22	**63.62**	62.70
		60	40.60	29.65	56.80	56.30	59.84	62.00	46.68	64.38	61.58	60.18	64.88	**65.36**	64.60
	F1	**20**	28.05	18.24	46.73	43.05	47.59	52.48	37.15	54.44	50.44	45.07	51.22	54.74	**54.78**
		40	33.29	25.43	52.57	49.23	53.47	55.57	40.42	58.30	56.26	53.55	57.04	**59.19**	58.63
		60	37.95	30.87	54.25	51.83	54.15	56.24	43.22	59.61	57.26	56.40	60.15	**61.32**	60.85
ACM-PSP	**ACC**	**20**	62.50	62.11	63.50	75.24	79.86	64.50	82.46	67.04	OOM	79.40	87.54	89.26	**90.04**
		40	63.49	62.77	66.30	81.64	82.98	65.62	85.26	68.88	OOM	82.36	87.80	89.50	**90.38**
		60	66.31	64.33	66.20	85.43	86.28	70.22	86.26	69.46	OOM	82.80	88.86	90.30	**90.72**
	F1	**20**	61.25	60.56	61.09	77.17	79.48	57.60	82.34	59.88	OOM	79.46	87.34	89.07	**89.88**
		40	62.87	60.86	64.96	80.73	82.85	58.31	85.25	64.50	OOM	82.53	87.73	89.50	**90.26**
		60	65.78	64.28	64.82	84.38	86.06	67.87	86.24	66.50	OOM	82.78	88.82	90.21	**90.61**

$\{0.01, 0.001, 0.0001\}$ and $\{1e-10, 5e-9, 1e-9, 5e-8, 1e-8\}$. For all methods, we run 5 times with the same partition and report the average results. And we use Accuracy (ACC) and macro F1-score (F1) to evaluate performance of models. As for the LPM-GCN, we further fine tune the hyper-parameters sets based on the settings for AM-GCN.

5.2 Node classification

The node classification results are reported in Table 4, where L/C means the number of labeled nodes per class. We have the following observations: Compared with all baselines, the proposed LPM-GCN and AM-GCN generally achieve the best and the runner-up performance on all datasets with all label rates. The results demonstrate the effectiveness of both LPM-GCN and AM-GCN.

- The extended model LPM-GCN always makes some improvements against AM-GCN on all datasets, which means the LP guided edge weights learning strategy is effective. What's more, the $AMGCN_{w/o}$ variants, without any constraints, still achieve very competitive performance against all baselines, demonstrating that our framework is stable and competitive.
- AM-GCN consistently outperforms GCN and *k*NN-GCN on all the datasets, indicating the effectiveness of the adaptive fusion mechanism in AM-GCN, because it can extract more useful information than only performing GCN and *k*NN-GCN respectively. And the adaptive ability of AM-GCN is significant for application because of the complexity of the real datasets.
- Comparing with GCN and *k*NN-GCN, we can learn that

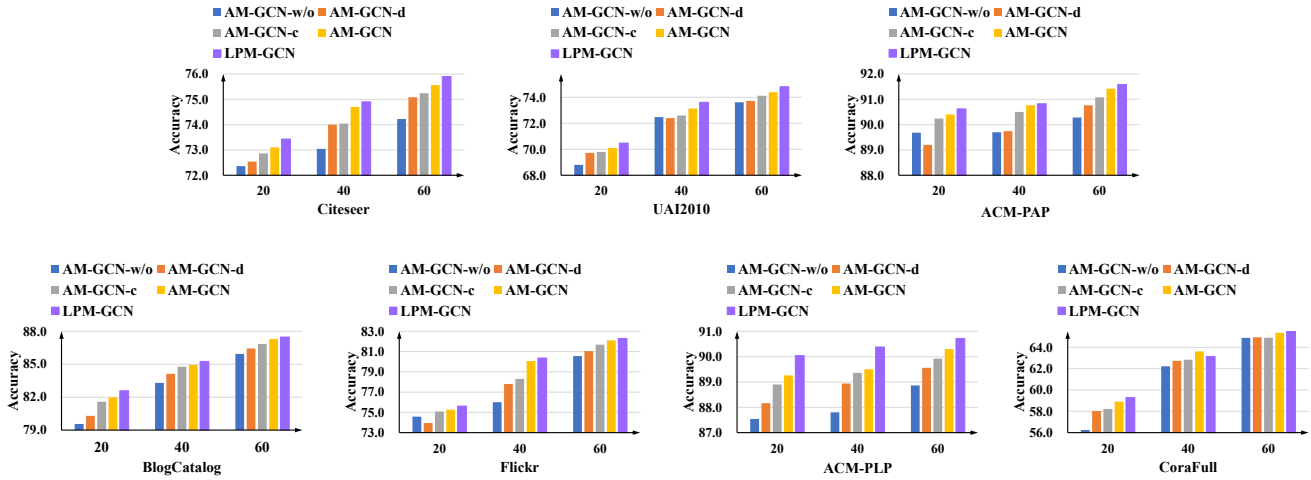
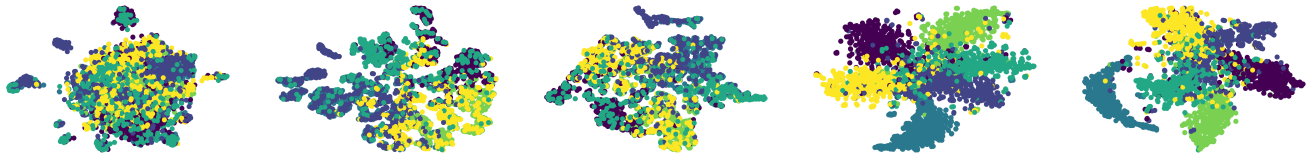


