Neural Networks 168 (2023) 459–470

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Exploring the role of edge distribution in graph convolutional networks

Liancheng He [b], Liang Bai [a,b,*], Xian Yang [c], Zhuomin Liang [b], Jiye Liang [a,b]

[a] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, Taiyuan, 030006, Shanxi, China
[b] Institute of Intelligent Information Processing, Shanxi University, Taiyuan, 030006, Shanxi, China
[c] Alliance Manchester Business School, The University of Manchester, Manchester, UK

ARTICLE INFO

Keywords:
Graph Neural Networks
Heterophilous graphs
Node representation learning
Edge distribution
Neighbor selection

ABSTRACT

Graph Convolutional Networks (GCNs) have shown remarkable performance in processing graph-structured data by leveraging neighborhood information for node representation learning. While most GCN models assume strong homophily within the networks they handle, some models can also handle heterophilous graphs. However, the selection of neighbors participating in the node representation learning process can significantly impact these models' performance. To address this, we investigate the influence of neighbor selection on GCN performance, focusing on the analysis of edge distribution through theoretical and empirical approaches. Based on our findings, we propose a novel GCN model called Graph Convolution Network with Improved Edge Distribution (GCN-IED). GCN-IED incorporates both direct edges, which rely on local neighborhood similarity, and hidden edges, obtained by aggregating information from multi-hop neighbors. We extensively evaluate GCN-IED on diverse graph benchmark datasets and observe its superior performance compared to other state-of-the-art GCN methods on heterophilous datasets. Our GCN-IED model, which considers the role of neighbors and optimizes edge distribution, provides valuable insights for enhancing graph representation learning and achieving superior performance on heterophilous graphs.

1. Introduction

Graph-structured data is prevalent in real-life scenarios, encompassing social networks, citation networks, and biological networks. Learning meaningful representations from graph data has been a significant research focus due to its complexity and irregular relationships between nodes. Graph Neural Networks (Atwood & Towsley, 2016; Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2008), particularly Graph Convolutional Networks (GCNs) (Bruna, Zaremba, Szlam, & LeCun, 2014; Defferrard, Bresson, & Vandergheynst, 2016), have emerged as powerful algorithms for capturing rich graph representations by aggregating information from neighboring nodes. GCNs have achieved success in various tasks, including node classification (Kipf & Welling, 2017; Li, Zemel, Brockschmidt, & Tarlow, 2016), graph classification (Ma, Wang, Aggarwal, & Tang, 2019; Ying et al., 2018), and link prediction (You, Ying, & Leskovec, 2019; Zhang & Chen, 2018). While many GCNs are designed with the assumption of homophilous graphs (McPherson, Smith-Lovin, & Cook, 2001), where nodes of the same class tend to form edges. Some works have found that GCNs can perform well on some heterophilous graphs with low homophily (Luan et al., 2022; Ma, Liu, Shah, & Tang, 2021). This challenges the notion that GCNs are solely effective on homophilous graphs.

This study focuses on the performance of GCNs on heterophilous graphs, which are characterized by connections between nodes from different classes. Heterophilous graphs are prevalent in real-world scenarios and cannot be overlooked. Examples include dating networks, where individuals often connect with those of the opposite gender (Pandit, Chau, Wang, & Faloutsos, 2007), and protein structures, where amino acids from different classes tend to be connected (Zhu et al., 2020). While GCNs generally do not perform well on most heterophilous graphs, several improved methods have been proposed to address this challenge. For instance, Geom-GCN (Pei, Wei, Chang, Lei, & Yang, 2020) suggests a two-level aggregation approach that combines neighborhoods in both the graph space and latent space for aggregation. H_2GCN (Zhu et al., 2020) enhances GCN performance on heterophilous graphs through feature embedding and concatenation of nodes and multi-hop neighbors for downstream tasks. U-GCN (Jin et al., 2021) improves GCN performance on heterophilous graphs by combining kNN neighbors, 1-hop neighbors, and 2-hop neighbors for

* Corresponding author at: Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Shanxi University, Taiyuan, 030006, Shanxi, China.
E-mail addresses: heliancheng@alu.sxu.edu.cn (L. He), bailiang@sxu.edu.cn (L. Bai), xian.yang@manchester.ac.uk (X. Yang), 202112407007@email.sxu.edu.cn (Z. Liang), ljy@sxu.edu.cn (J. Liang).

https://doi.org/10.1016/j.neunet.2023.09.048
Received 26 February 2023; Received in revised form 3 September 2023; Accepted 28 September 2023
Available online 4 October 2023

aggregation. However, the underlying reasons for their effectiveness on heterophilous graphs still require further exploration.

In addition to the aforementioned studies, there have been other research efforts focused on understanding the influence of graph structure on neighborhood aggregation. GIN (Xu, Hu, Leskovec, & Jegelka, 2019) and k-GNN (Morris et al., 2019) have utilized the WL test (Leman & Weisfeiler, 1968) to analyze the ability of Graph Neural Networks (GNNs) to capture different graph structures. However, distinguishing between different graph structures is not always necessary. Instead, achieving optimal results in node classification requires node representations that are close within classes and distant between classes. Some clique methods (Luzhnica, Day, & Lio, 2019; Molaei, Bousejin, Zare, Jalili, & Pan, 2021) have been developed to update the graph structure by performing edge update operations based on graph denseness and the maximum connected subgraphs. However, these methods have limitations in their ability to update edges, and incorporating node features to calculate similarity or distance can enable edge updates at both local and global levels. Empirical findings in Ma et al. (2021) reveal that, with fine-tuned hyperparameters, the GCN (Kipf & Welling, 2017) can match or even outperform heterophily-specific models on certain heterophilous graphs. These results suggest the presence of a "good" type of heterophily characterized by distinguishable neighborhood distributions. However, the Cross-Class neighborhood Similarity (CCNS) metric used in Ma et al. (2021) may not be applicable to all heterophilous graphs. To avoid introducing complex graph structures, we propose utilizing neighborhood distribution and analyzing the process of neighborhood aggregation through edge distribution. This approach provides insights into the role of neighbors in graph representation learning without relying on explicit differentiation of graph structures.

