THE IMPACT OF NEIGHBORHOOD DISTRIBUTION IN GRAPH CONVOLUTIONAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Convolutional Networks (GCNs) which aggregate information from neighbors to learn node representation, have shown excellent ability in processing graph-structured data. However, it is inaccurate that the notable performance of GCNs tends to depend on strong homophily assumption of networks, since GCNs can also perform well over some heterophilous graphs. Thus the impact of homophily on GCNs needs to be reconsidered. In this paper, we study what influences the aggregation of GCNs from the perspective of neighborhood distribution. Theoretical and empirical analysis is provided to reveal that the distinguishability of neighborhood distribution plays a more important role in the performance of GCN than homophily. Furthermore, we address that neighborhood structure and neighborhood range are two key factors for GCNs to promote neighborhood distinguishability. Based on the conclusion, we propose an improved graph convolution network (GCN-PND) including updating graph topology based on the similarity between local neighborhood distribution of nodes and designing extensible aggregation from multi-hop neighbors. We did extensive experiments on graph benchmark datasets to analyze the superiority of the proposed method. The experimental results demonstrate that GCN-PND is more effective on heterophilous datasets than most of existing state-of-the-art GCN methods.

1 INTRODUCTION

Graph-structured data widely exists in real life, such as social networks, citation networks and biological networks. Since the relationship between nodes is complicated and irregular, graph representation learning has received extensive attention. Especially, Graph Convolutional Networks(GCNs) (Bruna et al., 2014; Atwood & Towsley, 2016; Defferrard et al., 2016) are designed to learn expressive graph representation by aggregating information from neighborhood, which have been effectively applied to multiple tasks, such as node classification (Kipf & Welling, 2017; Li et al., 2016), graph classification (Ying et al., 2018; Ma et al., 2019) and link prediction (Zhang & Chen, 2018; You et al., 2019).

However, there is a critical weakness that restricts the expressive ability of most existing GCNs on general graph-structured data. A prevalent assumption that most GCNs seem to be tailor-made to work on homophilous graphs (McPherson et al., 2001), where nodes within the same classes tend to form edges, is not rigorous enough. Actually, GCNs can perform well on some heterophilous graphs with low homophily, which breaks the assumption. Further, heterophilous graphs, where nodes from different classes are more likely to be connected to each other, are so ubiquitous in real-world that we can't ignore them. For example, a large number of people tend to connect with people within the opposite gender in dating networks (Pandit et al., 2007), and amino acids from different classes are more likely to be connected in numerous protein structures (Zhu et al., 2020). Unfortunately, the performance of GCNs on most heterophilous graphs is poor. To solve the problem, many scholars have studied GCNs from two aspects, i.e., exploring the influencing factors and designing improved GCN methods for heterophilous graphs.

For exploring factors affecting the performance of GCNs, several existing works (Zhu et al., 2020; Lim et al., 2021; Zhu et al., 2021) posit that many GCNs implicitly rely on strong homophily and attempt to overcome the weakness of GCNs in heterophilous graphs. But recent work (Ma et al., 2021) empirically finds that the graph convolutional network (GCN) (Kipf & Welling, 2017) is

actually able to match or outperform heterophily-specific models on some heterophilous graphs after fine hyperparameters tuning. It means that there exists a type of "good" heterophily, which have high distinguishability of neighborhood distribution. However, the definition of Cross-Class Neighborhood Similarity (CCNS) in this paper overemphasizes the role of graph structure, which is not applicable to all heterophilous graphs. Therefore, it is still worth thinking about what limits the performance of GCNs on this type of graphs.

To enhance the performance of GCNs on heterophilous graphs, several improved methods have been developed. For example, in Geom-GCN (Pei et al., 2020; Jin et al., 2021), neighbors are selected from the whole graph using node feature itself for supplement to the original graph. (Zhu et al., 2020) proposed that higher-order neighborhoods help in the heterophily settings However, the reason why the methods were effective for partial heterophilous graphs is not deeply explored and analyzed. Furthermore, we note that several methods (Zhu & Koniusz, 2021; Chien et al., 2021; Sun et al., 2021) design deep graph convolutional networks to extract knowledge from high-order neighbors, which have achieved good performance in homophilous graphs. However, these methods typically do not perform well on heterophilous graphs. Besides, many experimental results (Chen et al., 2020; Feng et al., 2020) demonstrate that it is not a key factor to promote accuracy by blindly expanding the search scope of neighborhood, and an appropriate neighborhood range can help improve performance. Therefore, it is an important issue to improve the performance of GCNs from the perspective of related factors.

Despite the theoretical and practical advantages of the above-mentioned works, what affects the performance of GCNs on heterophilous graphs is still unclear for them, which needs to be further studied. Therefore, in this paper, we provide theoretical and experimental analysis to reveal that the distinguishability of neighborhood distribution is a more important factor affecting performance than homophily. Furthermore, we find that if neighborhood structure and neighborhood range around nodes are improved, the distinguishability of neighborhood distribution can be enhanced. Based on the conclusions, we propose an improved GCN model for promoting neighborhood distinguishability (GCN-PND). Its core steps are the graph topology updating based on local neighborhood distribution and adaptive neighborhood aggregation from multi-hop neighbors of each node. Extensive experiments on graph benchmark datasets demonstrate the superiority of our method.

2 Notations and Preliminaries

Consider a graph $G = (V, E)$ with n nodes and m edges. The nodes are described by the feature matrix $X \in \mathbb{R}^{n \times f}$, where f is the dimension of node features. The label matrix $Y \in \mathbb{R}^{n \times c}$ indicates which class each node belongs to. Further, $\mathcal{C}$ denotes the set of classes generated from label information so that the set of nodes with a label c can be denoted as $\mathcal{C}_c$. Let A denote the adjacency matrix and D is the diagonal degree matrix of A. $\widetilde{A} = A + I$ stands for the adjacency matrix with self-loops in the graph and $\widetilde{D} = D + I$ is degree matrix of $\widetilde{A}$. And $\hat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$ denotes the symmetric normalized adjacency matrix with self-loops.

Graph Convolutional Network. GCN(Kipf & Welling, 2017) defines a common convolution operation to aggregate direct neighbor information. A GCN layer can be described as follows:

$$H^{(l+1)} = \sigma\left(\hat{A}H^{(l)}W\right), \tag{1}$$

where W is the trainable parameter matrix for feature transformation. σ is the ReLU operation for nonlinear mapping. A two-layer GCN can capture neighborhood information within two hops.

Homophily. Homophily of edges in graphs is typically defined based on the probability of edge connection between nodes within the same class. In accordance with intuition following (Zhu et al., 2020), the homophily ratio of edges is the fraction of edges in a graph that connect nodes with the same class label, described by:

$$h = \frac{1}{|E|}\sum_{(i,j)\in E}\mathbb{1}\left(y_i = y_j\right), \tag{2}$$

where $|E|$ is the number of edges. $\mathbb{1}(\cdot)$ denotes the indicator function. A graph with high edge homophily ratio means that nodes tend to connect within the same class (strong homophily), whereas a graph with h close to 0 indicates nodes from different classes are more likely to connect edges (strong heterophily).

3 IMPACT OF NEIGHBORHOOD DISTRIBUTION

The aggregation in GCNs has been shown to be able to capture and discriminate some local graph topological and structural information (Xu et al., 2019; Morris et al., 2019). Although a recent study (Ma et al., 2021) shows that GCN can work well on some heterophilous graphs, the performance of GCNs is lower than that of MLP in the face of most heterophilous networks. It shows that it is hard for GCNs to recognize these local structure information. To make matters worse, graph structure plays a negative role in the process of feature aggregation within these graphs. In order to analyze the influence of graph topology on node feature distribution in neighborhood aggregation, we propose to use neighborhood distribution including neighborhood structure distribution (NSD) and node feature distribution (NFD). For example, given node i in Figure 1a, where different colors signify nodes with different classes. It is obvious that aggregated feature of node i is decided by the graph structure and range of neighborhood around node i, as well as the node feature obtained from the original node feature distribution. Next, we define neighborhood distribution and study how neighborhood distribution affects the performance of aggregated features.

