Exploring the Role of Node Diversity in Directed Graph Representation Learning

Jincheng Huang[1] , **Yujie Mo**[1] , **Ping Hu**[1] , **Xiaoshuang Shi**[1] , **Shangbo Yuan**[3] , **Zeyu Zhang**[2*] , **Xiaofeng Zhu**[1*]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China
[2]Huazhong Agricultural University
[3]School of Engineering and Design, Technical University of Munich
{huangjc0429, moyujie2017, chinahuping, xsshi2013}@gmail.com, ge47deq@mytum.de, zhangzeyu@mail.hzau.edu.cn, seanzhuxf@gmail.com

Abstract

Many methods of Directed Graph Neural Networks (DGNNs) are designed to equally treat nodes in the same neighbor set (*i.e.,* out-neighbor set and in-neighbor set) for every node, without considering the node diversity in directed graphs, so they are often unavailable to adaptively acquire suitable information from neighbors of different directions. To alleviate this issue, in this paper, we investigate a new way to first consider node diversity for representation learning on directed graphs, *i.e.,* neighbor diversity and degree diversity, and then propose a new NDDGNN framework to adaptively assign weights to both outgoing information and incoming information at the node level. Extensive experiments on seven real-world datasets validate the superior performance of our method compared to state-of-the-art methods in terms of both node classification and link prediction tasks.

1 Introduction

Graph neural networks (GNNs) have achieved great success by analyzing the undirected graph to extract representation from the graph data. However, many graphs are directed where each edge is bidirectional in real applications, such as traffic networks and web page networks. As a result, many previous GNNs considering undirected graphs (*e.g.,* [Kipf and Welling, 2017; Hamilton *et al.*, 2017; Velickovic *et al.*, 2018; Yang *et al.*, 2023b; Yang *et al.*, 2023a; Bi *et al.*, 2022b; Mo *et al.*, 2023; Mo *et al.*, 2024; Huang *et al.*, 2023a]) ignore the directional information, and thus resulting in discarding valuable information crucial for downstream tasks [Rossi *et al.*, 2023]. For example, in the web page network, web pages pointed to more links tend to have more traffic. If the directional information of the web page network is ignored, some web pages with few incoming edges and much outgoing edges will be predicted as high-traffic web pages. Hence, it is practical to consider the directional information for applying GNNs in the directed graphs.

[*]Corresponding authors: X. Zhu and Z. Zhang

Previous GNN methods for directed graphs primarily focus on delineating the directional information in the graph, including spectral DGNN methods [Zhang *et al.*, 2021; Tong *et al.*, 2020a; Tong *et al.*, 2021] and spatial DGNN methods [Rossi *et al.*, 2023; Tong *et al.*, 2020b]. The spectral DGNN methods mainly focus on constructing the Laplacian matrix with the directional information. For example, DiGCN [Tong *et al.*, 2020a] employs the transfer probability of personalized PageRank to approximate the Laplace matrix of directed graphs for considering the directional information [Brin, 1998]. MagNet [Zhang *et al.*, 2021] designs the magnetic Laplacian matrix to capture directional information by a phase matrix. However, many directed spectral methods use the same mapping function to obtain node representation for the outgoing information and the incoming information, and thus they may not output discriminative representation by ignoring the difference between the incoming information and the outgoing information [Rossi *et al.*, 2023].

Spatial DGNN methods are designed to first learn different mapping functions for incoming and outgoing information, and then combine two directional information to update node representation for downstream tasks. For example, DGCN [Tong *et al.*, 2020b] separately learns mapping functions for directional information based on their second-order information. Dir-GNN [Rossi *et al.*, 2023] first learns different mapping functions based on the first-order information of nodes, and then manually assigns weights to fuse these two kinds of node representation obtained from two mapping functions. To summarize, previous spatial DGNNs methods consider the difference between two kinds of directional information by two ways, *i.e.,* different mapping functions and manually setting different weights on the node representation learned from different directional information, but they still have limitations to be addressed.

First, the representation of every node is affected by its neighbors, which include an in-neighbor set and an out-neighbor set in the directed graph. The label of every node is determined by either the in-neighbor set or the out-neighbor set or even both sets, as shown in Figure 1. Hence, every node in either in-neighbors or out-neighbors should have different weights, neighbor diversity for short. However, previous spatial DGNNs assign them with the same weight. Second,

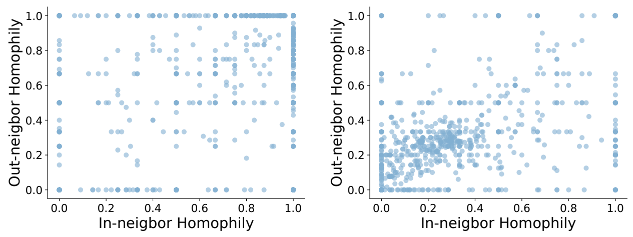


Figure 1: Visual illustration of neighbor diversity on the datasets Cora-ML (left) and Chameleon (right), where each scatter represents one node. The out-neighbor homophily is the proportion of one node and out-neighbors with the same category. The nodes above the diagonal is with high out-neighbor homophily and their labels are determined by its out-neighbor set. The labels of the node nearby are determined by both its out-neighbor set and in-neighbor set.

degree can reflect the centrality of a node and provide information about its surrounding structure. Moreover, degree information is very helpful for distinguishing two different nodes, especially in directed graphs including in-degree and out-degree. In particular, every node has different in-degree and out-degree, *i.e.,* degree diversity. However, previous spatial DGNNs ignore the degree information in directed graphs to weaken the representation ability of the node. Obviously, both neighbor diversity and degree diversity show the characteristics of nodes, *i.e.,* node diversity.

In this paper, we propose a new spatial DGNN method, namely **N**ode **D**iversity **D**irected **G**raph **N**eural **N**etwork (**NDDGNN**) shown in Figure 3, to model directional information for graph representation learning by considering node diversity. To do this, we follow the directed spatial methods to construct different mapping functions for the outgoing information and the incoming information. During the construction process, different from these previous spatial DGNNs assigning the same weight to all nodes in the same neighbor set (*i.e.,* out-neighbor set and in-neighbor set) for every node, our method assigns them with different weights by considering node diversity. Specifically, we consider neighbor diversity by defining in-Dirichlet energy and out-Dirichlet energy to encode the dissimilarity between every node and its neighbor and consider degree diversity by introducing degree embedding to encode in-degree information and out-degree information.