Fig. 3: Ablation experimental results of the LPM-GCN and AM-GCN and their variants on seven datasets.



(a) **DeepWalk** (b) **GCN** (c) **GAT** (d) **AM-GCN** (e) **LPM-GCN**

Fig. 4: Visualization of the learned node embeddings on BlogCatalog dataset. The Silhouette Coefficient for AM-GCN is 0.2529 and for LPM-GCN is 0.2608.

there does exist structural difference between topology graph and feature graph and performing GCN on traditional topology graph does not always show better result than on feature graph. For example, in BlogCatalog, Flickr and UAI2010, the feature graph performs better than topology. This further confirms the necessity of introducing feature graph in GCN.

- Moreover, compared with GCN, the improvement of AM-GCN is more substantial on the datasets with better feature graph (kNN), such as UAI2010, BlogCatalog, Flickr. This implies that AM-GCN introduces a better and more suitable kNN graph for label to supervise feature propagation and node representation learning.

5.3 Analysis of variants

In this section, we compare AM-GCN with its three variants and the extended model LPM-GCN on all datasets to validate the effectiveness of consistency constraints, disparity constraints and the LP guided edge weights learning.

- **AM-GCN**$_{w/o}$: A variant of AM-GCN without constraints $\mathcal{L}_c$ and $\mathcal{L}_d$.
- **AM-GCN**$_c$: A variant of AM-GCN only with the consistency constraint $\mathcal{L}_c$.
- **AM-GCN**$_d$: A variant of AM-GCN only with the disparity constraint $\mathcal{L}_d$.

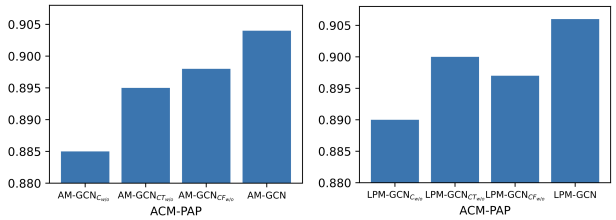
From the results in Figure 3, we can draw the following conclusions: (1) The results of LPM-GCN are consistently better than AM-GCN, indicating that the edge weights learning with label propagation is useful for the label-consistency learning and information fusion of our framework. (2) The results of AM-GCN are consistently better than all the other three variants, indicating the effectiveness of using the two constraints together. (3) The results of AM-GCN$_c$ and AM-GCN$_d$ are usually better than AM-GCN$_{w/o}$ on all datasets with all label rates, verifying the usefulness of the two constraints. (4) AM-GCN$_c$ is generally better than AM-GCN$_d$ on all datasets, which implies the consistency constraint plays a more vital role in this framework.

5.4 Visualization

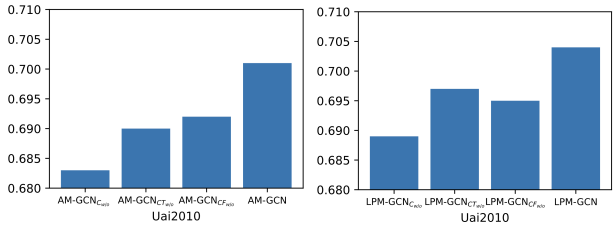
For a more intuitive comparison and to further show the effectiveness of our proposed model, we conduct the task of visualization on the BlogCatalog dataset. We use the output embedding on the last layer of LPM-GCN (or AM-GCN, GCN, GAT) before $softmax$ and plot the learned embedding of the test set using t-SNE [42]. The results of BlogCatalog in Figure 4 are colored by real labels.