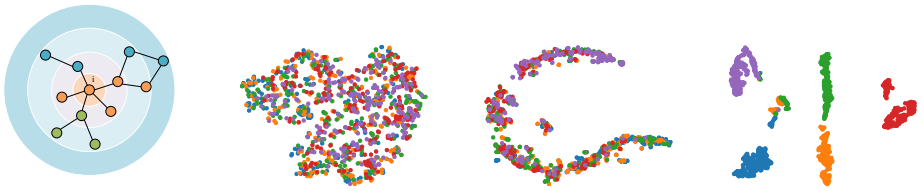
Consider a graph G, where node i has label Y_i and feature X_i. We define neighborhood distribution of nodes consisting of neighborhood structure distribution and node feature distribution. The neighborhood structure distribution around node i is expressed as $\mathcal{S}_i\left(\theta, J\right)$, where θ denotes the hidden parameter of distribution and J is the range of neighborhood (within J-hop neighborhood). Obviously, there are two versions of neighborhood structure distribution according to whether it contains a self-loop. However, it is hard to analyze neighborhood structure due to the complex graph structure of various types and sizes. Thus, we think about neighborhood distribution from the perspective of edges. The edge distribution can be expressed as $\mathcal{E}$. To facilitate analysis, we make assumptions as follows: (1) The node features in class c obey the distribution $\mathcal{F}_c$, and a single node contains incomplete class feature, which is equivalent to sampling from class feature distribution i.e. $x_i \sim \mathcal{F}_c$. (2) There are several edge distributions in class c and edges around node i satisfy $e_i \sim \mathcal{E}_c$ where $\mathcal{E}_c = \sum_{k=1}^{K} \theta_k \mathcal{E}_{ck}$, θ_k is weighting coefficient, $k \in [1, K]$ and K is the number of distributions. Edges around node i in class c are sampled from the edge distribution $\mathcal{E}_c$.

Based on these assumptions, we denote the neighborhood distribution of class c as $\mathcal{D}_c = \{\{\mathcal{F}_c, c \in \mathcal{C}\}, \{\mathcal{E}_c, c \in \mathcal{C}\}\}$. Since neighborhood distribution from social networks or protein molecular is typically difficult to be expressed formally, we introduce an example of a simple graph to analyze neighborhood distribution. Given a graph G with n classes where disjoint set $\mathcal{C}_k$ corresponding to the class c_k and $k \in [1, n]$. For the convenience of research, we define a unified edge distribution in the same class, which corresponds to the assumption (2) that $k = 1$ and $\theta_k = 1$. To connect edges between nodes, the connection probability p_{kj} between class c_k and class c_j are defined with $k \in [1, n]$ and $j \in [1, n]$. In this case, node features are sampled from Normal distribution $N\left(\mu_k, I_k\right)$ where $\mu_k \in \mathbb{R}^{n \times f}$ and f is length of node feature.

According to process of graph convolution in GCN, the node feature containing neighborhood distribution information of node i can be expressed as $h_i = \frac{1}{deg(i)} \sum_{j \in \mathcal{N}(i)} x_j$ where $deg\left(i\right)$ is degree of node i and $\mathcal{N}\left(i\right)$ denotes neighbors of node i . Moreover, based on the connection probability between classes, the edge distributions of nodes in class c_k can be expressed as $\mathcal{E}_{c_k} = \{\frac{p_{kj}}{\sum_{j=1}^{n} p_{kj}}\}$, where $j \in [1, n]$ Therefore, node feature after GCN operation still follow Normal distributions, described by:

$$h_i \sim N\left(\frac{\sum_{j=1}^{n} p_{kj}\mu_j}{\sum_{j=1}^{n} p_{kj}}, \frac{I}{\sqrt{\left(deg\left(i\right)\right)}}\right), for\ i \in \mathcal{C}_k\ and\ j \in [1, n]. \tag{3}$$

We denote $\mathbb{E}_{e_i \sim \mathcal{E}_{c_k}, x_i \sim \mathcal{F}_{c_k}}\left[x_i\right]$ as the mathematical expectation of node i in class c_k after graph convolution. Based on the properties of Normal distributions, the expectation of node features in class c_k can be expressed as $\mathbb{E}_{e_i \sim \mathcal{E}_{c_k}, x_i \sim \mathcal{F}_{c_k}}\left[x_i\right] = \frac{\sum_{j=1}^{n} p_{kj}\mu_j}{\sum_{j=1}^{n} p_{kj}}$ where $j \in [1, n]$. Intuitively, we can find that the variance of neighborhood distribution is related to the degree of nodes in this particular distribution, where the variance decreases as the degree increases. In addition, the Monte Carlo method is an effective method for estimation of mathematical expectation. Hence, when edges are added for each node sampling from edge distribution according to the principle of independent and identical distribution, nodes in class c_k tend to converge to the expectation $\mathbb{E}_{e_i \sim \mathcal{E}_{c_k}, x_i \sim \mathcal{F}_{c_k}}\left[x_i\right]$ following Law of large numbers as:



(a) Graph topology around node i

(b) Original NFD in squirrel

(c) Aggregated NFD using original graph topology

(d) Aggregated NFD using synthetic graph topology.

Figure 1: Examples of neighborhood distribution. (a) shows an example of the neighborhood structure centered on node i. (b)(c)(d) are generated from t-SNE (Van der Maaten & Hinton, 2008) on dataset Squirrel. Specifically, (b) is generated from original NFD using MLP, (c) shows aggregated NFD using original graph topology (h=0.22) and (d) displays aggregated NFD using synthetic graph topology by adding inter-class edges according to a certain distribution (h=0.1) using GCN.

$$\frac{1}{deg\left(i\right)}\sum_{j\in\mathcal{N}(i)}x_j\rightarrow\mathbb{E}_{e_i\sim\mathcal{E}_{c_k},x_i\sim\mathcal{F}_{c_k}}\left[x_i\right],deg\left(i\right)\rightarrow\infty\ and\ k\in\{1,2\}. \tag{4}$$