Compared to previous DGNNs, the main contributions of our method are summarized as follows:

- We first observe node diversity for representation learning in directed graphs and then propose a new directed GNN method considering node diversity (including neighbor diversity and degree diversity) for graph representation learning. As a result, our method is able to adaptively learn weights for every node.

- We experimentally verify on real-world datasets the superiority of our proposed method, compared to SOTA methods, in terms of node classification and link prediction.

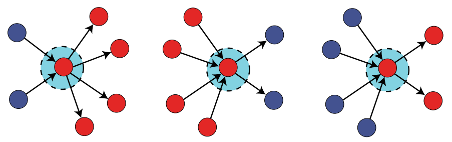


Figure 2: A case study of node diversity in the dataset Cora-ML, where different colors (*i.e.,* blue and red) represent different categories. The case study includes three nodes, *i.e.,* Node # 550 (left), Node #2162 (middle), and Node # 2714 (right).

2 Methodology

Let $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X}, \mathbf{Y})$ be a directed and unweighted graph, where $\mathbf{V} = \{v_i | i = 1, 2, \ldots, n\}$, $\mathbf{E} = \{(v_i, v_j) | v_i, v_j \in \mathbf{V}\}$, $\mathbf{X} \in \mathbb{R}^{n \times f}$, and $\mathbf{Y} \in \mathbb{R}^{n \times c}$, respectively, denote the node set, the edge set, the node feature matrix and the label matrix, and $\mathbf{A} \in \{0, 1\}^{n \times n}$ be the adjacency matrix of the graph $\mathcal{G}$, $\mathbf{D} = \sum_i A_{i,j}$ is the degree matrix, and $\tilde{\boldsymbol{\Delta}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ is the normalize Laplacian matrix. By denoting $\mathcal{N}_{i,\rightarrow}$ as the out-going neighbor set of the i-th node v_i and $\mathbf{D}_{\rightarrow} = \sum_j (\mathbf{A}_{i,j})$ as the out-degree diagonal matrix, we obtain the symmetric normalize directed out-neighbor matrix $\mathbf{S}_{\rightarrow} = \mathbf{D}_{\rightarrow}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}_{\leftarrow}^{-\frac{1}{2}}$ and row normalize directed out-neighbor matrix $\mathbf{S}'_{\rightarrow} = \mathbf{D}_{\rightarrow}^{-1} \mathbf{A}$. Similarly, we can define $\mathcal{N}_{i,\leftarrow}$, $\mathbf{D}_{\leftarrow}$, $\mathbf{S}_{\leftarrow} = \mathbf{S}_{\rightarrow}^{T}$ and $\mathbf{S}'_{\leftarrow} = \mathbf{S}'^{T}_{\rightarrow}$.

2.1 Motivation

To deal with the directional information in the directed graph, previous spatial DGNNs first separate the directional information into incoming information and outgoing information, and then learn individual mapping functions for them, *i.e.,* the outgoing mapping function and the incoming mapping function. Each function outputs one kind of representation for every node. Specifically, following DirGNN [Rossi *et al.*, 2023] to denote two directional representation as $\mathbf{S}_{\rightarrow} \mathbf{H}^{(l)} \mathbf{W}_{\rightarrow}^{(l)}$ and $\mathbf{S}_{\leftarrow} \mathbf{H}^{(l)} \mathbf{W}_{\rightarrow}^{(l)}$, where $\mathbf{W}_{\rightarrow}^{(l)}$ and $\mathbf{W}_{\leftarrow}^{(l)}$, respectively, represent the learnable weights, we fuse them by manually assigning a hyper-parameter to update the node representation $\mathbf{H}^{(l+1)}$ in the (l + 1)-th layer by:

$$\mathbf{H}^{(l+1)} = \gamma \mathbf{S}_{\rightarrow} \mathbf{H}^{(l)} \mathbf{W}_{\rightarrow}^{(l)} + (1 - \gamma) \mathbf{S}_{\leftarrow} \mathbf{H}^{(l)} \mathbf{W}_{\leftarrow}^{(l)} \quad (1)$$

where γ is the hyper-parameter. Although the weight γ is set to distinguish two kinds of directional information, it actually assigns the same weight to every node representation in the same mapping information. This leaves at least two issues to be addressed. First, the hyper-parameter γ easily results in expensive computation cost. Second, all nodes share the same weight for two kinds of directional information by ignoring node diversity.

In this paper, we propose a new spatial directed graph neural network framework shown in Figure 3 to address the above issues.

2.2 Node Diversity

There are many facts to result in node diversity. For example, neighbor diversity, degree diversity, structure diversity, e.t.c. For simplicity, in this paper, we investigate node diversity by focusing on neighbor diversity and degree diversity.