From Figure 4, we can find that the results of DeepWalk, GCN and GAT are not satisfactory, because the nodes with different labels are mixed together. Apparently, the visualization of AM-GCN and LPM-GCN perform better, where the learned embedding has a more compact structure, the highest intra-class similarity, and the clearest distinct boundaries among different classes.

To further analyze the visualization results between AM-GCN and LPM-GCN, we calculate the Silhouette Coefficient [43], where a higher Silhouette Coefficient score relates to a better defined clusters. Then we record the average Silhouette Coefficient for all nodes on the BlogCatalog dataset with 0.2529 for AM-GCN and 0.2608 for LPM-GCN. We can



(a) **ACM-PAP**



(b) **UAI2010**

Fig. 5: Analysis on the common embeddings $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$.

see that LPM-GCN have better clusters compared with AM-GCN, which means that the learned embeddings of LPM-GCN between different classes are more separable than those of AM-GCN, implying that LP guided edge weights learning will make the embedding more discriminative.

5.5 Analysis of Common Convolutional Embedding

To demonstrate the effectiveness of combing $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$, we include another ablation studies on both AM-GCN and LPM-GCN with ACM-PAP and UAI2010. The corresponding variants are shown as follow:

- **AM-GCN**$_{C_{w/o}}$: A variant of AM-GCN without both $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$.
- **AM-GCN**$_{CT_{w/o}}$: A variant of AM-GCN without $\mathbf{Z}_{CT}$.
- **AM-GCN**$_{CF_{w/o}}$: A variant of AM-GCN without $\mathbf{Z}_{CF}$.
- **LPM-GCN**$_{C_{w/o}}$: A variant of LPM-GCN without both $\mathbf{Z}_{CT}$ and $\mathbf{Z}_{CF}$.
- **LPM-GCN**$_{CT_{w/o}}$: A variant of LPM-GCN without $\mathbf{Z}_{CT}$.
- **LPM-GCN**$_{CF_{w/o}}$: A variant of LPM-GCN without $\mathbf{Z}_{CF}$.

From the results in Figure 5, we can see that representations without any common embedding Z_{CT} or Z_{CF} have the worst performance. And the representations combining both Z_{CT} and Z_{CF} have the best results on all three datasets with AM-GCN and LPM-GCN. From these results we can see that it is necessary to utilize both common embeddings Z_{CT} and Z_{CF} to achieve better prediction results for AM-GCN and LPM-GCN.

5.6 Analysis of attention mechanism

In order to investigate whether the attention values learned by our proposed model are meaningful, we analyze the attention distribution and attention learning trend using LPM-GCN, respectively. And **Topology**, **Common** and **Feature** in Figure 6 and Figure 7 correspond to the attention weights of embedding $\mathbf{Z}_T$, $\mathbf{Z}_C$ and $\mathbf{Z}_F$ in Eq. (10). The results of AM-GCN is similar with that of LPM-GCN.

Analysis of attention distributions. Our proposed models learns two specific embeedings and one common embedding, each of which is associated with the attention values. We conduct the attention distribution analysis on all datasets with 20 label rate, where the results are shown in Figure 6. As we can see, for Citeseer, ACM, CoraFull, the attention values of specific embeddings in topology space are larger than the values in feature space, and the values of common embeddings are between them. This implies that the information in topology space should be more important than the information in feature space. To verify this, we can see that the results of GCN are better than *k*NN-GCN in Table 4. Conversely, for UAI2010, BlogCatalog and Flickr, in comparison with Figure 6 and Table 4, we can find *k*NN-GCN performs better than GCN, meanwhile, the attention values of specific embeddings in feature space are also larger than those in topology space. In summary, the experiment demonstrates that our proposed models is able to adaptively assign larger attention value for more important information. Due to the space limitation, we mainly show the results of LPM-GCN in this paper, while similar results of AM-GCN are provided in our KDD paper [22].

Analysis of attention trends. Here we also analyze the changing trend of attention values during the training process with LPM-GCN. Here we take Citeseer, BlogCatalog as examples in Figure 7, where **x**-axis is the epoch and **y**-axis is the average attention value of each embedding. We can see that at the beginning, the average attention values of **Topology**, **Feature**, and **Common** are almost the same, with the training epoch increasing, the attention values become different. For example, in BlogCatalog, the attention value for topology gradually decreases, while the attention value for feature keeps increasing. This phenomenon is consistent with the conclusions in Table 4 and Figure 6, i.e., the feature graph (*k*NN-GCN) performs better than GCN and the information in feature space is more important than in topology space. We can see that LPM-GCN can learn the importance of different embeddings step by step.