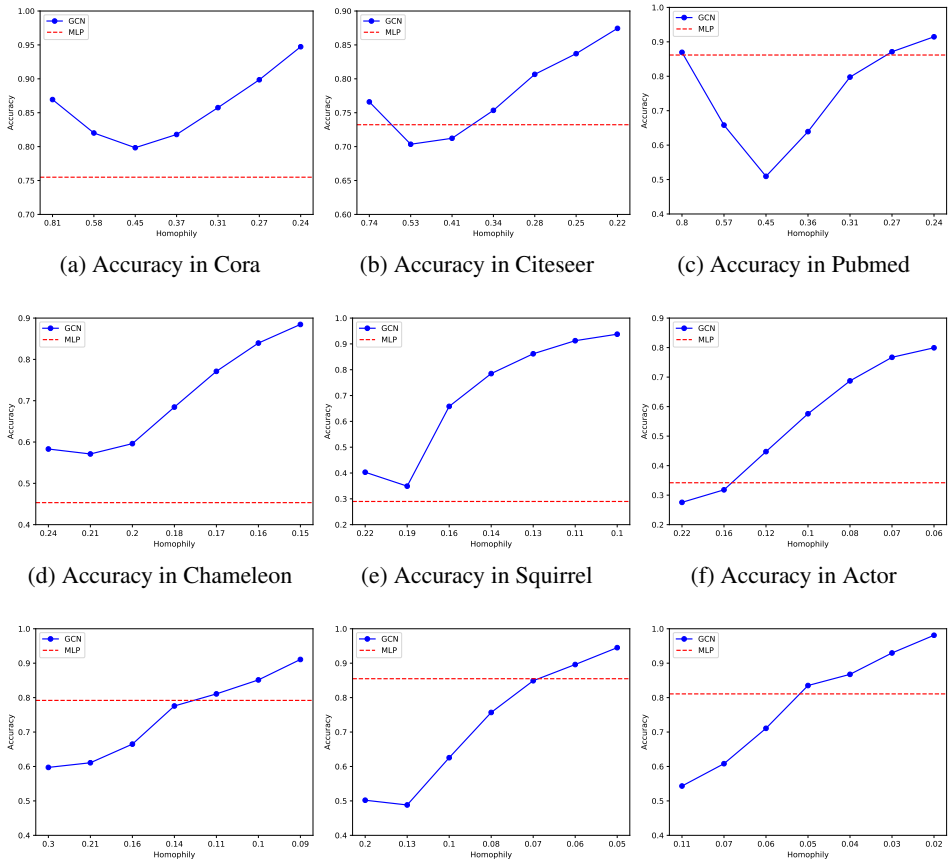
In this case, when we generate enough edges according to the distribution, if there is a large distance of mathematical expectation $\mathbb{E}_{e_i\sim\mathcal{E}_{c_k},x_i\sim\mathcal{F}_{c_k}}\left[x_i\right]$ and $\mathbb{E}_{e_m\sim\mathcal{E}_{c_j},x_m\sim\mathcal{F}_{c_j}}\left[x_m\right]$ between any two classes c_k and c_j, the neighborhood distribution in each class is distinguishable so that node can be classified correctly, even if the homophily of graph is low. Further, we denote a graph distribution following assumptions (1)-(2) as $\mathcal{G}=\{\mathcal{V},\{\mathcal{F}_c,c\in\mathcal{C}\},\{\mathcal{E}_c,c\in\mathcal{C}\}\}$ with node set $\mathcal{V}$ and analyze the aggregated feature distribution using direct neighbors.

Theorem 3.1. *Consider a graph $\mathcal{G}=\{\mathcal{V},\{\mathcal{F}_c,c\in\mathcal{C}\},\{\mathcal{E}_c,c\in\mathcal{C}\}\}$ which follows Assumptions (1)(2). For node $i\in\mathcal{V}_c$, aggregated feature obtained by direct neighbors is given by $h_i=\frac{1}{deg(i)}\sum_{j\in\mathcal{N}(i)}x_j$, where the dimension of node feature is f and all elements of feature are bounded in $[a,b]$. And expectation of nodes in $\mathcal{C}_c$ is described by $\mathbb{E}[h_i]=\mathbb{E}_{e_i\sim\mathcal{E}_c,x_i\sim\mathcal{F}_c}\left[x_i\right]$. For $t>0$, the probability that the distance between the aggregated feature h_i and its expectation is larger than t is bounded by*

$$\mathbb{P}\left(\left\|h_i-\mathbb{E}[h_i]\right\|_p\geq t\right)\leq 2\cdot f\cdot exp\left(-\frac{2deg(i)t^2}{\left(b-a\right)^2f^{\frac{2}{p}}}\right). \tag{5}$$

The proof can be found in Appendix A. Theorem 3.1 indicates that the distance between aggregated node feature and expectation is small with a high probability if the degree of nodes is high. As the degree of nodes increases, node features are more likely to approach expectations. To ensure the distinguishability of aggregated feature distribution for node classification, the distribution between classes should be far away from each other.

Empirical investigation for neighborhood distribution To examine our assumptions and inferences, we conduct a series of experiments using two-layer GCN on nine graphs (see Appendix B for more details). Specially, for fast and effective convergence, we add inter-class edges sampling from specific edge distribution i.e. generate one-way edges from class c_k to c_{k+1}. From Figure 2, we observe that with the increase of edge between classes, the homophily of each graph is declining, but the accuracy is not always decreasing. It indicates that homophily may not the most important factor affecting the accuracy of GCN. In details, the accuracy of GCN experiences a process of first decline and then rise in most datasets including all homophilous datasets (Cora, Citeseer and Pubmed) and partial heterophilous datasets (Chameleon, Squirrel and Wisconsin). The integration of the original and newly added distribution will make the distinguishability of neighborhood distribution confused at first. As the increasing neighborhood structure distribution dominates, the distinguishability in different classes will continuously enhance, and the performance continues to improve. In other datasets like Actor, Cornell and Texas, when a small number of edges are added according to the edge distribution, the accuracy can be improved. It is likely that the aggregated feature using original



(a) Accuracy in Cora (b) Accuracy in Citeseer (c) Accuracy in Pubmed

(d) Accuracy in Chameleon (e) Accuracy in Squirrel (f) Accuracy in Actor

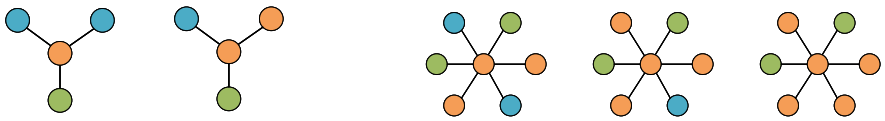
(g) Accuracy in Cornell (h) Accuracy in Wisconsin (i) Accuracy in Texas

Figure 2: The performance of GCN while adding inter-class edges. The x-axis is homophily and the y-axis is accuracy. Note that the red dotted line indicates the MLP prediction accuracy.

graph topology become difficult to distinguish. When adding edges according to edge distribution, the chaos is broken so that neighborhood distribution become gradually distinguishable. The results demonstrate that the performance of GCN is not determined by homophily, and graph structure affects the distinguishability of neighborhood distribution and thus the performance of aggregation. In addition, we conduct experiments about the impact of neighborhood range using SGC Wu et al. (2019) in Appendix B.3. The results show that neighborhood expansion is more likely to play a role in a "good" graph structure. And local neighborhood information (within 2-hop neighborhood) can be reliable for classification, especially in heterophilous graphs.

To intuitively study how neighborhood structure affects distinguishability, we perform a set of t-SNE visualization on commonly used heterophilous graph datasets Squirrels, including using node features without graph topology, using original graph structure for aggregation, and using synthetic graph topology for aggregation. In Figure 1, experimental results show that a "good" graph structure can improve the performance of GCN, even if the original features are difficult to distinguish in Figure 1b (using MLP). Besides, the effect of the original graph structure may not be obvious on some datasets like Squirrel (see Figure 1c). However, when we use the synthetic graph with low homophily (h=0.1) to aggregate neighbors, the distinguishability of aggregated node feature is greatly improved (see Figure 1d). It also demonstrates that neighborhood structure affects the distinguishability of neighborhood distribution and then the performance of GCN. Therefore, how to update the graph topology to get a "good" neighborhood structure has become a key problem.

Furthermore, we note that some recent works start focusing on local neighborhood information (Jin et al., 2021; Liu et al., 2022). LAGNN proposed that local neighborhood could enhance the



(a) An example for similar nodes.

(b) An example for similar neighborhoods.

Figure 3: Examples for finding neighbors. (a) shows an example that two nodes with different neighborhoods but similar node feature. (b) shows that three nodes have different node features, but two adjacent nodes in example have similar neighborhood information.

expressive power of existing GNNs. U-GCN proposed that 1-hop,2-hop and kNN neighbors could be used in different types of networks respectively, including complete homophily, complete random and complete heterophily. The findings also inspire us that not only the node itself but also local neighborhood information can be used to find neighbors for better neighborhood structure.