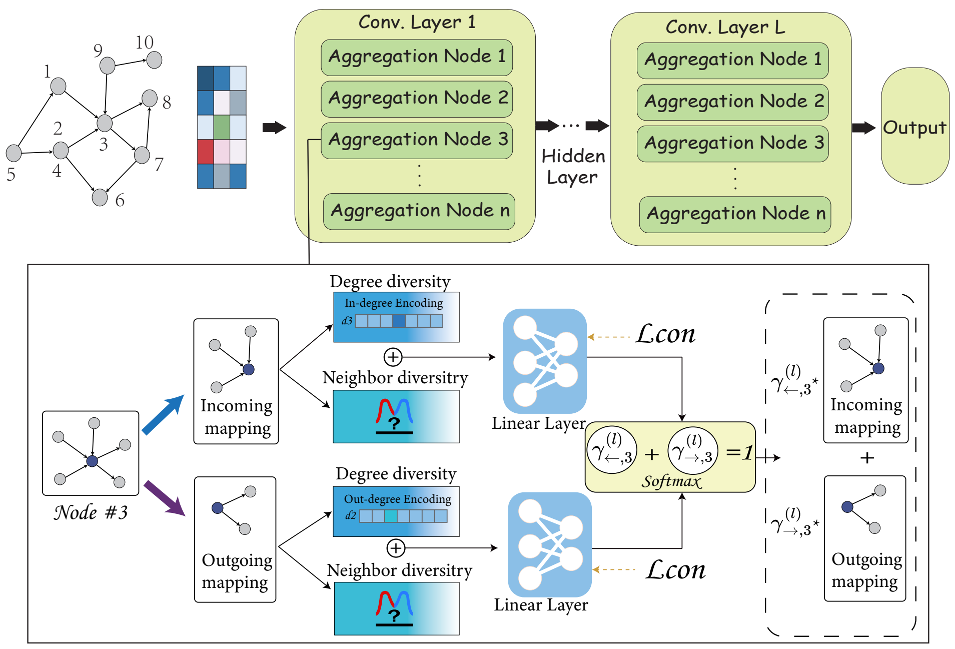


Figure 3: The flowchart of the proposed NDDGNN with a layer for node #3 as an example. Specifically, NDDGNN first maps the in-neighbors and out-neighbors of each node separately and then computes their neighbor diversity and degree diversity, respectively. After that, NDDGNN employs the combined neighbor diversity and degree diversity to learn the weights of each node as well as normalizes the weights by Softmax. NDDGNN also investigates the consistency regularization (*i.e.*, $\mathcal{L}_{con}$) to ensure the prevention of learned abnormal weights. Finally, the weights are used to fuse the in-neighbor and out-neighbor representations for updating the representation of node i.

Neighbor Diversity

Previous spatial DGNNs assign the same weight to all nodes in the same neighbor set, by ignoring neighbor diversity. Given an example in Figure 2, the label of the centric node in the left of Figure 2 is benefited from its out-neighbors because it and its out-neighbor have the same color, while the label of the centric node in the middle of Figure 2 is benefited from its in-neighbors. In this case, it is obvious that the different nodes has different weights to either in-neighbor set and out-neighbor set. However, previous spatial DGNNs assign them with the same weight. In this section, we consider neighbor diversity to assign them with different weights.

The out-neighbor set of every node is different from the set of other nodes, while the in-neighbor set of every node is different from the one of other nodes. Therefore, it is reasonable to learn different weights for the out-neighbor set and the in-neighbor set. The category of every node is related to the majority class in the neighbor set (*i.e.*, either out-neighbor set or in-neighbor set). Specifically, if the category of every node is same as the category of the majority class in the neighbor set, the weight of this neighbor set should be large for achieving small intra-class difference. Since we have label information of a part of nodes, so the feature difference between every node and its connected nodes can be used to measure the intra-class difference, and thus determining the weights of the neighbor-set. Considering Dirichlet energy [Xu *et al.*, 2023] is one of popular ways to measure the adjacent node feature difference, in this paper, we employ it to guide to learn the weights. Specifically, given the node representation $\mathbf{H} = [\mathbf{h_1}, \mathbf{h_2}, \ldots, \mathbf{h_n}] \in \mathbb{R}^{\mathbf{n}\times\mathbf{f}}$, the Dirichlet energy of l layer $E\left(\mathbf{H}^{(\mathbf{l})}\right)$ is defined as follows:

$$E(\mathbf{H}^{(l)}) = \sum_{i}^{n} \sum_{j\in\mathcal{N}_i} s'_{ij} \left\|\mathbf{h}_i^{(l)} - \mathbf{h}_j^{(l)}\right\|_2^2, \tag{2}$$

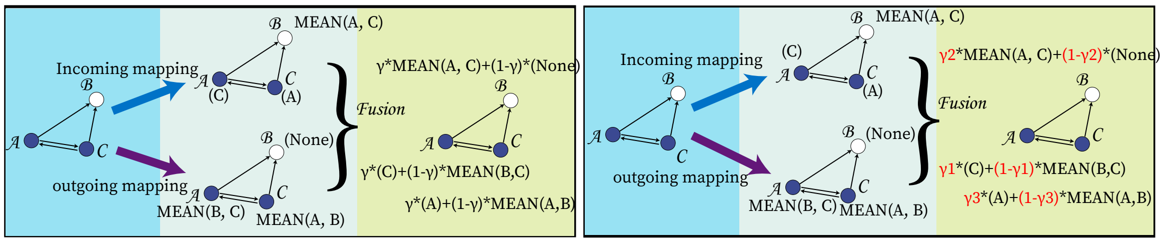
where s'_{ij} denotes weight of edges.

Since different dimensions carry different information, it is necessary to assign different weights to different dimensions. However, Dirichlet energy assigns the same weight to all dimensions. In this paper, we introduce learnable parameters in Dirichlet energy, enabling the model to assign distinct weights to different dimensions and can be adaptively adjusted according to downstream tasks, which can be written as:

$$E(\mathbf{H}^{(l)}) = \sum_{i}^{n} \sum_{j\in\mathcal{N}_i} s'_{ij} \left\|(\mathbf{h}_i^{(l)} - \mathbf{h}_j^{(l)}) \odot \mathbf{w}^{(l)}\right\|_2^2, \tag{3}$$

where $\mathbf{w}^{(l)} \in \mathbb{R}^{f\times 1}$ is a learnable parameter vector and $\odot$ is element-wise product.

Although Dirichlet energy is used to compute the whole graph, we focus on the difference between every node and its neighbor sets. To facilitate subsequent calculations and reduce complexity, we use row directed normalized adjacency matrix for s'_{ij} here (*i.e.*, $s'_{ij,\leftarrow} = \frac{1}{d_{i,\leftarrow}}$, $s'_{ij,\rightarrow} = \frac{1}{d_{i,\rightarrow}}$). Thus, for out-neighbor set of all nodes, the matrix form of out-



(a) Dir-GNN (b) Proposed

Figure 4: A case to illustrate the difference of weight assignment between previous spatial DGNN (*e.g.,* Dir-GNN [Rossi *et al.*, 2023]) and our proposed method.