In this paper, we introduce the concept of neighborhood distribution, which encompasses both edge distribution and feature distribution, to explore the impact of edge distribution on feature aggregation. Through theoretical and experimental analyses, we investigate the characteristics of a "good" edge distribution and examine how neighborhood affects the reliability of aggregated features. To enhance the edge distribution, we propose a method that utilizes local neighborhoods to update the direct edge distribution, improving the overall graph topology. Additionally, we present an extensible neighborhood aggregation module to complement the hidden edge distribution. Our main contributions can be summarized as follows:

- We conduct a comprehensive investigation into the effect of edge distribution on neighborhood aggregation, utilizing both theoretical and experimental approaches.
- We propose GCN-IED, a novel framework that improves the edge distribution of graphs by updating the graph topology and introducing an extensible neighborhood aggregation module.
- Extensive experimental results demonstrate the outstanding performance of our proposed model, particularly on heterophilous graphs.

2. Related work

GCNs (Hamilton, Ying, & Leskovec, 2017; Kipf & Welling, 2017; Vaswani et al., 2017) belong to the broader family of Graph Neural Networks (Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2009), which have gained significant popularity due to their utilization of graph convolution operators for aggregation. Researchers have explored various techniques to enhance the process of neighborhood aggregation within GCNs by modifying the neighborhood distribution. This distribution encompasses both the direct edge distribution and the hidden edge distribution, and efforts have been made to improve it from two primary perspectives.

Several approaches have been explored to modify the direct edge distribution by updating the graph topology. For example, GAT (Veličković et al., 2018), GATv2 (Brody, Alon, & Yahav, 2022), and CAT (Javaloy, Martin, Levi, & Valera, 2023) have improved neighborhood aggregation by optimizing edge weights and considering the edge perspective. LDS (Franceschi, Niepert, Pontil, & He, 2019) has introduced additional edges to the graph structure using a bi-level framework (Franceschi, Frasconi, Salzo, Grazzi, & Pontil, 2018). IDGL (Chen, Wu, & Zaki, 2020) has enforced smoothness between nodes and their neighbors through a smoothness loss. In the case of heterophilous graphs, Geom-GCN (Pei et al., 2020) and U-GCN (Jin et al., 2021) have achieved notable performance by identifying neighbors from high-order neighborhoods. Understanding the underlying reasons for the success of these methods on both homophilous and heterophilous graphs is an important consideration.

In addition to modifying the direct edge distribution, there have been efforts to change the neighborhood distribution by considering the hidden edge distribution. MixHop (Abu-El-Haija et al., 2019) and JKNet (Xu et al., 2018) have utilized high-order information through concatenation. Graph diffusion and propagation techniques have also gained popularity due to their simplicity and effectiveness. Methods such as PageRank (Andersen, Chung, & Lang, 2006; Klicpera, Bojchevski, & Günnemann, 2019; Page, Brin, Motwani, & Winograd, 1999) and heat kernel (Chamberlain et al., 2021; Kloster & Gleich, 2014; Kondor & Lafferty, 2002; Zhao, Dong, Ding, Kharlamov, & Tang, 2021) offer effective ways to adjust the influence of different hop neighbors during aggregation. While some methods (Chien, Peng, Li, & Milenkovic, 2021; Feng et al., 2020; Sun, Lin, & Zhu, 2021) have been designed to extract knowledge from high-order neighbors and have shown good performance on homophilous graphs, they often struggle on heterophilous graphs. Moreover, these methods often require careful hyperparameter tuning to control the weighting of different hop neighbors.

Furthermore, the updating of edges is closely related to other graph rewiring techniques. For example, Differentiable Graph Module (DGM) (Kazi, Cosmo, Ahmadi, Navab, & Bronstein, 2022) introduced a learnable function that predicts edge probabilities optimized for downstream tasks using the Gumbel-Top-k trick. Several works have explored the issue of over-smoothing in hidden edges, proposing new metrics (Chen, Lin, et al., 2020; Rusch, Bronstein, & Mishra, 2023) or studying the impact of aggregation steps on learning performance (Keriven, 2022). Homophily has also been a focus in some graph rewiring methods. FairDrop (Spinelli, Scardapane, Hussain, & Uncini, 2021) proposed a biased edge dropout algorithm to address homophily and enhance fairness in learning node embeddings. SELENE (Zhong, Gonzalez, Grattarola, & Pang, 2022) investigated learning embeddings in unsupervised heterophilous scenarios. ACM (Luan et al., 2022) examined heterophily from the perspective of post-aggregation node similarity and introduced new homophily metrics. However, it is important to note that edges form the foundation of the graph structure and determine which neighbors are aggregated. In contrast to the aforementioned approaches, our analysis focuses on how neighbor aggregation impacts performance after aggregation, providing a more intuitive understanding from the perspective of edges.