For local neighborhood distribution, there are two main situations to be considered. The first is that the similarity of nodes of the same class is much higher than that of nodes between classes. In this case, the node itself can be used for similarity measurement even if two nodes have different neighbors (Figure 3a). If the node feature fails, 0-hop, 1-hop and 2-hop neighbors can be combined to find neighbors for constructing a new graph topology. For example, the three central nodes in Figure 3b belong to the same class but differ from each other, so it is hard to directly use the node feature to find new neighbors. However, their local neighborhood information may not change dramatically, so it is possible to use local neighborhood information to determine neighbors for aggregation. After finding more "good" neighbors, more reliable neighbors are obtained to facilitate the exploration of neighborhood range and supplement abundant feature information.

4 GCN-PND Model

Neighborhood structure influences the distinguishability of neighborhood distribution and then the performance of GCNs. In neighborhood structure, there are mainly two aspects that need to be considered, i.e., neighborhood structure around the central node and neighborhood range. So we propose to design a graph convolutional operator from the two aspects.

In spectral graph theory, the normalized graph Laplacian matrix is defined as $L = I - \hat{A}$ with eigendecomposition $L = U\Lambda U^T$, where Λ is a diagonal matrix of the eigenvalues of L and $U \in \mathbb{R}^{n\times n}$ is the matrix that consists of the eigenvectors of L. The graph convolution operation, a filter $g_\theta = diag(\theta)$ with parameters θ acts on x is defined as $g_\theta(L) * x = U g_\theta(\Lambda) U^{\mathrm{T}} x$. Further, graph convolution operation can be approximated by the K-th order polynomial of Laplacians (Defferrard et al., 2016; Chen et al., 2020) as

$$U g_\theta(\Lambda) U^{\mathrm{T}} x \approx U \left(\sum_{k=0}^{K} \theta_k \Lambda^k \right) U^{\mathrm{T}} x = \left(\sum_{k=0}^{K} \theta_k L^k \right) x, \tag{6}$$

where $\theta \in \mathbb{R}^{K+1}$ corresponds to a vector of polynomial coefficients. Similarly, graph diffusion (Klicpera et al., 2019) attempts to calculate weighting coefficients for multi-hop neighbors in the form of polynomials. However, the difference between each node may be ignored if the same layer uses a common coefficient. So we consider a node-level aggregation strategy. Further, the graph diffusion or propagation should be not limited to common adjacency matrix forms. Therefore, the generalized convolution operation on a node i can be described as:

$$g_\theta * x = \left(\sum_{k=0}^{K} \theta_{(k,i)} T^k \right) x, \tag{7}$$

where $\theta_{(k,i)}$ denotes weighting coefficients from k-hop neighbors of node i and T is a general transition matrix. Especially, T can be variant of initial adjacency matrix (such as random walk transition matrix $T = AD^{-1}$ and symmetric transition matrix $T = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$) or updated adjacency matrix

by graph topology learning. Finally, we conclude the generalized convolution operation in GCNs model including updating graph topology and designing extensible neighborhood aggregation.

Updating graph topology: Before updating graph topology, the node's embedding need to be generated to measure the distance between different nodes. In our model, we use a simple and effective way to aggregate local neighborhood as follows:

$$H^{(l+1)} = \left((1-\alpha)\hat{A} + \alpha I\right) H^{(l)}, \tag{8}$$

where $H^{(0)} = f_\theta(X)$, $f_\theta(X)$ denotes the fully connected neural network with activation function ReLU on the feature matrix X, and α is the teleport probability. The generated node embedding contains local neighborhood information (within 2-hop neighbors) for further topology updating.

Then, we calculate the similarity matrix $S \in \mathbb{R}^{n\times n}$ utilizing Cosine similarity, which adopts the cosine value of the angle between two node embeddings to measure the similarity. Specifically, let $h^{(L)}$ be node embedding after aggregation among L-hop ($L \leq 2$) neighborhood, the similarity S_{ij} between node v_i and node v_j is: $S_{ij} = cos(h_i^{(L)}, h_j^{(L)})$. Then, a new adjacency matrix S_{kNN} is obtained by choosing top k similar node pairs. In addition, we set threshold ρ on $\hat{A}$ for edge deletion to obtain A_D. The new graph topology is obtained by combining two adjacency matrices as:

$$A_C = A_D + \lambda S_{kNN} \tag{9}$$

where λ is a hyperparameter for edges of kNN graph. Note that we reconstruct the original graph topology by setting the threshold to 1 for all heterophilous graphs. For homophilous graphs, graph topology just needs to be tuned by setting a low threshold. Further, let e_{ij} be weighting coefficient of edge between node i and node j in A_C where $j \in \mathcal{N}_i$ and $\mathcal{N}_i$ is direct neighbors of node i, we normalize edges for stability of aggregation as:

$$e_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}_i} exp(e_{ij})}. \tag{10}$$

Designing extensible neighborhood aggregation: Based on updated graph topology, the second design involves how to aggregate from multi-hop neighbors of each node. Let $Z^{(0)} = H^{(0)} \in \mathbb{R}^{n\times f'}$ denote the new feature matrix in latent space after feature transformation. Feature matrix $Z^{(k)}$ from k-hop neighbors after feature propagation using combined adjacency matrix A_C is described by:

$$Z^{(k)} = A_C Z^{(k-1)}. \tag{11}$$

Then we stack feature matrices $\left\{Z^{(0)}, \cdots, Z^{(K)}\right\}$ to generate feature tensor T where $T_{[:,j,:]} = Z^{(j)}$ and $T \in \mathbb{R}^{n\times(K+1)\times f'}$. Feature tensor T contains information from multi-hop neighbors of nodes, the weight (importance) distribution of node neighborhood in T is calculated as:

$$w_{i,k} = \frac{exp\left(\vec{u}_i \vec{t}_{i,k}\right)}{\sum_{k=0}^{K} exp\left(\vec{u}_i \vec{t}_{i,k}\right)}, \tag{12}$$

where $U \in \mathbb{R}^{n\times f'}$ is the parameter matrix of the network and $\vec{u}_i$ is the learnable parameter vector for node i. $\vec{t}_{i,k} = T_{[i,k,:]}$ is a vector in T, which reflects the feature from k-hop neighborhood of node i. $w_{i,k}$ denotes the weighting coefficient for the k-hop neighborhood of node i. Further, we aggregate neighborhood information for each node according to weighting coefficients as:

$$\vec{z}_i = \sigma(\sum_{k=0}^{K} w_{i,k}\vec{t}_{i,k}). \tag{13}$$

The aggregated node features are input to MLP, the final layer of the network, to obtain the output of nodes for prediction. The algorithm can be found in the Appendix C.

5 Other Related Work

GCNs (Kipf & Welling, 2017; Vaswani et al., 2017; Hamilton et al., 2017) design graph convolution operators, which have aroused widespread concern. Many subsequent methods (Gilmer et al.,

2017; Abu-El-Haija et al., 2019) attempt to improve the process of neighborhood aggregation, but in essence, they are to enhance the distinguishability of neighborhood distribution in various classes.

Improve neighborhood structure distribution. There are two common ways to improve neighborhood structure distribution for aggregation: updating graph topology and increasing propagation steps. LDS (Franceschi et al., 2019) adds numerous edges while learning graph structure by bilevel framework (Franceschi et al., 2018). Geom-GCN (Pei et al., 2020) and U-GCN (Jin et al., 2021) which find neighbors from high-order neighborhood achieve great performance in some heterophilous graphs. MixHop (Abu-El-Haija et al., 2019), JKNet (Xu et al., 2018) and H2GCN (Zhu et al., 2020) utilize high-order information by concatenation. Moreover, graph diffusion and propagation become popular due to simplicity and effectiveness. Especially, PageRank (Page et al., 1999; Andersen et al., 2006; Klicpera et al., 2019) and heat kernel (Kondor & Lafferty, 2002; Kloster & Gleich, 2014; Chamberlain et al., 2021; Zhao et al., 2021) are suitable choices for adjusting the proportion of different hop neighbors during aggregation.