Dirichlet energy $\mathbf{e}_{\rightarrow}^{(l)}$ in l layer can be defined as follows:

$$\begin{aligned}\mathbf{e}_{\rightarrow}^{(l)} &= ((\mathbf{I}+\mathbf{S}'_{\rightarrow})(\mathbf{H}^{(l)} \odot \mathbf{H}^{(l)}) \\ &\quad - 2(((\mathbf{I}+\mathbf{S}'_{\rightarrow})\mathbf{H}^{(l)}) \odot \mathbf{H}^{(l)} - \mathbf{H}^{(l)} \odot \mathbf{H}^{(l)}))\mathbf{w}_{\rightarrow}^{(l)}\end{aligned} \tag{4}$$

where $\mathbf{e}_{\rightarrow}^{(l)} \in \mathbb{R}^{n\times 1}$, which i-th element represents the out-Dirichlet energy of node i and $\mathbf{I}$ is the identity matrix. The same as in-Dirichlet energy $\mathbf{e}_{\leftarrow}^{(l)}$ for in-neighbor set:

$$\begin{aligned}\mathbf{e}_{\leftarrow}^{(l)} &= ((\mathbf{I}+\mathbf{S}'_{\leftarrow})(\mathbf{H}^{(l)} \odot \mathbf{H}^{(l)}) \\ &\quad - 2(((\mathbf{I}+\mathbf{S}'_{\leftarrow})\mathbf{H}^{(l)}) \odot \mathbf{H}^{(l)} - \mathbf{H}^{(l)} \odot \mathbf{H}^{(l)}))\mathbf{w}_{\leftarrow}^{(l)}\end{aligned} \tag{5}$$

On large-scale graphs, to reduce the time complexity, we rewrite equation (3) as:

$$E(\mathbf{H}^{(l)}) = \sum_{i}^{n} \left\| (\mathbf{h}_i^{(l)} - \sum_{j\in\mathcal{N}_i} s'_{ij}\mathbf{h}_j^{(l)}) \odot \mathbf{w}^{(l)} \right\|_2^2 \tag{6}$$

This equation calculates the difference between each node and two neighbors like equation (3) and ignores the variance within the neighbors, we prove it in Appendix. Thus, the out-Dirichlet energy in equation (4) can be write as:$\mathbf{e}_{\rightarrow}^{(l)} = (\mathbf{H}-\mathbf{S}\mathbf{H})^2\mathbf{w}_{\rightarrow}^{(l)}$, the same as in-Dirichlet energy. Here, we have extended the Dirichlet energy to adaptively characterize the differences between each node and its in-neighbors and out-neighbors in directed graphs. However, there is a problem in using the corresponding Dirichlet energy as the weights of two kinds of neighbor sets. In general, the larger the difference in features between a node and its neighbors, the more likely they belong to different categories. Therefore, directly using Dirichlet energy as weights would lead to assigning larger weights to the neighbor set with more different categories nodes and smaller weights to the neighbor sets with more same categories nodes. This is contrary to our motivation. Therefore, we add the inverse to correct them.

$$\begin{cases}\mathbf{q}_{\rightarrow}^{(l)} = -\mathbf{e}_{\rightarrow}^{(l)} \\ \mathbf{q}_{\leftarrow}^{(l)} = -\mathbf{e}_{\leftarrow}^{(l)},\end{cases} \tag{7}$$

where every element of $\mathbf{q}_{\rightarrow}^{(l)} \in \mathbb{R}^{n\times 1}$ and $\mathbf{q}_{\leftarrow}^{(l)} \in \mathbb{R}^{n\times 1}$ are adaptive weights of the two kinds of neighbor set for every node, respectively.

Different from current spatial DGNNs manually set the weight, Eq. (7) automatically learning $\mathbf{q}_{\rightarrow}^{(l)}$ and $\mathbf{q}_{\leftarrow}^{(l)}$ by considering the difference between every node and its neighbors by in-Dirichlet energy and out-Dirichlet energy. As a result, for all nodes, the neighbor sets with smaller feature differences (*i.e.,* more similar categories), larger weights will be assigned compared to the other neighbor set. *i.e.,* characterizing neighbor diversity for every node. However, ever node is related to its neighbors as well as its degree, which also needs to be paid attention on node diversity.

Degree Diversity

Previous spatial DGNNs combine the in-neighbors and out-neighbors without considering the in-degree and out-degree. However, degree information can improve GNNs's ability to distinguish the different nodes, especially in directed graphs since there are two types of degree. For example, as shown in Figure 2, for simply analysis, we assume that nodes of the same category have the same features. The previous spatial DGNNs utilize the equation 1 to update nodes representation for nodes in left and right of Figure 2, the two nodes would obtain the same representations. However these two nodes are not the same node because their number of neighbors are different (*i.e.,* in-degree and out-degree). Therefore, without considering degree information will weaken the node representation ability of DGNNs.

In this paper, to overcome above issue, we propose to learn degree embedding for integrating degree information with the process of representation learning. Specifically, denoting $\mathbf{Deg}_{\leftarrow} \in \mathbb{R}^{n\times f}$ and $\mathbf{Deg}_{\rightarrow} \in \mathbb{R}^{n\times f}$, as learneable in-degree embedding and out-degree embedding, respectively, Eq. (7) becomes:

$$\begin{cases}\mathbf{q}_{\rightarrow}^{(l)} = -\mathbf{e}_{\rightarrow}^{(l)} + \mathbf{Deg}_{\rightarrow}\mathbf{w}_{\rightarrow}^{(l)} \\ \mathbf{q}_{\leftarrow}^{(l)} = -\mathbf{e}_{\leftarrow}^{(l)} + \mathbf{Deg}_{\leftarrow}\mathbf{w}_{\leftarrow}^{(l)},\end{cases} \tag{8}$$

Based on Eq. (8), nodes with the same in-degree have the same in-degree embedding. Similarly, nodes with the same out-degree also have the same out-degree embedding. Hence, when two nodes share same in out-/in-degrees and out-/in-neighbor differences, their weights during aggregation will be similar. In contrast, even if two nodes have very similar features for themselves and their neighbors, as long as their degree information is different they will not be mapped to the same point in the feature space (*i.e.,* the model has the ability to distinguish between the two nodes). However, previous spatial DGNNs by [Rossi *et al.*, 2023; Tong *et al.*, 2020a] assign the same weight to both nodes in all cases.