3. Notations and preliminaries

Consider a graph $G = (V, E)$ consisting of n nodes and m edges. Each edge connecting node i and node j can be represented as e_{ij}. The nodes in the graph are described by a feature matrix $\boldsymbol{X} \in \mathbb{R}^{n \times f}$, where f is the dimension of the node features. To indicate the class membership of each node, we use a label matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times C}$, where C represents the total number of classes. The element in the ith row and cth column of $\boldsymbol{Y}$ is denoted as $y_{i,c}$. The value of $y_{i,c}$ is 1 if node i belongs to class c, and 0 otherwise. Let $\boldsymbol{A}$ represent the adjacency matrix and $\boldsymbol{D}$ denote the diagonal degree matrix of $\boldsymbol{A}$. The adjacency matrix, including self-loops, is denoted as $\widetilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, and the degree matrix of $\widetilde{\boldsymbol{A}}$ is $\widetilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$. The symmetric normalized adjacency matrix with self-loops is given by $\hat{\boldsymbol{A}} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$. The edge distribution,

denoted as $\mathcal{E}$, represents the connected edges in the graph. For a specific class c, the edge distribution is denoted as $\mathcal{E}_c$. Specifically, let $|\mathcal{E}_c|$ denote the number of edges connected to class c, and $|\mathcal{E}_{cc'}|$ represent the number of edges between class c and class c'. The edge distribution of class c is described by $\mathcal{E}_c = \{p_{c,1}, \ldots, p_{c,c'}, \ldots, p_{c,C}\}$, where the values $p_{c,c'}$ can be approximated by $|\mathcal{E}_{cc'}|/|\mathcal{E}_c|$.

Homophily. In a graph, homophily refers to the tendency of edges to connect nodes within the same class. The homophily ratio of edges in a graph can be calculated as the fraction of edges connecting nodes with the same class label, as defined in Zhu et al. (2020):

$$r = \frac{1}{|\mathcal{E}|} \sum_{e_{i,j} \in \mathcal{E}} \boldsymbol{y}_i \boldsymbol{y}_j{}^T, \tag{1}$$

where $|\mathcal{E}|$ is the number of edges, $\boldsymbol{y}_i$ represents the ith row vector of the label matrix $\boldsymbol{Y}$, and $(\cdot)^T$ denotes the transpose operator. A graph with a high homophily ratio indicates that nodes are more likely to connect with others within the same class, while a graph with a homophily ratio close to 0 suggests that nodes from different classes are more likely to connect. It is important to note that heterogeneity, as defined in Zhang, Song, Huang, Swami, and Chawla (2019), is distinct from heterophily. Heterogeneous graphs, such as knowledge graphs, consist of multiple types of nodes and different types of relationships between nodes. In a content-associated heterogeneous graph defined in Zhang et al. (2019), the graph $G = (V, E, O_V, R_E)$ is composed of various types of nodes V and edges E, where O_V represents the set of object types and R_E represents the set of relation types.

Graph Convolutional Networks. The graph convolutional layer in Vanilla GCN (Kipf & Welling, 2017) is defined as:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)}\right), \tag{2}$$

where σ is the ReLU activation function.

In the spectral graph theory, the graph convolution operation is defined as $g_\gamma(\boldsymbol{L}) * \boldsymbol{x} = \boldsymbol{U} g_\gamma(\boldsymbol{\Lambda}) \boldsymbol{U}^T \boldsymbol{x}$, where $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$ with eigendecomposition $\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ and $g_\gamma(\boldsymbol{\Lambda}) = diag(\gamma)$ is a filter for dealing with signal $\boldsymbol{x}$. Furthermore, the graph convolution operation can be approximated using a polynomial of the Laplacian matrix, as shown by

$$\boldsymbol{U} g_\gamma(\boldsymbol{\Lambda}) \boldsymbol{U}^T \boldsymbol{x} \approx \boldsymbol{U}\left(\sum_{k=0}^{K} \gamma_k \boldsymbol{\Lambda}^k\right)\boldsymbol{U}^T\boldsymbol{x} = \left(\sum_{k=0}^{K} \gamma_k \boldsymbol{L}^k\right)\boldsymbol{x}, \tag{3}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{K+1}$ is a vector of polynomial coefficients. In the vanilla GCN, $K = 1$ and the graph convolution is performed using the renormalization trick. It is evident that the polynomial form provides a more general framework for describing the graph convolution process, which can be expressed as

$$\boldsymbol{h} = \left(\sum_{k=0}^{K} \theta_k \boldsymbol{T}^k\right)\boldsymbol{x}, \tag{4}$$

where $\boldsymbol{T}$ is a matrix that can be learned or computed to aggregate neighborhood information, θ_k represents the weight coefficient for the k-hop neighbors of x, and K determines the range of aggregation.

4. Theoretical and empirical analysis of the impact of edge distribution

4.1. Theoretical analysis

To gain a deeper understanding of how graph topology affects node features during neighborhood aggregation, we introduce the concept of edge distribution, which comprises two key components: the direct edge distribution and the hidden edge distribution. The direct edge distribution primarily influences the graph's topology, while the hidden edge distribution is closely related to the presence of multi-hop neighbors.

In order to facilitate our analysis, we introduce the following assumptions for each node i belonging to class c: (1) the feature of node i, denoted as $\boldsymbol{x}_i$, follows the distribution $\mathcal{F}_c$, i.e., $\boldsymbol{x}_i \sim \mathcal{F}_c$; (2) the direct connection between node i and another node j, represented by the edge e_{ij}, follows the distribution $\mathcal{E}_c$, i.e., $e_{ij} \sim \mathcal{E}_c$. Here, $\mathcal{E}_c$ represents the edge distribution specific to class c.