Design loss for distinguishability. It is likely to be consistent between the results generated by the same node under different data augmentations, different models and different views. GRAND (Feng et al., 2020) uses consistency loss to align predictions after multiple data enhancements. Graph contrastive learning (Hassani & Khasahmadi, 2020; Zhu et al., 2021; Xia et al., 2022) typically reduce the distance between positive samples and increase the distance between negative samples or maximize mutual information between different views. In addition, the relationship between neighbors can also constrain the neighborhood distribution. IDGL (Chen et al., 2020) sets smoothness loss to keep node and mean value of neighbors consistence. Similarly, Propagation-regularization (Yang et al., 2021) is also proposed to constrain the representation of nodes and their neighbors.

6 EXPERIMENTS

Datasets and settings. For semi-supervised node classification, we use three standard citation network datasets including Cora, Citeseer and Pubmed (Sen et al., 2008), which have strong homophily. We apply the standard fixed split (Yang et al., 2016), which includes 20 nodes per class for training, 500 nodes for validation and 1000 nodes for testing. For all benchmarks of full-supervised node classification, we adopt 10 random splits (48%/32%/20%) of nodes per class for training/validation/testing, which are provided by Geom-GCN (Pei et al., 2020). Statistics of the datasets are summarized in Appendix D.1.

Baselines. We compare GCN-PND with seven semi-supervised baselines including GCN (Kipf & Welling, 2017), GAT (Vaswani et al., 2017), MixHop (Abu-El-Haija et al., 2019), APPNP (Klicpera et al., 2019), SGC (Wu et al., 2019), SSGC (Zhu & Koniusz, 2021) and GPRGNN (Chien et al., 2021). For full-supervised node classification, we compare our model with 14 baselines: GCN, GAT, GraphSAGE (Hamilton et al., 2017), JKNet (Xu et al., 2018) with concatenation, APPNP, GCNII and its variant, Geom-GCN (Pei et al., 2020) with three variants, H2GCN (Zhu et al., 2020) with two variants, GPRGNN (Chien et al., 2021) and MLP. See Appendix D.2 for more details.

Semi-supervised node classification. We report the classification accuracy with both mean and standard deviation after 50 runs in Table 1. Note that we reprint the accuracy of the original paper due to results reproduced by MixHop being far lower than the accuracy of the paper. As shown, the accuracy of our GCN-PND significantly outperforms when compared to other models. In addition, we observe that models which extend the range of neighborhood can achieve great accuracy on all datasets, such as APPNP and GPRGNN. It indicates that properly expanding the neighborhood in the homophilous graph is conducive to improving the distinguishability of neighborhood distribution.

Full-supervised node classification. We evaluate the performance of GCN-PND in the task of full-supervised classification. In addition, we set up another form (GCN-PND*) using adjacency matrix without self-loops. From table 2, GCN-PND achieves the state-of-the-art performance on 7 out of 9 datasets compared with the other 14 baselines, except Geom-GCN-I on dataset Citeseer and GCNII* on dataset Pubmed. We think GCNII* and Geom-GCN-I are suitable for specific homophilous datasets rather than heterophilous graphs. Significantly, GCN-PND outperforms other baselines on heterophilous datasets, which indicates the strong ability in processing heterophilous graphs. On heterophilous datasets Squirrel and Chameleon, the accuracy of our model is nearly 19% and 11% higher than the best baseline GPRGNN respectively, and the graph without self-loops is much

Table 1: Summary of semi-supervised classification accuracy(%) on Cora, Citeseer and Pubmed. We mark with daggers (†) the reprinted results from the respective papers. Boldface letters are used to mark the best results.

Method	Cora	Citeseer	Pubmed
GCN	81.4±0.7	70.8±0.7	79.1±0.3
GAT	83.1±0.6	72.4±0.7	77.6±0.8
MixHop†	81.9±0.4	71.4±0.8	80.8±0.6
SGC	80.8±0.0	71.4±0.0	78.9±0.0
SSGC	82.6±0.1	73.0±0.0	80.0±0.1
APPNP	83.2±0.6	71.0±0.8	80.0±0.5
GPRGNN	83.6±0.4	71.6±0.4	79.6±0.2
GCN-PND	**85.0±0.7**	**73.6±0.4**	**81.0±0.5**
w/o GTU	84.9±0.6	73.1±0.6	79.6±0.4
w/o MHA	84.2±0.5	72.2±0.5	80.1±0.6

Table 2: Mean classification accuracy(%) of full-supervised node classification using 10 random splits. Note that boldface letters are used to mark the best results.

Method	Chameleon	Squirrel	Actor	Cornell	Wisconsin	Texas	Cora	Citeseer	Pubmed
GCN	59.98	38.75	27.16	61.35	49.41	55.95	87.12	76.40	87.03
GAT	55.75	35.24	27.18	60.54	50.00	56.22	87.67	76.09	85.48
GraphSAGE	62.24	44.25	34.1	62.24	77.25	71.89	86.88	76.72	88.71
JKNet	60.39	45.38	25.91	58.92	48.63	58.38	84.79	71.79	85.92
APPNP	54.39	35.11	26.53	58.65	45.69	58.92	87.36	75.29	86.64
GPRGNN	66.48	48.86	35.27	84.71	83.53	83.53	87.78	76.62	87.85
GCNII	58.99	38.87	33.76	72.43	72.75	71.62	88.21	76.96	89.38
GCNII*	63.20	41.46	34.88	77.03	81.18	76.22	88.01	76.94	**90.27**
Geom-GCN-I	60.37	33.14	29.10	56.75	58.63	57.57	85.25	**78.05**	89.99
Geom-GCN-P	60.92	38.09	31.65	59.45	64.51	68.38	84.83	75.40	88.08
Geom-GCN-S	60.19	36.14	30.32	55.67	56.86	60.26	85.27	74.90	84.70
H2GCN-1	52.96	36.42	34.31	79.46	83.14	83.24	86.21	77.11	89.43
H2GCN-2	58.38	32.33	34.49	78.11	84.31	80.00	87.93	77.06	89.55
MLP	45.33	28.98	34.20	79.19	85.49	81.08	75.49	73.23	86.17
GCN-PND	71.34	64.45	**35.31**	**86.76**	**85.69**	84.32	**88.39**	77.95	89.52
GCN-PND*	**77.63**	**68.02**	35.01	84.32	85.29	**85.68**	87.91	77.30	89.46
w/o GTU	56.18	40.27	30.02	61.89	63.33	64.86	87.95	76.80	89.21
w/o MHA	71.24	65.17	31.45	83.24	86.49	83.24	87.20	77.75	88.73

better due to the neighborhood distribution without central node being more distinguishable. In datasets Cornell, Wisconsin and Texas, node features are distinguishable between classes. Hence, we use local neighborhood distribution with $\alpha = 1$ for graph topology updating. In Actor, low distinguishability of neighborhood distribution results in low performance for all models.

Ablation study. To examine the contributions of different components in GCN-PND, we conduct an ablation study as follows: We use the original graph for aggregation from multi-hop neighbors adaptively without graph topology updating (w/o GTU). We use updated graph topology in GCN model without multi-hop aggregation(w/o MHA). In the task of semi-supervised and full-supervised node classification, we observe that GCN with updated graph topology, i.e. w/o MHA, improves performance of GCN on all datasets, especially heterophilous graphs. In addition, the average performance of GCN-PND will decline with any component removed, which demonstrates the significance of the designed component. Further, the impact of neighborhood range can be found in Appendix D.3.