Overall Architecture

To consider the node diversity in directed graph, we propose a simple but effective framework called NDDGNN as shown in Figure 3. Specifically, for a directed graph, we learn the weights vector (*i.e.,* $\mathbf{q}_{\leftarrow}^{(l)}, \mathbf{q}_{\rightarrow}^{(l)} \in \mathbb{R}^{n\times 1}$), where $\mathbf{q}_{i,\leftarrow}^{(l)}$ and $\mathbf{q}_{i,\rightarrow}^{(l)}$ are in-neighbor and out-neighbor weight of node i in layer l, respectively. Then through the softmax normalize to ensure the $\mathbf{q}_{\leftarrow}^{(l)} + \mathbf{q}_{\rightarrow}^{(l)} = [1, 1, \cdots, 1]^T$ as follow:

$$diag(\mathbf{\Gamma}_{\rightarrow}^{(l)}) = \frac{\exp(\mathbf{q}_{\rightarrow}^{(l)})}{\exp(\mathbf{q}_{\rightarrow}^{(l)}) + \exp(\mathbf{q}_{\leftarrow}^{(l)})}, \tag{9}$$

where $diag(\mathbf{\Gamma}_{\rightarrow}^{(l)})$ stands for the diagonal value of diagonal matrix $\mathbf{\Gamma}_{\rightarrow}^{(l)}$. Moreover, the equation holds, *i.e.,* $\mathbf{\Gamma}_{\leftarrow}^{(l)} = \mathbf{I} - \mathbf{\Gamma}_{\rightarrow}^{(l)}$.

In this paper, we take into account node diversity to overcome the issues in previous spatial DGNNs by using Eq. (10) to update node representation $\mathbf{H}^{(l+1)}$ in the $(l+1)$-th layer.

$$\mathbf{H}^{(l+1)} = \mathbf{\Gamma}_{\rightarrow}^{(l)}\mathbf{S}_{\rightarrow}\mathbf{H}^{(l)}\mathbf{W}_{\rightarrow}^{(l)} + \mathbf{\Gamma}_{\leftarrow}^{(l)}\mathbf{S}_{\leftarrow}\mathbf{H}^{(l)}\mathbf{W}_{\leftarrow}^{(l)}, \tag{10}$$

where $\mathbf{\Gamma}_{\leftarrow}^{(l)} \in \mathbb{R}^{n\times n}$ is the diagonal matrix consisting of $[\mathbf{\Gamma}_{(1,1)\leftarrow}^{(l)}, \mathbf{\Gamma}_{(2,2)\leftarrow}^{(l)}, \ldots, \mathbf{\Gamma}_{(n,n)\leftarrow}^{(l)}]$ as the diagonal elements in l layer, including both **neighbor diversity** and **degree diversity**, the same as $\mathbf{\Gamma}_{\rightarrow}$. With this form, it is possible to achieve that each node has different weights assigned to outgoing and incoming neighbors. The difference between previous spatial DGNNs (*e.g.,* DirGNN) and our proposed method is shown in Figure 4, where previous spatial DGNNs is a special case of our proposed method in Eq. (10).

By aggregating the neighbors' representation and the node representation in the previous layer, in this paper, node representation $\mathbf{H}^{(l+1)}$ in the $(l+1)$-th layer of our proposed method is updated by:

$$\mathbf{H}^{(l+1)} = \alpha\mathbf{H}^{(l)} + \mathbf{\Gamma}_{\rightarrow}^{(l)}\mathbf{S}_{\rightarrow}\mathbf{H}^{(l)}\mathbf{W}_{\rightarrow}^{(l)} + \mathbf{\Gamma}_{\leftarrow}^{(l)}\mathbf{S}_{\leftarrow}\mathbf{H}^{(l)}\mathbf{W}_{\leftarrow}^{(l)}, \tag{11}$$

where α is a hyper-parameter to preserve the information of the previous layer. We show the model's real running time in the Appendix.

2.3 Consistency Regularization.

In our method, we decompose the directional information into incoming information and outgoing information. We expect that weights learnt for incoming information follow the same distribution, while weights learnt from outgoing information also follow the same distribution. To do this, we propose the consistency regularization to mitigate and avoid learning the abnormal weights that deviate from the distribution. Specifically, we first calculate the average of all weights for either incoming information or outgoing information by:

$$\overline{\gamma}_{\rightarrow} = \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{\Gamma}_{(i,i),\rightarrow}, \quad \overline{\gamma}_{\leftarrow} = \frac{1}{n}\sum_{i=0}^{n-1}\mathbf{\Gamma}_{(i,i),\leftarrow}. \tag{12}$$

We then propose to minimize the distance between $\overline{\gamma}_{\rightarrow}$ and $\mathbf{\Gamma}_{i,\rightarrow}$, as well as the distance between $\overline{\gamma}_{\leftarrow}$ and $\mathbf{\Gamma}_{i,\leftarrow}$ by:

$$\mathcal{L}_{con} = \frac{1}{n}\sum_{i=0}^{n}\left\|\overline{\gamma}_{\rightarrow} - \mathbf{\Gamma}_{(i,i),\rightarrow}\right\|_2^2 + \frac{1}{n}\sum_{i=0}^{n}\left\|\overline{\gamma}_{\leftarrow} - \mathbf{\Gamma}_{(i,i),\leftarrow}\right\|_2^2. \tag{13}$$

2.4 Objective Function

In each epoch, we employ both the cross entropy loss and the consistency regularization loss in Eq. (13) to conduct representation learning. The final objective function of the proposed method is formulated as:

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda\mathcal{L}_{con}, \tag{14}$$

where $\mathcal{L}_{sup}$ is the cross entropy loss and λ is a hyper-parameter to balance two losses.

3 Experiments

3.1 Experimental Setting

Datasets

We evaluate the effectiveness of the proposed method on 2 homophilic datasets and 5 heterophilic datasets. Homophilic datasets include Cora-ML and Citeseer-Full [Bojchevski and Günnemann, 2018]. Heterophilic datasets include Chameleon, Squirrel [Pei *et al.*, 2020], Roman-Empire [Platonov *et al.*, 2023], Arxiv-Year [Leskovec *et al.*, 2005], and Snap-Patents [Leskovec and Krevl, 2014]. In particular, Arxiv-Year and Snap-Patents are **large-scale** datasets. More details of the used datasets can be found in Appendix.