The neighborhood distribution of class c is denoted as $\mathcal{D}_c = \{\mathcal{F}_c, \mathcal{E}_c\}$, which determines the node features after aggregation. Suppose the expectation and variance of the feature vectors $\boldsymbol{x}_i$ from class c are $\mathbb{E}_c[\boldsymbol{x}_i] = \boldsymbol{\mu}_c \in \mathbb{R}^f$ and $\mathbb{D}_c[\boldsymbol{x}_i] = \boldsymbol{\Sigma}_c \in \mathbb{R}^{f\times f}$ (f is the dimension of features), respectively. For each node i, its neighborhood aggregation is defined as $\boldsymbol{u}_i = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \boldsymbol{x}_j$, where $\mathcal{N}(i)$ denotes the direct neighbors of node i. Then, we have the following theorem:

Theorem 4.1. *Consider a graph in which the feature of each node i from class c satisfies $\mathbb{E}_c[\boldsymbol{x}_i] = \boldsymbol{\mu}_c$ and $\mathbb{D}_c[\boldsymbol{x}_i] = \boldsymbol{\Sigma}_c$. Let $p_{c,c'}$ denote the connection probability between class c and class c'. Under the assumption that node features are independent of each other, when enough edges are added, the aggregated feature of node i satisfies the following condition:*

$$\boldsymbol{u}_i \sim^a N\left(\frac{\sum_{c'=1}^{C} p_{c,c'}\boldsymbol{\mu}_{c'}}{\sum_{c'=1}^{C} p_{c,c'}}, \frac{\sum_{c'=1}^{C} p_{c,c'}\boldsymbol{\Sigma}_{c'}}{|\mathcal{N}(i)|\sum_{c'=1}^{C} p_{c,c'}}\right). \tag{5}$$

Proof. The number of connected edges between node i and nodes from class c' is $\frac{p_{c,c'}}{\sum_{c'=1}^{C} p_{c,c'}}|\mathcal{N}(i)|$. Since the node features in the same class satisfy the independent identical distribution condition, according to Lindeberg-Levy CLT (central limit theorem), we have the following:

$$\sum_{e_{ij}\in\mathcal{E}, y_{j,c'}=1} \boldsymbol{x}_j \sim^a N\left(\frac{p_{c,c'}|\mathcal{N}(i)|}{\sum_{c'=1}^{C} p_{c,c'}}\boldsymbol{\mu}_{c'}, \frac{p_{c,c'}|\mathcal{N}(i)|}{\sum_{c'=1}^{C} p_{c,c'}}\boldsymbol{\Sigma}_{c'}\right). \tag{6}$$

For all classes, the aggregated information of node i is as follows

$$\boldsymbol{u}_i = \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} \boldsymbol{x}_j = \frac{1}{|\mathcal{N}(i)|}\sum_{c'=1}^{C}\sum_{e_{ij}\in\mathcal{E}, y_{j,c'}=1} \boldsymbol{x}_j. \tag{7}$$

Hence, $\boldsymbol{u}_i$ consists of C Normal Distributions approximately and asymptotically obeys the following distribution:

$$\begin{aligned}\boldsymbol{u}_i &\sim^a N\left(\sum_{c'=1}^{C}\frac{p_{c,c'}|\mathcal{N}(i)|\boldsymbol{\mu}_{c'}}{|\mathcal{N}(i)|\sum_{c'=1}^{C} p_{c,c'}}, \sum_{c'=1}^{C}\frac{p_{c,c'}|\mathcal{N}(i)|\boldsymbol{\Sigma}_{c'}}{|\mathcal{N}(i)|^2\sum_{c'=1}^{C} p_{c,c'}}\right)\\ &= N\left(\frac{\sum_{c'=1}^{C} p_{c,c'}\boldsymbol{\mu}_{c'}}{\sum_{c'=1}^{C} p_{c,c'}}, \frac{\sum_{c'=1}^{C} p_{c,c'}\boldsymbol{\Sigma}_{c'}}{|\mathcal{N}(i)|\sum_{c'=1}^{C} p_{c,c'}}\right).\end{aligned} \tag{8}$$

which completes the proof. □

By examining Eq. (5), it can be observed that the variance of the neighborhood distribution is influenced by the degree of nodes. Specifically, as the degree of nodes increases, the variance tends to decrease. This relationship indicates that when there are a sufficient number of edges, we can expect the node representation $\boldsymbol{u}_i$ to be close to its expected value $\mathbb{E}_c[\boldsymbol{u}_i]$. Another theorem further describes the relationship between $\boldsymbol{u}_i$ and its expectation $\mathbb{E}_c[\boldsymbol{u}_i]$.

Theorem 4.2. *Suppose all elements of $\boldsymbol{u}_i$ are bounded in $[a, b]$. For $t_1 \geq f \cdot t_2 > 0$, the probability of the distance between $\boldsymbol{u}_i$ and its expectation $\mathbb{E}_c[\boldsymbol{u}_i]$ larger than t_1 is bounded by*

$$\mathbb{P}\left(\|\boldsymbol{u}_i - \mathbb{E}_c[\boldsymbol{u}_i]\|_1 \geq t_1\right) \leq 2f \cdot exp\left(-\frac{2|\mathcal{N}(i)|t_2^2}{(b-a)^2}\right). \tag{9}$$

Proof. Before proving Theorem 4.2, we introduce Hoeffding's inequality as follows: Let $q_1, \ldots, q_n$ be independent bounded random variables with $q_i \in [a, b]$, where $i \in [1, n]$ and $-\infty < a \leq b < \infty$. For all $t \geq 0$,

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}\left(q_i - \mathbb{E}[q_i]\right)\right| \geq t\right) \leq 2 \cdot exp\left(-\frac{2nt^2}{(b-a)^2}\right). \tag{10}$$

Let $\boldsymbol{u}_i[k]$ be the kth element of $\boldsymbol{u}_i$. Suppose for all i and k, $\boldsymbol{u}_i[k] \in [a, b]$ and $-\infty < a \leq b < \infty$. Following Hoeffding's inequality, for all $t_k \geq 0$