5.7 Analysis on the learned edge weights

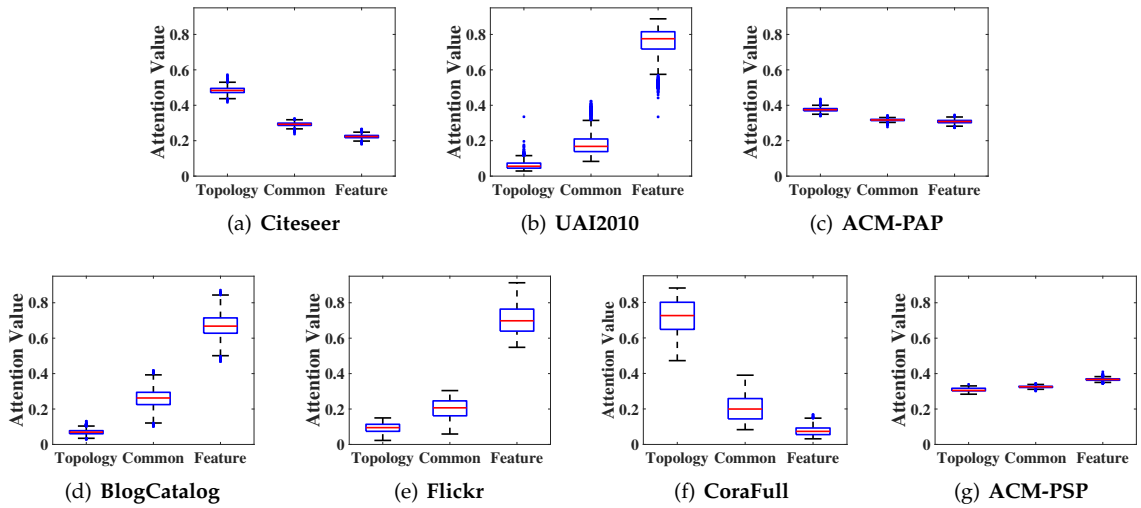
In order to verify that the LP guided edge weights learning is able to increase the intra-class consistency and is suitable for the AM-GCN framework, we evaluate the updated graph and original input graph with *Modularity*.

Modularity is a measure of the quality of a particular division of a network [44], [45]. Formally, the Modularity score is defined as :

$$Q = \frac{1}{2m}\sum(\mathbf{A}_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j). \tag{28}$$

Here $\mathbf{A}$ is the adjacency matrix of an undirected weighted graph with self-loop, $\mathbf{A}_{ij}$ is the edge weight of node i and node j, $m = \frac{1}{2}\sum_{ij}\mathbf{A}_{ij}$, $k_i = \sum_j \mathbf{A}_{ij}$, C_i is the class that node i belongs to and $\delta(C_i, C_j) = 1$ if $C_i = C_j$. Given the network structure, Modularity can evaluate whether the results of community division is good or not. And the higher Modularity score means a better partition. On the other hand, considering that a higher Modularity usually implies a clearer community structure, we can compare the original input graph structure and updated graph structure based on modularity, so as to check which one will emerge more clearer community structure.

The results are shown in Table 3, and we have the following conclusions: (1) The Modularity scores of all datasets



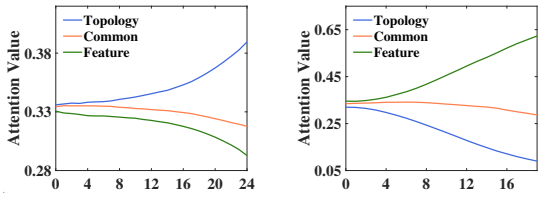
(a) **Citeseer** (b) **UAI2010** (c) **ACM-PAP**

(d) **BlogCatalog** (e) **Flickr** (f) **CoraFull** (g) **ACM-PSP**

Fig. 6: Analysis of attention distribution with **LPM-GCN**.

TABLE 3: Modularity comparisons on updated graphs with **LPM-GCN**. **Old** and **New** respectively represent the modularity scores for original input graph and updated graph. We use Bold to mark the higher improvement.

Dataset	Topology Graph			Feature Graph		
	Old	New	Improvement	Old	New	Improvement
Citeser	0.6167	0.6687	**+5.20**%	0.4526	0.4707	+1.81%
UAI	0.2926	0.3008	+0.82%	0.5133	0.5363	**+2.30**%
ACM-PAP	0.4986	0.5107	**+1.21**%	0.4450	0.4504	+0.54%
BlogCatalog	0.2333	0.2345	+0.12%	0.6280	0.6398	**+1.18**%
Flickr	0.1333	0.1335	+0.02%	0.7106	0.7160	**+0.54**%
ACM-PSP	0.3014	0.3016	+0.02%	0.4450	0.4506	**+0.56**%



(a) **Citeseer** (b) **BlogCatalog**

Fig. 7: The attention changing trends w.r.t epochs.