7 CONCLUSION

We study and analyze the impact of neighborhood distribution in graph convolutional networks, and find that an important factor affecting the performance of graph convolution neural networks is the distinguishability of neighborhood distribution. According to the factors affecting the neighborhood distribution, we propose the graph convolution operator including updating graph topology and designing extensible neighborhood aggregation. Further, we design GCN-PND based on our convolution operator. Experimental results demonstrate the effectiveness of our model.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Hrayr Harutyunyan, Nazanin Alipourfard, Kristina Lerman, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International Conference on Machine Learning*, pp. 21–29, 2019.

Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 475–486. IEEE, 2006.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Ben Chamberlain, James Rowbottom, Maria Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pp. 1407–1418, 2021.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems*, pp. 19314–19326, 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural Information Processing Systems*, pp. 3844–3852, 2016.

Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *Advances in Neural Information Processing Systems*, pp. 22092–22103, 2020.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1568–1577. PMLR, 2018.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International Conference on Machine Learning*, pp. 1972–1982, 2019.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272, 2017.

William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Neural Information Processing Systems*, pp. 1025–1035, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126. PMLR, 2020.

Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, 34, 2021.

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

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*, 2019.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, 2019.

Kyle Kloster and David F Gleich. Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1386–1395, 2014.

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, volume 2002, pp. 315–322, 2002.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

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. *Advances in Neural Information Processing Systems*, 34, 2021.

Songtao Liu, Rex Ying, Hanze Dong, Lanqing Li, Tingyang Xu, Yu Rong, Peilin Zhao, Junzhou Huang, and Dinghao Wu. Local augmentation for graph neural networks. In *International Conference on Machine Learning*, pp. 14054–14072. PMLR, 2022.

Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 723–731, 2019.

Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? *arXiv preprint arXiv:2106.06134*, 2021.

Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 4602–4609, 2019.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Shashank Pandit, Duen Horng Chau, Samuel Wang, and Christos Faloutsos. Netprobe: a fast and scalable system for fraud detection in online auction networks. In *Proceedings of the 16th international conference on World Wide Web*, pp. 201–210, 2007.

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*, 2020.

Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *Ai Magazine*, 29(3):93–106, 2008.

Ke Sun, Zhouchen Lin, and Zhanxing Zhu. Adagcn: Adaboosting graph convolutional networks into deep models. In *International Conference on Learning Representations*, 2021.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, pp. 5998–6008, 2017.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pp. 6861–6871, 2019.

Jun Xia, Lirong Wu, Ge Wang, Jintao Chen, and Stan Z Li. Progcl: Rethinking hard negative mining in graph contrastive learning. In *International Conference on Machine Learning*, pp. 24332–24346. PMLR, 2022.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5449–5458, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks. In *International Conference on Learning Representations*, 2019.

Han Yang, Kaili Ma, and James Cheng. Rethinking graph regularization for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 4573–4581, 2021.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *International Conference on Machine Learning*, pp. 7134–7143. PMLR, 2019.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

Jialin Zhao, Yuxiao Dong, Ming Ding, Evgeny Kharlamov, and Jie Tang. Adaptive diffusion in graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Hao Zhu and Piotr Koniusz. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2021.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11168–11176, 2021.

A PROOF OF THEOREM 3.1

Before proving Theorem 3.1, we introduce Hoeffding's inequality as follow: Let $X_1, \ldots, X_n$ be independent bounded random variables with $X_i \in [a, b]$, where $i \in [1, n]$ and $-\infty < a \leq b < \infty$. For all $t \geq 0$,

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i])\right| \geq t\right) \leq 2 \cdot exp\left(-\frac{2nt^2}{(b-a)^2}\right). \tag{14}$$

Proof. Let $x_i[k], k = 1, \ldots, f$ denote the k-th element of node x_i. For dimension k, $\{x_j[k], j \in \mathcal{N}(i)\}$ is a set of independent bounded random variables. Utilizing Hoeffding's Inequality, for any $t_1 \geq 0$, we have the bound as following:

$$\begin{aligned}\mathbb{P}\left(|h_i[k] - \mathbb{E}[h_i[k]]| \geq t_1\right) &= \mathbb{P}\left(\left|\frac{1}{deg\,(i)}\sum_{j\in\mathcal{N}(i)}(X_j[k] - \mathbb{E}[X_j[k]])\right| \geq t_1\right) \\ &\leq 2 \cdot exp\left(-\frac{2deg(i)t_1^2}{(b-a)^2}\right).\end{aligned} \tag{15}$$

Hence, for all elements in $k = 1, \ldots, f$, we have

$$\begin{aligned}\mathbb{P}\left(\|h_i - \mathbb{E}[h_i]\|_p \geq f^{\frac{1}{p}}t_1\right) &\leq \mathbb{P}\left(\bigcup_{k=1}^{f}\left\{\frac{1}{deg\,(i)}\sum_{j\in\mathcal{N}(i)}(x_j[k] - \mathbb{E}[x_j[k]]) \geq t_1\right\}\right) \\ &\leq \sum_{k=1}^{f}\mathbb{P}\left(\frac{1}{deg\,(i)}\sum_{j\in\mathcal{N}(i)}(x_j[k] - \mathbb{E}[x_j[k]]) \geq t_1\right) \\ &\leq 2 \cdot f \cdot exp\left(-\frac{2deg(i)t_1^2}{(b-a)^2}\right).\end{aligned} \tag{16}$$

Let $t = f^{\frac{1}{p}}t_1$, i.e. $t_1 = \frac{t}{f^{\frac{1}{p}}}$, then we have

$$\mathbb{P}\left(\|h_i - \mathbb{E}[h_i]\|_p \geq t\right) \leq 2 \cdot f \cdot exp\left(-\frac{2deg(i)t^2}{(b-a)^2 f^{\frac{2}{p}}}\right), \tag{17}$$

which completes the proof. □

B ADDITIONAL DETAILS IN SECTION 3.3.1

We present details of the GCN model and the algorithm for changing edges in graphs. GCN uses two layers with nonlinear activation function ReLU after the first layer and the hidden units are set to 64. We adopt the dropout rate as 0.5 for node features. The early stopping criterion uses a patience of $p = 100$ and a maximum of $n = 1500$. Further, we use the Adam optimizer with a learning rate of $l = 0.01$ and cross-entropy loss. The L_2 regularization parameter is set as 5×10^{-4} on all datasets.

B.1 ADD EDGES ACCORDING TO DISTRIBUTION.

The algorithm of adding edges according to distribution is presented in Algorithm 1. To prevent the generated edges from concentrating on a few classes whereas classes with few nodes have no chance to add new edges, we recalculate the distribution according to the number of edges in each class. Specifically, the proportion of the number of edges between two classes in the total number of edges is taken as the initial edge distribution, and the edge distribution is updated by both initial edge distribution and specific edge distribution as shown in Algorithm 1. And then, the edge is generated by new obtained edge distribution. In the datasets Texas and Cornell, the algorithm is

slightly different. Due to the condition that there is a class that contains only one node has no edges (isolated node), our default class distribution sampling number is at least 1.