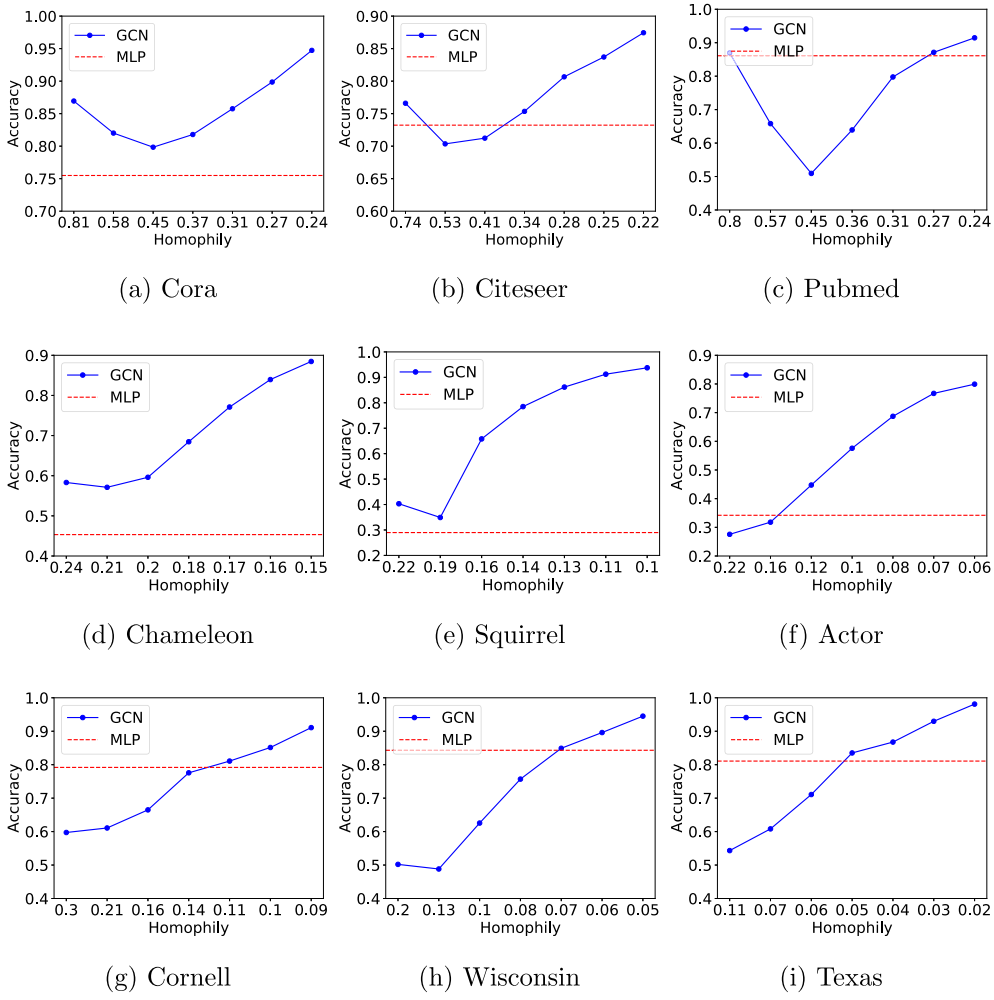


Fig. 1. The performance of GCN is plotted as a function of the homophily on the x-axis and the accuracy on the y-axis, with the increasing number of inter-class edges. The red dotted line shows the MLP prediction accuracy.

we have:

$$\begin{aligned}&\mathbb{P}\left(\left|\boldsymbol{u}_i[k]-\mathbb{E}_c[\boldsymbol{u}_i[k]]\right|\geq t_k\right)\\&=\mathbb{P}\left(\left|\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\left(\boldsymbol{x}_j[k]-\mathbb{E}[\boldsymbol{x}_j[k]]\right)\right|\geq t_k\right)\\&\leq 2\cdot exp\left(-\frac{2|\mathcal{N}(i)|t_k^2}{(b-a)^2}\right).\end{aligned}\tag{11}$$

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

$$\begin{aligned}&\mathbb{P}\left(\|\boldsymbol{u}_i-\mathbb{E}_c[\boldsymbol{u}_i]\|_1\geq\sum_{k=1}^{f}t_k\right)\\&\leq 2\sum_{k=1}^{f}exp\left(-\frac{2|\mathcal{N}(i)|t_k^2}{(b-a)^2}\right)\leq 2f\cdot exp\left(-\frac{2|\mathcal{N}(i)|t_{min}^2}{(b-a)^2}\right),\end{aligned}\tag{12}$$

where $t_{min}=min(t_1,\ldots,t_k,\ldots,t_f)$.

Let $t_1=\sum_{k=1}^{f}t_k$ and $t_2=t_{min}$, we have $t_1\geq f\cdot t_2$ and

$$\mathbb{P}\left(\|\boldsymbol{u}_i-\mathbb{E}_c[\boldsymbol{u}_i]\|_1\geq t_1\right)\leq 2f\cdot exp\left(-\frac{2|\mathcal{N}(i)|t_2^2}{(b-a)^2}\right),\tag{13}$$