on both the topology graph and the feature graph are improved, indicating the LP guided edge weight updating can effectively help learn a better network structure and improve the intra-class smoothing, i.e. edge weights of nodes in the same class are increased. (2) From the Modularity scores on the initial topology graph and feature graph, we can also know about the structure difference between the two spaces. And larger modularity means larger label consistency for the graph. For Citeseer and ACM-PAP, the topological modularity is larger than that of feature space, indicating that the label consistency of topology space is larger. And for UAI, BlogCatalog, Flickr and ACM-PSP, the feature structure is better than the topology structure. These are also consistent with the analysis of attention mechanism in section 5.6. (3) Furthermore, the attention mechanism can also influence the learning process of the edge weights updating. By comparing Figure 6 and Table 3, we can find

TABLE 4: Comparisons on Kernels for constructing feature graph.

Datasets	Metrics	L/C	AMGCN$_C$	LPMGCN$_C$	AMGCN$_H$	LPMGCN$_H$
Citeseer	**ACC**	**20**	73.10	73.36	72.20	72.80
		40	74.70	74.90	72.50	73.20
		60	75.56	75.94	73.70	74.60
	F1	**20**	68.42	69.28	68.04	68.27
		40	69.81	69.98	68.45	69.95
		60	70.92	71.38	69.81	70.67
UAI2010	**ACC**	**20**	70.10	70.48	69.20	68.90
		40	73.14	73.62	71.30	72.50
		60	74.40	74.82	72.20	72.80
	F1	**20**	55.61	56.40	54.57	54.68
		40	64.88	64.02	62.34	62.86
		60	65.99	66.27	63.79	64.46
ACM-PAP	**ACC**	**20**	90.40	90.62	89.58	89.92
		40	90.76	90.82	89.65	90.20
		60	91.42	91.58	90.04	91.06
	F1	**20**	90.43	90.53	89.42	88.56
		40	90.66	90.77	88.57	89.67
		60	91.36	91.54	89.95	90.88

that for datasets whose topological attention is larger than that of feature, the improvement on the topology graph is also larger than the feature graph. Conversely, for UAI, BlogCatalog, Flickr, ACM-PSP, the improvements on the feature graph are larger.

5.8 Analysis on the Similarity Matrix Construction

We further compare the effectiveness of using cosine similarity or heat kernel to construct the feature graph in Section 3.1. We conduct experiments on three datasets Citeseer, UAI2010 and ACM-PAP with three label rates and two metrics. The experimental results are shown in the Table 5. From the results, we can find the cosine similarity currently used in AM-GCN and LPM-GCN for feature graph constructing works better than the heat kernel on these three datasets.

5.9 Parameter Study

In this section, we investigate the sensitivity of parameters on Citeseer, BlogCatalog datasets with LPM-GCN models. Actually, the analysis of the parameters of AM-GCN has been provided in our KDD paper. Therefore, limited by space, we only report the results of our extended model LPM-GCN.

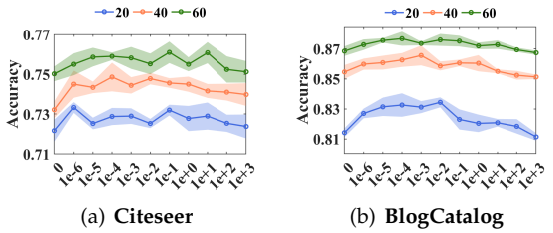
Analysis of consistency coefficient γ. We test the effect of the weight γ of the consistency constraint in Eq. (18), and vary it from {1e-6, 1e+3}. The results are shown in Figure 8. As we can see, with the increase of the consistency coefficient, the performance raises first and then starts to drop slowly. Basically, LPM-GCN is stable when the γ is within the range from 1e-4 to 1e+4 on all datasets. We can also see that the curves of 20, 40, 60 label rates show a similar changing trend.

Analysis of disparity constraint coefficient β. We then test the effect of the weight β of the disparity constraint in Eq. (18) and vary it from 0 to 1e-5. The results are shown in Figure 9. Similarly, with the increase of β, the performance also raises first, but the performance will drop quickly if β is larger than 1e-6 for Citeseer, while for BlogCatalog, it is relatively stable.

Analysis of *k*-nearest neighbor graph k. In order to check the impact of the top k neighborhoods in kNN graph, we study the performance of LPM-GCN with a various number of k ranging from 2 to 10 in Figure 10. Generally, accuracy increases first and then starts to decrease. This is probably because if the graph becomes dense, the feature is very easy to be smoothed, and also, too many edges may also introduce some noisy edges here.