Algorithm 1: Adding edges sampling from distribution

Input: A, $\mathcal{D}$, r
Output: new adjacency matrix A'

Calculate number of edges m.
Calculate proportion of edges between classes $\mathcal{P}$.
Calculate specific distribution of class c: $\mathcal{D}_c = \mathcal{D}_c * \mathcal{P}_c$.
Calculate adding numbers $n = m * r$.
Initialize k=1.
while $k \leq n$ **do**
    Sample node $i \sim Uniform(\mathcal{V})$.
    Obtain the label c of node i.
    Sample a set $\mathcal{V}_c$ from $\mathcal{D}_c$.
    Sample a node $j \sim Uniform(\mathcal{V}_c)$.
    Update edge set $\mathcal{E}$.
    $k \leftarrow k + 1$.
end
Update A using edge set $\mathcal{E}$.
return updated adjacency matrix A'

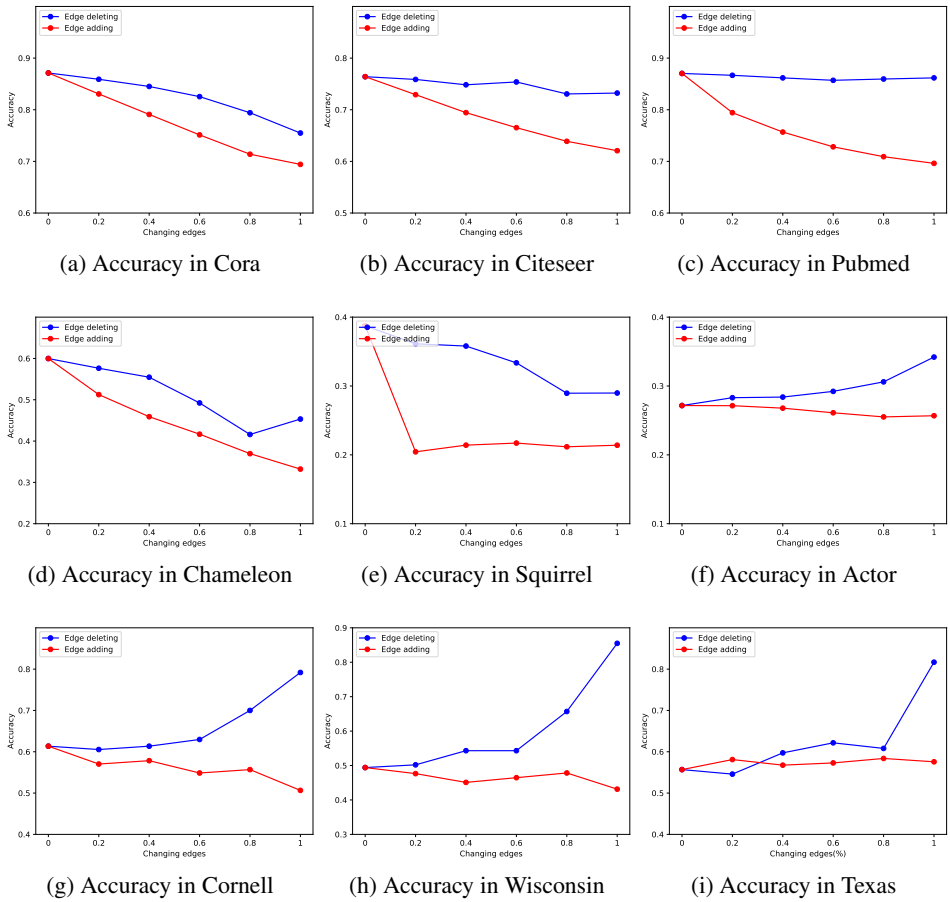
The details of adding inter-class edges according to the distribution are summarized in Table 3. Each dataset contains two rows of data, the first row is the ratio of added edges (r) compared with the number of original edges, and the second row is the homophily (h) after adding edges. For instance, at the beginning, the homophily of the Cora graph is 0.81 ($h = 0.81$ and $r = 0$), and the homophily is about 0.27 after increasing inter-class edges by 2 times ($h = 0.27$ and $r = 2$).

Table 3: Details of adding inter-class edges in different datasets, where the first row is the ratio of adding edges, and the second row is the homophily after adding edges.

Cora	0	0.4	0.8	1.2	1.6	2	2.4
	0.81	0.58	0.45	0.37	0.31	0.27	0.24
Citeseer	0	0.4	0.8	1.2	1.6	2	2.4
	0.74	0.53	0.41	0.34	0.28	0.25	0.22
Pubmed	0	0.4	0.8	1.2	1.6	2	2.4
	0.8	0.57	0.45	0.36	0.31	0.27	0.24
Chameleon	0	0.1	0.2	0.3	0.4	0.5	0.6
	0.24	0.21	0.2	0.18	0.17	0.16	0.15
Squirrel	0	0.2	0.4	0.6	0.8	1	1.2
	0.22	0.19	0.16	0.14	0.13	0.11	0.1
Actor	0	0.4	0.8	1.2	1.6	2	2.4
	0.22	0.16	0.12	0.1	0.08	0.07	0.06
Cornell	0	0.5	1	1.5	2	2.5	3
	0.3	0.21	0.16	0.14	0.11	0.1	0.09
Wisconsin	0	0.5	1	1.5	2	2.5	3
	0.2	0.13	0.1	0.08	0.07	0.06	0.05
Texas	0	0.5	1	1.5	2	3	5
	0.11	0.07	0.06	0.05	0.04	0.03	0.02

B.2 Random Addition and Deletion of Edges

To further study the influence of neighborhood structure distribution on feature aggregation, we randomly add or delete edges on 9 datasets. Then, a two-layer GCN is used to aggregate neighborhood features. The experimental results under different datasets are shown in Figure 4. For homophilous datasets, adding edges randomly makes it difficult to distinguish the features between classes, so the performance drops rapidly, even lower than the node classification without graph topology (only MLP is used). In the process of randomly deleting edges, we find that only the Cora dataset is



(a) Accuracy in Cora (b) Accuracy in Citeseer (c) Accuracy in Pubmed

(d) Accuracy in Chameleon (e) Accuracy in Squirrel (f) Accuracy in Actor

(g) Accuracy in Cornell (h) Accuracy in Wisconsin (i) Accuracy in Texas

Figure 4: The performance of GCN while adding or deleting edges randomly. The x-axis is the ratio of changing edges compared original graph and the y-axis is accuracy. The blue line indicates the prediction accuracy of GCN after randomly deleting edges, and the red line indicates the prediction accuracy of GCN after randomly adding edges.

greatly affected, while the citeseer and PubMed datasets have little performance change. The results also show that when classifying nodes without graph structure, homophilous datasets can typically perform well, and the original graph structure plays the role of icing on the cake.

For heterophilous datasets, adding edges randomly will lead to a decline in performance. Specifically, the original graph structure of some datasets like Actor, Wisconsin and Texas is poor, which makes the neighborhood feature distribution difficult to distinguish, and the performance change is small after random addition of edges. In contrast, the performance of GCN on Squirrel and Chameleon decreases rapidly. When randomly deleting edges, the situation becomes complicated. The performance will degrade on some heterophilous datasets like Chameleon and Squirrel, whereas performance is on the rise on Actor, Cornell, Wisconsin and Texas. The experimental results demonstrate that there are ”good” and ”bad” graph structures in heterophilous graphs. ”Good” graph structures can improve the distinguishability of the aggregated feature distribution, while ”bad” graph structures make it difficult to distinguish the feature distribution after aggregation.

B.3 The Impact of Neighborhood Range in SGC

In order to study the impact of neighborhood range on aggregation, we conducted experiments on 9 datasets using SGC in Figure 5. The specific parameters of SGC used in the experiment are as follows. We set a single-layer MLP for mapping. The dropout rate and L_2 regularization parameter are set to 0. Further, the maximum epoch is 1000 and the patience of early stopping is set to 100.

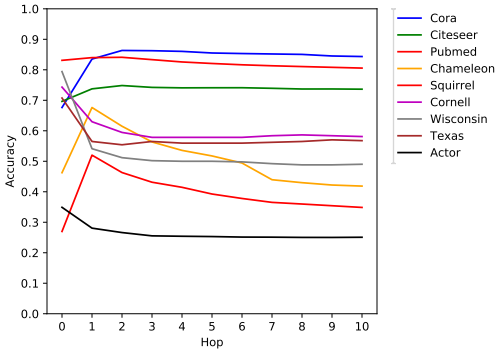


Figure 5: The impact of neighborhood range in SGC on 9 datasets.