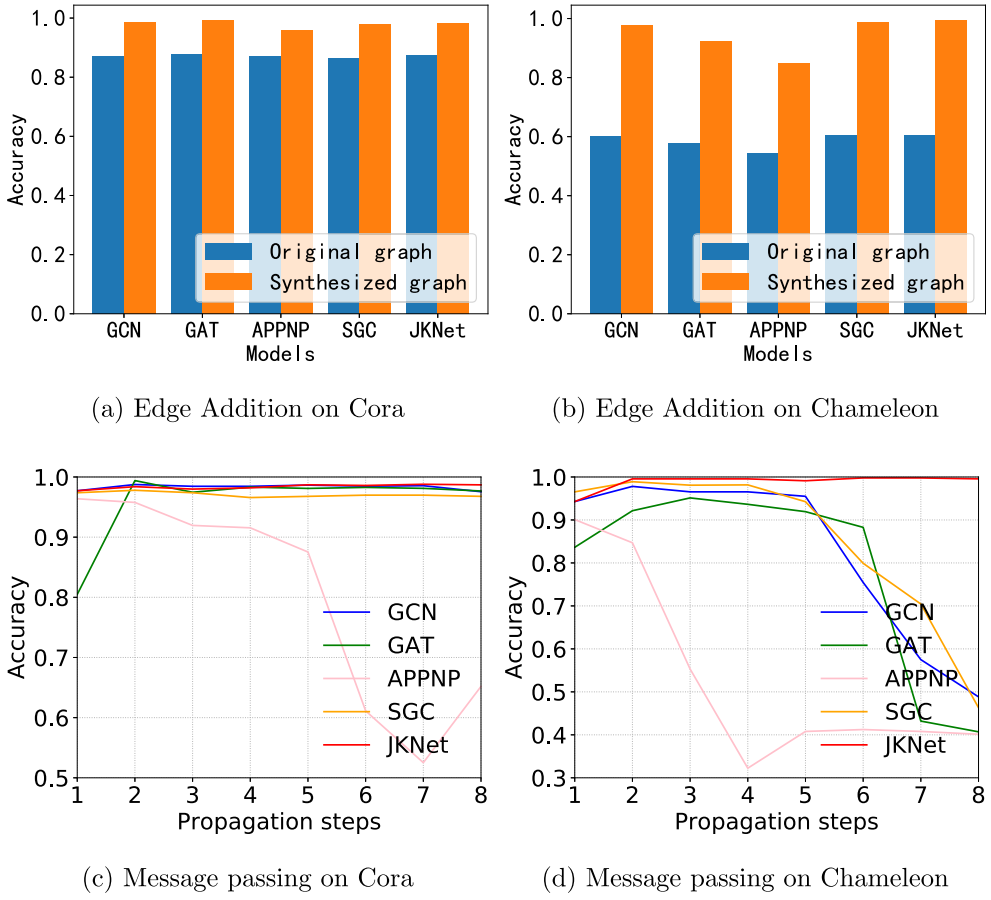
which completes the proof. □

Theorem 4.2 highlights that there is a high probability for the aggregated node features to be close to their expectations when the degree of nodes is high. This implies that as the degree of nodes increases, the node features tend to approach their expected values. In order to ensure distinguishable aggregated node features, it is important for the feature distributions between different classes to be far apart from each other. Moreover, the applicability of the theorem can be extended to Message Passing Phase in MPNN. During this phase, the edge indices serve to record the direct neighbors surrounding each node, and neighborhood information is aggregated using edges for node updates. To validate the findings of this theorem, we will proceed with conducting experimental analyses.

4.2. Empirical investigation of edge distribution

4.2.1. Empirical analysis of GCN

We conducted experiments using a two-layer GCN on nine different graphs to evaluate our assumptions and conclusions. The GCN model utilized two layers with the ReLU activation function applied after the first layer, and the hidden units were set to 64. The dropout rate of 0.5 was applied to the node features. We employed the Adam optimizer with a learning rate of 0.01 and utilized the cross-entropy loss function. The L_2 regularization parameter was set to 5×10^{-4} for all datasets. The early stopping criterion used a patience of $p=100$ and a maximum of $n=1500$ epochs. To investigate the impact of homophily, we introduced inter-class edges sampled from specific distributions, thereby varying the level of homophily in the graphs. The accuracy results were recorded and presented in Fig. 1. Based on our observations, the following conclusions can be drawn:



(a) Edge Addition on Cora (b) Edge Addition on Chameleon

(c) Message passing on Cora (d) Message passing on Chameleon

Fig. 2. Impact of edge distribution on multiple MPNN models. (a) and (b) compare the accuracy of node classification for each model on the original graph and the synthesized graph after adding inter-class edges to reduce homophily. (c) and (d) explore the influence of neighborhood variation on aggregation performance.

- Increasing the number of edges between classes led to a decrease in homophily for each graph. However, the accuracy did not always decrease accordingly. In fact, all nine datasets exhibited high accuracy despite having low homophily. This suggests the presence of a "good" edge distribution in heterophilous graphs, which can enhance the neighborhood distribution for effective aggregation.
- neighborhood aggregation in homophilous graphs outperformed the Multilayer Perceptron (MLP) baseline, as depicted in Figs. 1(a), 1(b), and 1(c). In heterophilous graphs, two distinct cases were observed, as shown in Figs. 1(d) to 1(i). A "good" edge distribution improved the node representation after aggregation, while a "bad" edge distribution made it difficult to distinguish the aggregated features. These findings suggest that homophilous graphs naturally possess a "good" edge distribution, while heterophilous graphs exhibit more uncertainty in this regard.

The experimental results highlight the substantial influence of edge distribution on the performance of GCNs. Notably, a "good" edge distribution contributes to a more distinct neighborhood distribution, thereby enhancing the effectiveness of GCNs. Consequently, it becomes crucial to devise strategies for improving the edge distribution to optimize neighborhood aggregation. However, constructing a "good" edge distribution with low homophily poses a considerable challenge. Consequently, we delve into the transformation of heterophilous graphs into homophilous graphs, specifically focusing on the identification of suitable neighbors for each node.

4.2.2. Comprehensive analysis of edge distribution on MPNNs

MPNNs (Message Passing Neural Networks) (Gilmer, Schoenholz, Riley, Vinyals, & Dahl, 2017) serve as a versatile framework for message passing on graphs, consisting of two key phases: the message passing and the readout phases. In order to evaluate the wide-ranging applicability of the theory expounded in Section 4.1, we conducted a comprehensive analysis using various MPNNs on two extensively employed datasets: Cora and Chameleon. The findings are visually represented in Fig. 2. The edge addition procedure, as described in Section 4.1, was applied, resulting in homophily values of 0.24 for Cora and 0.15 for Chameleon. Refer to Table 1 for a detailed breakdown of these values. Notably, all models leveraged two-hop neighbor information without the inclusion of any regularization terms.