Comparison Methods

The comparison methods include 3 traditional undirected graph methods (*i.e.,* GCN [Kipf and Welling, 2017], GAT [Velickovic *et al.*, 2018], and graphSAGE [Hamilton *et al.*, 2017]), 6 state-of-the-art undirected graph methods (*i.e.,* H2GCN [Zhu *et al.*, 2020], GPRGNN [Chien *et al.*, 2021], LINKX [Lim *et al.*, 2021], FSGNN [Maurya *et al.*, 2021], ACM-GCN [Luan *et al.*, 2022], DloGNN [Li *et al.*, 2022], and Gradient Gating [Rusch *et al.*, 2023]), and three state-of-the-art directed graph methods (*i.e.,* DiGCN [Tong *et al.*, 2020a], MagNet [Zhang *et al.*, 2021] and Dir-GNN [Rossi *et al.*, 2023]).

Evaluation Protocol

To evaluate the effectiveness of the proposed method, we follow the settings in [Rossi *et al.*, 2023] to conduct node classification tasks and link prediction tasks. Specifically, for the node classification task, we split all datasets in Dir-GNN [Rossi *et al.*, 2023] and the detail can be found in the Appendix. For directed graph link prediction task, we remove 10% of edges for testing, 5% for validation, and use the rest of the edges for training. We randomly generate 10 splits for each graph and the graph connectivity is maintained during the splitting.

Setting-Up

We conduct all experiments on a server with Nvidia RTX 4090 (24GB memory each). We employ early stopping on the validation accuracy for each experiment. Moreover, we conduct each experiment on ten random seeds and report the average results.

In the proposed method, we optimize all parameters by the Adam optimization [Kingma and Ba, 2015] with the learning rate in the range of $\{0.005, 0.01\}$ and set the weight

	Small-scale Datasets					Large-scale Datasets	
Datasets	Cora-ML	Citeseer-Full	Chameleon	Squirrel	Roman-Empire	Arxiv-Year	Snap-Patents
MLP	77.48±1.23	80.01±1.23	46.36±2.57	34.14±1.94	65.76±0.42	36.70±0.15	31.34±0.21
GCN	52.37±1.77	54.65±1.22	64.82±2.24	53.43±2.01	73.69±0.74	46.02±0.26	51.02±0.06
GAT	54.12±1.56	55.15±1.31	45.56±3.16	39.14±2.88	71.16±0.63	45.12±0.28	OOM
H2GCN	62.86±1.45	68.34±1.36	59.39±0.98	37.90±0.02	60.11±0.52	49.09±0.10	OOM
GPRGNN	68.88±1.66	70.12±1.24	62.85±2.90	54.35±0.87	64.85±0.27	45.07±0.21	40.19±0.03
LINKX	72.32±1.41	78.75±1.34	68.42±1.38	61.81±1.80	37.55±0.36	56.00±0.17	61.95±0.12
FSGNN	67.51±1.65	66.35±1.16	78.27±1.28	74.10±1.89	79.92±0.56	50.47±0.21	65.07±0.03
ACM-GCN	69.96±1.53	73.61±1.32	74.76±2.20	67.40±2.21	69.66±0.62	47.37±0.59	55.14±0.16
GloGNN	73.78±1.69	76.13±1.14	57.88±1.76	71.21±1.84	59.63±0.69	54.79±0.25	62.09±0.27
Grand.Gating	76.63±1.63	80.36±1.13	71.40±2.38	64.26±2.38	82.16±0.78	63.30±1.84	69.50±0.39
Di-GCN	87.36±1.06	92.75±0.75	52.24±3.65	37.74±1.54	52.71±0.32	OOM	OOM
MagNet	85.26±1.05	93.38±0.89	58.22±2.87	39.01±1.93	88.07±0.27	60.29±0.27	OOM
Dir-GNN	84.45±1.69	92.79±0.59	79.71±1.26	75.31±1.92	91.23±0.32	64.08±0.26	73.92±0.05
NDDGNN	**88.14±1.28**	**94.17±0.58**	**79.79±1.04**	**75.38±1.95**	**91.76±0.27**	**65.02±0.32**	**74.15±0.04**

Table 1: Node classification accuracy (%) on directed graphs. The best results are in bold. OOM indicates out of memory.

Datasets	Cora-ML	Citeseer-Full	Chameleon	Squirrel
GCN	76.24±4.25	71.16±4.41	86.03±1.53	90.64±0.49
GAT	64.58±12.28	67.28±2.90	85.12±1.57	90.37±0.45
MagNet	82.20±3.36	85.89±4.61	86.28±1.23	90.13±0.53
Dir-GNN	82.23±4.01	70.62±6.56	82.76±2.72	89.17±0.52
NDDGNN	**83.04±9.56**	**87.93±3.92**	**90.30±1.82**	**92.40±1.01**

Table 2: Link prediction accuracy (%) on directed graphs. The best results are in bold.

+N	+Deg	$+\mathcal{L}_{con}$	Cora-ML	Citeseer-Full	Chameleon	Squirrel
-	-	-	52.37±1.77	54.65±1.22	64.82±2.24	53.43±2.01
✓	-	-	81.98±1.19	93.56±0.67	72.87±1.32	60.30±2.27
-	✓	-	86.52±1.35	93.36±0.54	73.44±2.12	61.88±5.62
✓	✓	-	86.89±1.36	94.11±0.59	76.84±1.60	75.20±2.02
✓	-	✓	83.14±1.53	94.06±0.68	75.48±1.74	73.73±8.03
-	✓	✓	87.58±1.17	93.89±0.61	78.75±1.81	74.59±1.88
✓	✓	✓	**88.14±1.28**	**94.17±0.58**	**79.79±1.04**	**75.38±1.95**

Table 3: Ablation study. $+\mathbf{N}$ denotes neighbor diversity, $+\mathbf{Deg}$ denotes degree diversity.

decay as 0. Moreover, we set the number of model layers in the range of $\{4, 5, 6\}$, set the dropout in the range of $\{0.0, 0.35, 0.5, 0.6\}$, and set the size of the hidden unit in the range of $\{32, 128, 256\}$. We set α for preserving the representation of the previous layer in the range of $\{0.0, 0.3, 0.5, 0.8, 1.0\}$, and set λ in our regularization term in the range of $\{0.0, 0.1, 0.2, 0.9\}$. Notably, we employ the same parameters for both node classification and link prediction tasks. In addition, we set the parameters of all comparison methods according to the original literature so that they output the best performance.

3.2 Results Analysis

Node Classification

We evaluate the effectiveness of our method on the node classification task and report the classification accuracy of all methods on all datasets in Table 1. Obviously, the proposed method achieves the best performance on all datasets.