The results show that with the expansion of the neighborhood range, the performance of SGC in homophilous graphs becomes stable after the 2-hop neighborhood. In heterophilous graphs, there are two situations that need to be further analyzed. In Chameleon and Squirrel, the accuracy of SGC rises first and then falls, which means local neighborhood is distinguishable in some heterophilous datasets. However, with the further expansion of the neighborhood, a large amount of useless information makes it difficult to distinguish the aggregated features so the accuracy drops rapidly. In other heterophilous graphs, performance degrades as the scope of aggregation expands, which indicates that graph topology plays a negative role in neighborhood aggregation. Therefore, we draw a conclusion that whether neighborhood expansion can improve performance depends on the graph structure. Further, we observe that in both homophilous and heterophilous graphs, good performance can be achieved by using local neighborhood information (within 2-hop neighborhood). It indicates that it is possible to find neighbors with local neighborhood information.

C ALGORITHM OF GCN-PND

To understand the content of the GCN-PND conveniently, we have sorted out the algorithm of the GCN-PND, as shown in Algorithm 2. Specifically, the algorithm mainly includes two parts, i.e. updating graph topology (Eq.(8), Eq.(9) and Eq.(10)) and designing extensible neighborhood aggregation (Eq.(11), Eq.(12) and Eq.(13)). Finally, the prediction is output by MLP.

Algorithm 2: The framework of GCN-PND

Input: Feature matrix X, adjacent matrix A, neighborhood range hops of K, local neighborhood range L, parameter λ, node pairs of top k, threshold ρ.
Output: Prediction P

while *not convergence* **do**
  Map X using the fully connected network $f_\theta(X)$
  for $l = 0$ *to* L **do**
    Aggregate local neighborhood from l-hop using Eq.(8)
  end
  Calculate Cosine similarity between nodes and choose top k node pairs to add edges
  Set threshold ρ on $\hat{A}$ to delete edges and get combined graph using Eq.(9)
  Normalize weighting coefficient of edge by softmax function using Eq.(10)
  for $k = 0$ *to* K **do**
    Feature propagation to extend neighborhood range using Eq.11.
  end
  Stack feature matrices $\left\{Z^{(0)}, \cdots, Z^{(K)}\right\}$ to generate feature tensor T.
  Compute weighting coefficient of each hop using Eq.(12).
  Aggregate neighborhood according to weighting coefficients using Eq.(13).
end
Output prediction P using MLP: $P = f_{mlp}(Z)$

D EXPERIMENTAL DETAILS

D.1 DETAILS OF DATASETS

The dataset statistics are summarized in Table 4 including nodes, edges, features, classes and homophily.

Table 4: Dataset statistics.

Dataset	Cora	Citeseer	Pubmed	Chameleon	Squirrel	Actor	Cornell	Wisconsin	Texas
#Nodes	2708	3327	19171	2277	5201	7600	183	183	251
#Edges	5429	4732	44338	36101	217073	33544	295	309	499
#Features	1433	3703	500	2325	2089	931	1703	1703	1703
#Classes	7	6	3	4	4	4	5	5	5
homophily(h)	0.81	0.74	0.8	0.24	0.22	0.22	0.31	0.21	0.11

D.2 DETAILS OF BASELINES

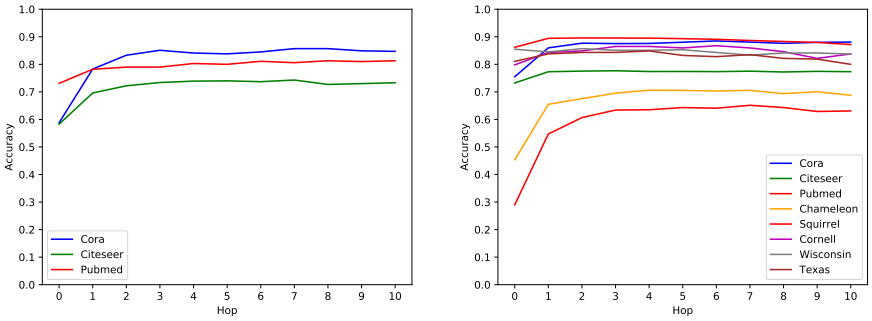
For semi-supervised node classification, GCN, GAT and SGC use the original settings as paper, where GAT uses a sparse version. The propagation step K is set to 10 for APPNP and GPRGNN, and 16 for SSGC. α is set to 0.1 in APPNP and GPRGNN, and 0.05 in SSGC. In our model, we set the weight decay parameter as 5×10^{-4} and learning rate of 0.01. The early stopping criterion uses patience of $p = 100$ and the maximum is set to 1500 for all datasets. α is set as 0.1 for aggregation and the hidden layer has 64 hidden units, with dropout $d = 0.5$ on both edges and feature tensor. For updating graph, we set $\lambda = 0.1$ for all datasets and use 1, 2, and 2 propagation steps for neighborhood aggregation. For aggregation from multi-hop neighbors, we adopt 8, 4, and 8 propagation steps in Cora, Citeseer and Pubmed separately. After fine-tuning, we set dropout rate as 0.5 and 0.8 on input layer and hidden layer in Cora and Pubmed, 0.2 and 0.2 in Citeseer.

For full-supervised node classification, all models adopt 64 hidden units in hidden layer, patience of $p = 100$. MLP, GCN, GAT, GraphSAGE and JKNet use weight decay parameter as 5×10^{-4}. Specially, GraphSAGE is reproduced using algorithm of original paper without node sampling. JKNet use concatenation for aggregation layers (5 layers). The hyperparameters are set following original papers in GPRGNN, GCNII, Geom-GCN and H2GCN. In our model, we use two adjacency matrices, where difference is whether there exists self-loop. After the line search on hyper-parameters, we tune parameters from the following options:

- weight decay: $\{5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4}\}$
- range of local neighborhood: $\{0, 1, 2\}$
- α: $\{0, 0.1, 0.5, 1\}$
- hops: $\{2, 4, 5, 6\}$
- dropout rate: $\{0, 0.2, 0.5\}$
- λ : $\{0.2, 0.3, 0.4, 1\}$
- top k edges: $\{2\}$
- threshold: $\{0.5, 0.7, 1\}$

D.3 THE IMPACT OF NEIGHBORHOOD RANGE IN GCN-PND.

Figure 6 shows that aggregated features typically have higher performance than those before aggregation (0-hop) in both Semi-supervised node classification (SSNC) and full-supervised node classification (FSNC). It shows the effectiveness of updating graph topology. Specifically, the expansion of the neighborhood on updated graph does not introduce a lot of noise and useless information. Further, GCN-PND can effectively expand the range of neighborhood aggregation. From Figure 6, it is obvious that the performance of the model becomes stable with the increase of the range of neighborhood aggregation. In addition, the results show that our model can reach the peak of accuracy within 10-hop neighborhood. We think since new neighbors are added to each node when the graph topology is updated, the convergence speed of aggregation is accelerated. The results



(a) SSNC (b) FSNC

Figure 6: Figure (a) shows the impact of neighborhood range in semi-supervised node classification (SSNC). Figure (b) shows the impact of neighborhood range in full-supervised node classification (FSNC). The x-axis is the number of hops for aggregation and the y-axis is accuracy.

also show that an appropriate neighborhood range is conducive to improving the distinguishability of neighborhood distribution. Besides, the influence of neighborhood range on different datasets is different. For example, Squirrel is more easily affected by the neighborhood range, while Citeseer is less affected by the neighborhood range. We will study it in future work.