The results shown in Figs. 2(a) and 2(b) indicate that the model performs better on the synthesized dataset than the original dataset. This suggests that the inclusion of inter-class edges does not necessarily have a negative impact on the model's predictive capability. Furthermore, the distinguishability of aggregated node features tends to be stronger when the neighboring edges of each class follow a specific distribution and are more abundant. This observation is consistent with the theory discussed in Section 4. On the other hand, Figs. 2(c) and 2(d) demonstrate that expanding the neighborhood in heterophilous graphs can lead to a rapid decline in performance due to the indistinguishable nature of aggregated node features. Among the models examined, JKNet exhibits the highest stability. This can be attributed to JKNet's ability to concatenate and preserve neighbor information from different layers, effectively mitigating information pollution during neighborhood expansion.

Table 1
Inter-class Edge Addition in Different Datasets: γ denotes the added edge ratio, and r represents the resulting homophily.

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

4.2.3. Impact of neighborhood range on neighborhood aggregation

To find reliable neighbors, we need to evaluate node features. We utilize the SGC model (Wu et al., 2019) to investigate how the range of the neighborhood influences the hidden edge distribution. The algorithm for adding edges based on the distribution is outlined in Algorithm 1. To ensure fast and effective convergence, we specifically add inter-class edges sampled from a predetermined edge distribution, where one-way edges are generated from class c_k to c_{k+1}. To avoid creating an imbalance where some classes have a high concentration of edges while others have very few or no added edges, we recalculate the distribution based on the number of edges in each class. Initially, we determine the edge distribution by calculating the proportion of the number of edges between two classes in relation to the total number of edges. Using this initial edge distribution and the predetermined edge distribution, we generate the edges. However, slight modifications are made to the algorithm for the Texas and Cornell datasets due to the presence of isolated nodes (classes with only one node and no edges). In these cases, our default class distribution sampling number is at least 1.

Algorithm 1 Adding edges sampling from distribution.

Input: A, D
Parameter: γ
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 $n = m * \gamma$.
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 while
Update A using edge set $\mathcal{E}$.
return updated adjacency matrix A'.

The procedure for adding inter-class edges based on the distribution is summarized in Table 1. Each dataset is represented by two rows of data. The first row indicates the ratio of added edges (γ) relative to the original number of edges, while the second row represents the resulting homophily (r) after the edges are added. For instance, in the Cora graph, the initial homophily is 0.81 ($r = 0.81$) with no added edges ($\gamma = 0$). After increasing the number of inter-class edges by a factor of 2, the homophily decreases to approximately 0.27 ($r = 0.27$) with $\gamma = 2$.

Empirical results: One common approach to updating the graph topology is by utilizing node features to identify k-nearest neighbors. However, relying solely on node features may not yield optimal results, as valuable information from the edge distribution can be overlooked. In Fig. 3(a), we introduce a specific edge distribution to the Cora dataset, creating synthetic datasets with varying levels of homophily. In Fig. 3(b), we compare the influence of different-hop neighbors on post-aggregation performance across nine datasets. Our observations indicate that expanding the neighborhood has little adverse impact on homophilous graphs. In heterophilous graphs, local neighborhood information within a 2-hop radius exhibits greater reliability for classification, while hidden edges connected to higher-order neighbors can have a detrimental effect on neighborhood aggregation. These findings suggest the feasibility of utilizing local neighborhood information to identify neighbors and update the original edge distribution.

5. The GCN-IED model

As illustrated in Fig. 4, we propose the GCN-IED model for obtaining direct edges by updating the graph topology and exploring hidden edges from multi-hop neighbors through extensible neighborhood aggregation.

5.1. Updating graph topology

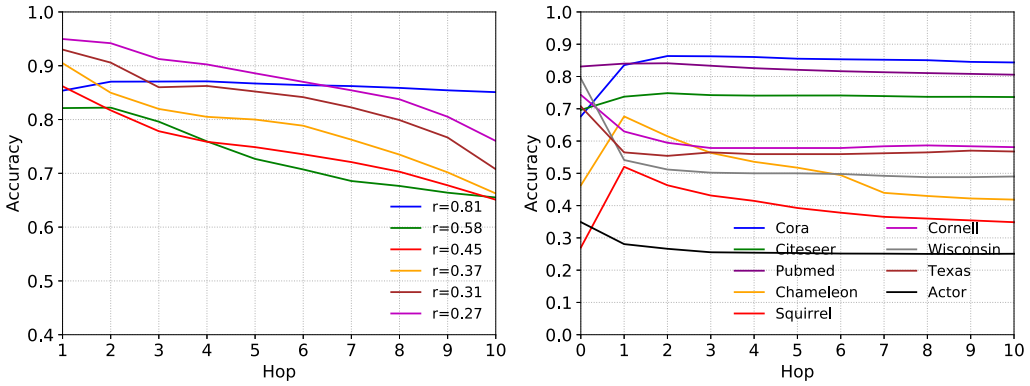
Before updating the graph topology, it is necessary to generate a similarity matrix that measures the distance between nodes. This similarity matrix, denoted as $\boldsymbol{S} \in \mathbb{R}^{n \times n}$, is obtained by computing the cosine similarity between node embeddings. These node embeddings capture information from the neighborhood through the following process:

$$\boldsymbol{H}^{(L)} = (\hat{\boldsymbol{A}})^L \boldsymbol{H}^{(0)}, \tag{14}$$

where $\boldsymbol{H}^{(0)} = f_\theta(\boldsymbol{X})$, and $f_\theta(\boldsymbol{X})$ represents a fully connected neural network (FCNN) with ReLU activation applied to the feature matrix $\boldsymbol{X}$. The matrix $(\hat{\boldsymbol{A}})^L$ denotes the Lth power of the matrix $\hat{\boldsymbol{A}}$, which incorporates information from the 1 to L-hop neighbors in the computation of node embeddings for similarity measurement. In this paper, we set $L \leq 2$ to limit the influence on the local neighborhood range and avoid interference from high-order information.