First, compared with the undirected graph methods (*i.e.,* GCN, GAT, graph-SAGE, H2GCN, GPRGNN, LINKX, FSGNN, ACM-GCN, DloGNN, and Grand.Gating), the proposed method consistently obtains substantial improvements. For example, the proposed method on average improves by 11.5%, 13.81%, 8.38%, 11.12%, 1.72%, 4.65%, and 9.6%, compared with the best undirected graph method Grand.Gating on all datasets. This can be attributed to the fact that these state-of-the-art undirected graph methods are primarily designed to operate on symmetric graphs, thus their architectures might not be well-suited to effectively handle the inherent directional information present in directed graphs. More seriously, directly applying undirected graph methods to directed graph datasets can even introduce negative effects. For example, MLP outperforms many undirected graph methods (*e.g.,* GCN, GAT, H2GCN, and GPRGNN) on directed datasets (*e.g.,* Cora-ML and Citeseer). This suggests that these undirected graph methods act as a negative impact for the utilization of the graph structure.

Second, compared with the directed graph methods, the proposed method achieves the best performance, followed by Dir-GNN, MagNet and Di-GCN. For example, the proposed method on average improves by 0.99%, compared with the best directed graph method Dir-GNN on all datasets. This suggests that the proposed framework that it is effective to take node diversity into account, compared to directed graph methods.

Link Prediction

We evaluate the effectiveness of the proposed method on the link prediction tasks by reporting the results (*i.e.,* prediction accuracy) on four datasets in Table 2. We can observe that our method consistently achieves the best performance compared to the directed graph methods. For example, the proposed method average improves by 0.84%, 2.04%, 4.02%,

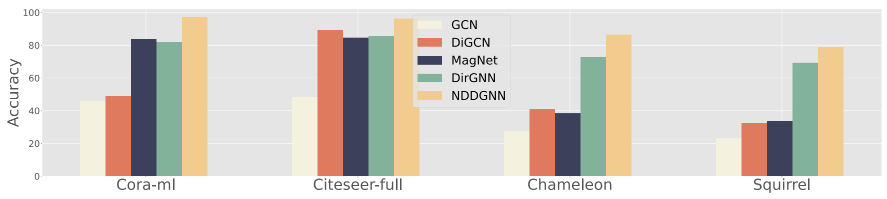


Figure 5: Classification accuracy of the top 5% of nodes with the largest degree differences under all directed graph methods and GCN on four datasets.

and 2.27%, compared with the best directed graph method MagNet on all datasets. This indicates that our method can achieve good performance in structure-dominated tasks.

Effectiveness on Nodes with Complex Structure

We further evaluate the effectiveness of the proposed method to cope with complex structure (*e.g.,* nodes with extremely different out-degree and in-degree) in directed graphs. To do this, we select the top 5% nodes with the largest out-degree-in-degree differences and visualize the classification performance on four datasets in Figure 5. First, the undirected graph method GCN almost loses its representation ability for complex structural cases in directed graphs, especially in the Chameleon and Squirrel datasets where the maximum degree difference is more than a thousand. This is due to the inability of undirected graph methods to capture such differences. Second, the proposed NDDGNN achieves the best performance on complex nodes compared to other directed graph methods, which benefits from node diversity.

Ablation Study

The proposed method mainly consists of three components, *i.e.,* neighbor diversity (*i.e.,* $+\mathbf{N}$), degree diversity (*i.e.,* $+\mathbf{Deg}$) and consistency loss (*i.e.,* $+\mathcal{L}_{con}$). To verify the effectiveness of each component in the proposed method, we investigate the performance of all variants (except $\mathcal{L}_{con}$, since $\mathcal{L}_{con}$ depends on $+\mathbf{N}$ or $+\mathbf{Deg}$) on the node classification task by reporting the results in Table 3.

First, the variant method that considers both the neighbour diversity and the degree diversity leads to a large performance improvement, compared with the variant method considering only one diversity of them. This suggests that both neighborhood diversity and degree diversity are critical and not interchangeable. Second, the variant method that considers the consistency loss averagely improves by 3.45%, compared with the counterpart variant methods without the consistency loss. This indicates that the consistency loss avoids learning the abnormal weights to some extent. Therefore, the effectiveness of each component of the proposed method is verified.

4 Related Work

We briefly review the study related to this work, *i.e.,* directed graph mining methods and adaptive node level methods.

4.1 Directed Graph Mining Methods

[Ma *et al.*, 2019] constructs a directed Laplacian matrix based on identities that involve the random walk matrix and its stationary distribution.

DGCN [Tong *et al.*, 2020b] leverages both the first-order and second-order proximity information to collectively describe and distinguish the neighborhood of each vertex from those vertices. MotifNet [Monti *et al.*, 2018] is capable of dealing with directed graphs by exploiting local graph motifs. MagNet [Zhang *et al.*, 2021] utilizes the magnetic Laplacian matrix to encode undirected geometric structures in the magnitudes and directional information in the phases. DiGCN [Tong *et al.*, 2020a] defines the Laplacian matrix of the directed graph based on PageRank. [Tong *et al.*, 2021] extends DiGCN to obtain different Laplacian matrices under different hyper-parameters. Recently, Dir-GNN [Rossi *et al.*, 2023] proposes first aggregating outgoing and incoming information separately, and then simply fusing the information of outgoing and incoming information with a fixed hyper-parameter weight for all nodes.

4.2 Graph Neural Networks With Adaptive Node Level Methods

GNN [Kipf and Welling, 2017; Yang *et al.*, 2023b; Yang *et al.*, 2023a; Bi *et al.*, 2022a] have shown excellent performance on graph representation learning. Some methods utilize the adaptive weight assignment technique to let nodes learn the appropriate representation. Such as GAT [Velickovic *et al.*, 2018] adaptively assigns weights to the neighbors of each node using attention mechanisms, SA-GNN [Huang *et al.*, 2023b] adopts different aggregation methods for homophilic and heterophilic neighbors for each node by classifying neighbor labels. Similarly, GBK-GNN [Du *et al.*, 2022] uses the bi-kernel to go through the homophily information of the modeled neighbors for each node. MM-GNN [Bi *et al.*, 2023] models the multi-order moments' information for every node. However, the previous methods did not consider the complex situation of having two kinds of neighbors and two degrees in a directed graph.