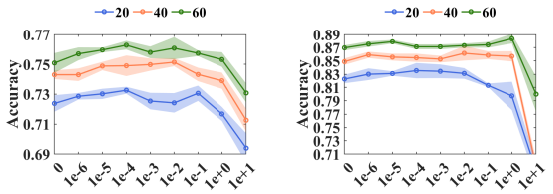
Analysis of LP balance coefficient λ. λ is a new parameter for LPM-GCN which is used as the balance coefficient in Eq. (27). It can control the LP influence against initial AM-GCN. Actually, different datasets may have different suitable balance coefficient. And in some cases, such as dataset BlogCatalog, larger λ i.e. more than 100, may also bring enhancement on results. But it is clear that training without the LP loss (i.e., $\lambda = 0$) always get worse results, justifying it is hard for AM-GCN part to learn both $\mathbf{W}_{set}$ and edge weights simultaneously without the assistance of LP-guided regularization.

Analysis of LP iteration times in LPM-GCN. We also evaluate the influence of the number of LP iterations in LPM-GCN in Citeseer, BlogCatalog datasets, and vary it from 1 to 9. From Figure 6 we observe that the performance raise at first when the number of iteration increase and then the accuracy stops increasing and decreases since a large number of LP will include more noisy nodes. We can also see that the curves of 20, 40, 60 label rates show a similar changing trend.



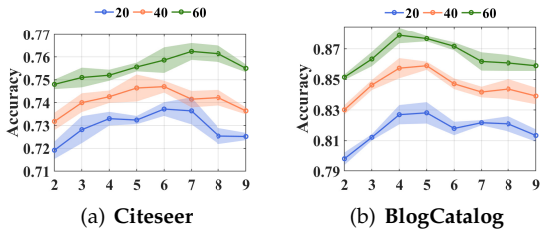
(a) **Citeseer** (b) **BlogCatalog**

Fig. 8: Analysis of parameter γ with **LPM-GCN**.



(a) **Citeseer** (b) **BlogCatalog**

Fig. 9: Analysis of parameter β with **LPM-GCN**.

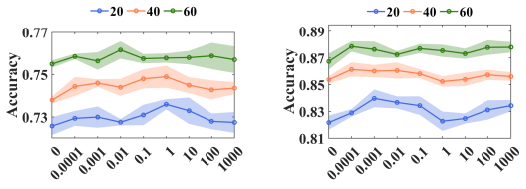


(a) **Citeseer** (b) **BlogCatalog**

Fig. 10: Analysis of parameter k with **LPM-GCN**.

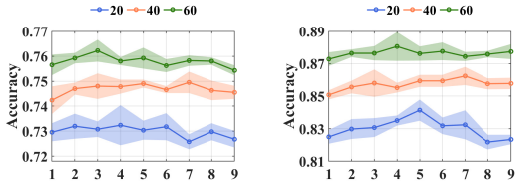
6 Related Work

Graph Neural Networks. Recently, Graph Neural Networks model and its variants [46], [47], [48], [49], [50], [51], [52] have been widely studied. Generally, GNNs fall into two categories, spectral-based [41], [53] and spatial-based. Actually, the spatial models usually follow a message passing manner where the key step is feature propagating and aggregating along the network topology. GCN [13] aggregates node features from one-hop neighbors. GAT [14] introduces an attention mechanism to aggregate node features with the learned weights. GraphSAGE [15] proposes to sample and aggregate features from a node's local neighborhood with mean/max/LSTM pooling. Some recent works give analysis on the roles of topology, feature or labels. For example, [18] shows that GCNs actually perform the Laplacian smoothing on node features, [19] and [17] prove that topological structures play the role of low-pass filtering on node features; DSP-GCN [54] argue that attributes may be interfered by the utilization of topology information and cause misclassifications; ConfGCN [55] focus on the label confidence scores to define the influence of one node on another during the propagation process. However, none of them take advantage of feature structural information and



(a) **Citeseer** (b) **BlogCatalog**

Fig. 11: Analysis of parameter λ with **LPM-GCN**.



(a) **Citeseer** (b) **BlogCatalog**

Fig. 12: Analysis of LP iteration times in **LPM-GCN**.

whether GCNs can adaptively extract the most correlated information for specific tasks remains unclear.

Graph Structure Learning and Attention Mechanism. Real-world graphs are often noisy and incomplete, which means topology graph is not always right for feature propagation in GCNs. Thus many works focus on learning graph structure while training GNNs. For example, LDS [56] uses a bilevel optimization process to learn the graph structure and the parameters of GCNs jointly. DIAL-GNN [57] casts the graph structure learning problems as a data-driven similarity metric learning problem in an inductive setting. Furthermore, some attention-based GNNs [14], [58], [59] are also trying to find a more suitable graph. As for LPM-GCN, it directly utilizes label similarities rather than feature similarities to supervise the edge weights learning and is more beneficial for final tasks.