5 Conclusion

In this paper, we proposed a new directed GNN method to deal with the directional information by considering the node diversity. For specificity, the node diversity is only composed of neighbor diversity and degree diversity. Moreover, our method investigates two kinds of node diversity to guide the assignments of node-level weights. Experimental results demonstrated the effectiveness of our proposed method, compared to SOTA methods in terms of node classification and link prediction tasks.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFA1004100.

References

[Bi *et al.*, 2022a] Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophily graphs better fit gnn: A graph rewiring approach. *arXiv preprint arXiv:2209.08264*, 2022.

[Bi *et al.*, 2022b] Wendong Bi, Bingbing Xu, Xiaoqian Sun, Zidong Wang, Huawei Shen, and Xueqi Cheng. Company-as-tribe: Company financial risk assessment on tribe-style graph with hierarchical graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2712–2720, 2022.

[Bi *et al.*, 2023] Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Mm-gnn: Mix-moment graph neural network towards modeling neighborhood feature distribution. In *ACM International Conference on Web Search and Data Mining(WSDM)*, pages 132–140, 2023.

[Bojchevski and Günnemann, 2018] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2018.

[Brin, 1998] Sergey Brin. The pagerank citation ranking: bringing order to the web. volume 98, pages 161–172, 1998.

[Chien *et al.*, 2021] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2021.

[Du *et al.*, 2022] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *ACM Web Conference*, pages 1550–1558, 2022.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems(Neurips)*, pages 1025–1035, 2017.

[Huang *et al.*, 2023a] Jincheng Huang, Lun Du, Xu Chen, Qiang Fu, Shi Han, and Dongmei Zhang. Robust mid-pass filtering graph convolutional networks. In *ACM Web Conference(WWW)*, pages 328–338, 2023.

[Huang *et al.*, 2023b] Jincheng Huang, Ping Li, Rui Huang, Na Chen, and Acong Zhang. Revisiting the role of heterophily in graph representation learning: An edge classification perspective. *ACM Transactions on Knowledge Discovery from Data*, 2023.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations(ICLR)*, 2015.

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

[Leskovec and Krevl, 2014] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection, 2014.

[Leskovec *et al.*, 2005] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *ACM International conference on Knowledge discovery in data mining(SIGKDD)*, pages 177–187, 2005.

[Li *et al.*, 2022] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning(ICML)*, pages 13242–13256. PMLR, 2022.

[Lim *et al.*, 2021] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. volume 34, pages 20887–20902, 2021.

[Luan *et al.*, 2022] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems(Neurips)*, 35:1362–1375, 2022.

[Ma *et al.*, 2019] Yi Ma, Jianye Hao, Yaodong Yang, Han Li, Junqi Jin, and Guangyong Chen. Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990*, 2019.

[Maurya *et al.*, 2021] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Improving graph neural networks with simple architecture design. *arXiv preprint arXiv:2105.07634*, 2021.

[Mo *et al.*, 2023] Yujie Mo, Yajie Lei, Jialie Shen, Xiaoshuang Shi, Heng Tao Shen, and Xiaofeng Zhu. Disentangled multiplex graph representation learning. In *ICML*, volume 202, pages 24983–25005, 2023.

[Mo *et al.*, 2024] Yujie Mo, Feiping Nie, Ping Hu, Heng Tao Shen, Zheng Zhang, Xinchao Wang, and Xiaofeng Zhu. Self-supervised heterogeneous graph learning: a homophily and heterogeneity view. In *ICLR*, 2024.

[Monti *et al.*, 2018] Federico Monti, Karl Otness, and Michael M Bronstein. Motifnet: a motif-based graph convolutional network for directed graphs. In *2018 IEEE Data Science Workshop(DSW)*, pages 225–228, 2018.

[Pei *et al.*, 2020] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International*

Conference on Learning Representations, (ICLR). OpenReview.net, 2020.

[Platonov *et al.*, 2023] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.

[Rossi *et al.*, 2023] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M. Bronstein. Edge directionality improves learning on heterophilic graphs. In *The Second Learning on Graphs Conference(LOG)*, 2023.

[Rusch *et al.*, 2023] T. Konstantin Rusch, Benjamin Paul Chamberlain, Michael W. Mahoney, Michael M. Bronstein, and Siddhartha Mishra. Gradient gating for deep multi-rate learning on graphs. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2023.

[Tong *et al.*, 2020a] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. Digraph inception convolutional networks. *Advances in neural information processing systems(Neurips)*, 33:17907–17918, 2020.

[Tong *et al.*, 2020b] Zekun Tong, Yuxuan Liang, Changsheng Sun, David S Rosenblum, and Andrew Lim. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970*, 2020.

[Tong *et al.*, 2021] Zekun Tong, Yuxuan Liang, Henghui Ding, Yongxing Dai, Xinke Li, and Changhu Wang. Directed graph contrastive learning. *Advances in neural information processing systems(Neurips)*, 34:19580–19593, 2021.

[Velickovic *et al.*, 2018] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2018.

[Xu *et al.*, 2023] Lei Xu, Lei Chen, Rong Wang, Feiping Nie, and Xuelong Li. Joint feature and differentiable k-nn graph learning using dirichlet energy. *arXiv preprint arXiv:2305.12396*, 2023.

[Yang *et al.*, 2023a] Xihong Yang, Yue Liu, Sihang Zhou, Siwei Wang, Wenxuan Tu, Qun Zheng, Xinwang Liu, Liming Fang, and En Zhu. Cluster-guided contrastive graph clustering network. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, volume 37, pages 10834–10842, 2023.

[Yang *et al.*, 2023b] Xihong Yang, Cheng Tan, Yue Liu, Ke Liang, Siwei Wang, Sihang Zhou, Jun Xia, Stan Z Li, Xinwang Liu, and En Zhu. Convert: Contrastive graph clustering with reliable augmentation. In *Proceedings of the 31st ACM International Conference on Multimedia (MM)*, pages 319–327, 2023.

[Zhang *et al.*, 2021] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs. *Advances in neural information processing systems(Neurips)*, 34:27003–27015, 2021.

[Zhu *et al.*, 2020] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in neural information processing systems(Neurips)*, 2020.