7 Conclusion

In this paper, we rethink the fusion mechanism of network topology and node features in GCN and surprisingly discover it is distant from optimal or even satisfactory with empirical experiments. Motivated by this fundamental problem, we study how to adaptively learn the most correlated information from topology and node features and sufficiently fuse them for classification. We propose a multi-channel model AM-GCN to learn both topology structural information and feature structural information at the same time and AM-GCN is able to learn suitable importance weights for the combination of these information. Moreover, we introduce the label propagation based edge weights learning and propose the LPM-GCN model to further improve the performance. Extensive experiments well demonstrate the superior performance over the state-of-the-art models on seven real-world datasets.

References

[1] W. Li, Y. Jia, and J. Du, "Recursive state estimation for complex networks with random coupling strength," *Neurocomputing*, vol. 219, pp. 1–8, 2017.

[2] F. Kou, J. Du, Y. He, and L. Ye, "Social network search based on semantic analysis and learning," *CAAI Trans. Intell. Technol.*, vol. 1, no. 4, pp. 293–302, 2016.

[3] F. Kou, J. Du, Z. Lin, M. Liang, H. Li, L. Shi, and C. Yang, "A semantic modeling method for social network short text based on spatial and temporal characteristics," *J. Comput. Sci.*, vol. 28, pp. 281–293, 2018.

[4] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *ICML*, 2019, pp. 21–29.

[5] J. Wu, J. He, and J. Xu, "Demo-net: Degree-specific graph neural networks for node and graph classification," in *SIGKDD*, 2019, pp. 406–415.

[6] H. Gao and S. Ji, "Graph u-nets," in *ICML*, 2019, pp. 2083–2092.

[7] M. Zhang, Z. Cui, M. Neumann, and C. Yixin, "An end-to-end deep learning architecture for graph classification," in *AAAI*, 2018, pp. 4438–4445.

[8] J. You, Rex, and J. Leskovec, "Position-aware graph neural networks," in *ICML*, 2019, pp. 7134–7143.

[9] T. N. Kipf and M. Welling, "Variational graph auto-encoders." 2016.

[10] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *SIGKDD*, 2018, pp. 974–983.

[11] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *WWW*, 2019, pp. 417–426.

[12] C. Shi, X. Han, L. Song, X. Wang, S. Wang, J. Du, and P. S. Yu, "Deep collaborative filtering with multi-aspect information in heterogeneous networks," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1413–1425, 2021.

[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[14] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[15] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1024–1034.

[16] J. Ma, P. Cui, K. Kuang, X. Wang, and wenwu zhu, "Disentangled graph convolutional networks," in *ICML*, 2019, pp. 4212–4221.

[17] F. Wu, T. Zhang, A. H. de Souza, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *ICML*, 2019, pp. 6861–6871.

[18] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018, pp. 3538–3545.

[19] H. Nt and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters." *arXiv preprint arXiv:1905.09550*, 2019.

[20] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," in *ICLR*, 2020.

[21] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," in *NeurIPS*, 2020.

[22] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *KDD*, 2020, pp. 1243–1253.

[23] S. Pal and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683–697, 1992.

[24] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Physical Review E*, vol. 83, no. 1, p. 16107, 2011.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *SIGKDD*, 2014, pp. 701–710.

[26] Y. Ding, H. Fan, M. Xu, and Y. Yang, "Adaptive exploration for unsupervised person re-identification," *ACM Trans. Multim. Comput. Commun. Appl.*, vol. 16, no. 1, pp. 3:1–3:19, 2020.

[27] L. Song, A. Smola, A. Gretton, K. M. Borgwardt, and J. Bedo, "Supervised feature selection via dependence estimation," in *ICML*, 2007, pp. 823–830.

[28] D. Niu, J. G. Dy, and M. I. Jordan, "Multiple non-redundant spectral clustering views," in *ICML*, 2010, pp. 831–838.

[29] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *ALT*, 2005, pp. 63–77.

[30] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," *Center for Automated Learning and Discovery, CMU: Carnegie Mellon University, USA.*, 2002.
[31] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *NeurIPS*, 2003, pp. 321–328.
[32] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, school of language technologies institute, 2005.
[33] H. Wang and J. Leskovec, "Unifying graph convolutional neural networks and label propagation," *arXiv preprint arXiv:2002.06755*, 2020.
[34] Y. Yang, Y. Zhuang, and Y. Pan, "Multiple knowledge representation for big data artificial intelligence: framework, applications, and case studies," *Frontiers Inf. Technol. Electron. Eng.*, vol. 22, no. 12, pp. 1551–1558, 2021.
[35] L. Yang, F. Wu, Y. Wang, J. Gu, and Y. Guo, "Masked graph convolutional network." in *IJCAI*, 2019, pp. 4070–4077.
[36] W. Wang, X. Liu, P. Jiao, X. Chen, and D. Jin, "A unified weakly supervised framework for community detection and semantic matching," in *PAKDD*, 2018, pp. 218–230.
[37] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *WWW*, 2019, pp. 2022–2032.
[38] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *WSDM*, 2019, pp. 393–401.
[39] A. Bojchevski and S. Günnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," in *ICLR*, 2018.
[40] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *WWW*, 2015, pp. 1067–1077.
[41] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NeurIPS*, 2016, pp. 3844–3852.
[42] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
[43] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.
[44] M. Newman, "Modularity and community structure in networks," *Bulletin of the American Physical Society*, 2006.
[45] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks." *Physical Review E*, vol. 69, no. 2, pp. 26 113–26 113, 2004.
[46] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *SIGKDD*, 2018, pp. 1416–1424.
[47] J. Chen, T. Ma, and C. Xiao, "Fastgcn: Fast learning with graph convolutional networks via importance sampling," in *ICLR*, 2018.
[48] T. Derr, Y. Ma, and J. Tang, "Signed graph convolutional networks," in *ICDM*, 2018, pp. 929–934.
[49] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *SIGKDD*, 2019, pp. 723–731.
[50] M. Qu, Y. Bengio, and J. Tang, "Gmnn: Graph markov neural networks," in *ICML*, 2019, pp. 5241–5250.
[51] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks," in *ICLR*, 2019.
[52] Z. Ying, I. Chami, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *NeurIPS*, 2019, pp. 4869–4880.
[53] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *ICLR*, 2014.
[54] L. Yang, Z. Chen, J. Gu, and Y. Guo, "Dual self-paced graph convolutional network: Towards reducing attribute distortions induced by topology," in *IJCAI*, 2019, pp. 4062–4069.
[55] S. Vashishth, P. Yadav, M. Bhandari, and P. P. Talukdar, "Confidence-based graph convolutional networks for semi-supervised learning," in *AISTATS*, 2019, pp. 1792–1801.
[56] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *ICML 2019 : Thirty-sixth International Conference on Machine Learning*, 2019, pp. 1972–1982.
[57] Y. Chen, L. Wu, and M. J. Zaki, "Deep iterative and adaptive learning for graph neural networks," *arXiv:1912.07832*, 2019.
[58] K. Zhang, Y. Zhu, J. Wang, and J. Zhang, "Adaptive structural fingerprints for graph attention networks," in *ICLR*, 2020.
[59] K. K. Thekumparampil, S. Oh, C. Wang, and L.-J. Li, "Attention-based graph neural network for semi-supervised learning," *arXiv preprint arXiv:1803.03735*, 2018.

Meiqi Zhu received the B.S. degree in 2019 and the M.S. degree in 2022 from Beijing University of Posts and Telecommunications. She currently works for the AntGroup, Beijing. Her current research interests are in graph neural networks, data mining and machine learning.

Xiao Wang is an Assistant Professor in the School of Computer Science, Beijing University of Posts and Telecommunications. He received his Ph.D. degree from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2016. He was a postdoctoral researcher in Department of Computer Science and Technology, Tsinghua University, Beijing, China. He got the China Scholarship Council Fellowship in 2014 and visited Washington University, as a joint training student from 2014 to 2015. His current research interests include data mining, social network analysis, and machine learning. Until now, he has published more than 50 papers in conferences such as AAAI, IJCAI, WWW, KDD, etc. and journals such as IEEE TKDE, IEEE Trans. on Cybernetics, etc.

Chuan Shi received the B.S. degree from the Jilin University in 2001, the M.S. degree from the Wuhan University in 2004, and Ph.D. degree from the ICT of Chinese Academic of Sciences in 2007. He joined the Beijing University of Posts and Telecommunications as a lecturer in 2007, and is a professor and deputy director of Beijing Key Lab of Intelligent Telecommunications Software and Multimedia at present. His research interests are in data mining, machine learning, and evolutionary computing. He has published more than 40 papers in refereed journals and conferences.

Yibo Li received the BS degree from the Beijing University of Posts and Telecommunications, China, in 2022. She is currently a master student in Beijing University of Posts and Communications, China. Her current research interests are in graph neural networks and machine learning.

Junping Du received the PhD degree in computer science from the University of Science and Technology Beijing (USTB), and then held postdoc fellowship in the Department of Computer Science, Tsinghua University, Beijing, China. She joined the School of Computer Science, Beijing University of Posts and Telecommunications (BUPT), in July 2006, where she is currently a professor of computer science. She was a visiting professor with the Department of Computer Science, Aarhus University, Denmark, from September 1996 until September 1997. Her current research interests include artificial intelligence, data mining, intelligent management system development, and computer applications.