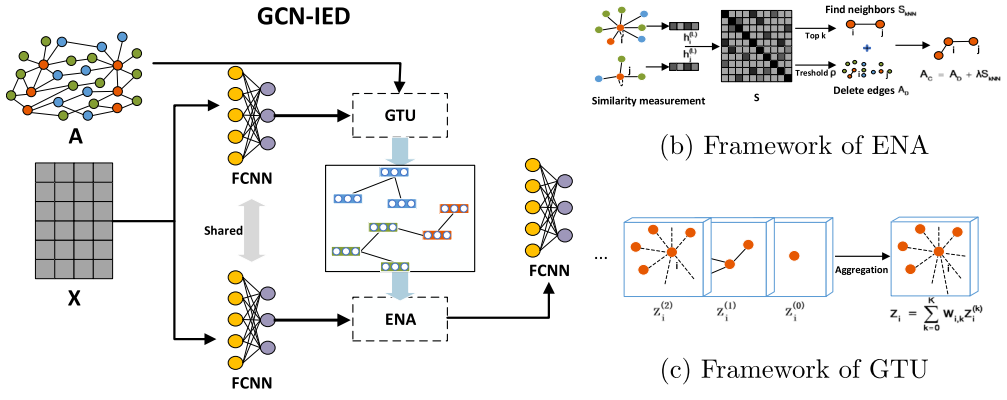
Let $\boldsymbol{h}_i^{(L)} \in \mathbb{R}^f$ denote the ith row vector of $\boldsymbol{H}^{(L)}$. The element at the ith row and jth column of $\boldsymbol{S}$ is given by:

$$s_{ij} = cosine(\boldsymbol{h}_i^{(L)}, \boldsymbol{h}_j^{(L)}), \tag{15}$$



(a) Synthetic datasets from Cora (b) Nine common datasets

Fig. 3. The impact of the neighborhood range in SGC on (a) synthetic datasets from Cora and (b) nine common datasets.



(a) Framework of GCN-IED (b) Framework of ENA (c) Framework of GTU

Fig. 4. Illustration of the proposed method. The method consists of two main components: Graph Topology Update (GTU) and Extensible neighborhood Aggregation (ENA) module that incorporates information from multi-hop neighbors.

We then get the matrix $\boldsymbol{S}_{kNN}$ by remaining the top k similar node pairs for each node, whose element on ith row and jth column is:

$$s_{i,j}^{kNN} = \begin{cases} s_{i,j}, & \text{if } s_{i,j} \geq \varphi_{k,i}, \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Here, $\varphi_{k,i}$ is the kth largest $s_{i,j}$ for all j.

The adjacency matrix of the updated graph topology is obtained as follows:

$$\boldsymbol{A}_C = \boldsymbol{A}_D + \lambda \boldsymbol{S}_{kNN}, \tag{17}$$

where $\boldsymbol{A}_D$ is derived from the adjacency matrix $\hat{\boldsymbol{A}}$ of the original topology, where only the elements larger than ρ in $\boldsymbol{S}$ are retained. The parameter λ controls the balance between the two adjacency matrices. The matrix $\boldsymbol{A}_C$ is further normalized as $\widetilde{\boldsymbol{A}}_C$ via:

$$\widetilde{a}_{i,j} = \frac{exp\left(a_{i,j}\right)}{\sum_{k \in \mathcal{N}_i} exp\left(a_{i,k}\right)}, \tag{18}$$

where $a_{i,j}$ denotes the edge weight of node i connected to node j.

5.2. Designing extensible neighborhood aggregation

Once we have the updated graph topology, the next step is to discover hidden edges through extensible neighborhood aggregation. The feature matrix $\boldsymbol{Z}^{(k)}$, obtained by propagating features from the k-hop neighbors using the matrix $\widetilde{\boldsymbol{A}}_C$, is expressed as follows:

$$\boldsymbol{Z}^{(k)} = (\widetilde{\boldsymbol{A}}_C)^k \boldsymbol{H}^{(0)}. \tag{19}$$

Let $\boldsymbol{z}_i^{(k)}$ denote the ith row vector of $\boldsymbol{Z}^{(k)}$, representing the information aggregated from the k-hop neighbors of node i. The aggregated information for each node i can be expressed as follows:

$$\widetilde{\boldsymbol{z}}_i = \sum_{k=0}^{K} \frac{exp\left(\boldsymbol{v}_i \boldsymbol{z}_i^{(k)}\right)}{\sum_{k=0}^{K} exp\left(\boldsymbol{v}_i \boldsymbol{z}_i^{(k)}\right)} \boldsymbol{z}_i^{(k)}, \tag{20}$$

where the trainable vector $\boldsymbol{v}_i \in \mathbb{R}^f$ is used to learn the importance of each k-hop embedding of node i. The distribution of neighbors around each node may vary, and thus, we set a trainable vector for each node. Unlike U-GCN, which focuses only on adding kNN graph information, we optimize the aggregated graph structure by considering both direct edges and hidden edges. Subsequently, the vector $\widetilde{\boldsymbol{z}}_i$ is passed through a fully connected neural network (FCNN) with the Softmax activation function to obtain the prediction vector $\widetilde{\boldsymbol{y}}_i$. The prediction loss is calculated using the cross entropy loss function:

$$\mathcal{L} = -\sum_{i \in \mathcal{Y}^y} \sum_{c=1}^{C} y_{i,c} log \widetilde{y}_{i,c}, \tag{21}$$

where $y_{i,c}$ is a binary indicator (0 or 1) equal to 1 if the correct label of node i is c, $\widetilde{y}_{i,c}$ is the cth element of $\widetilde{\boldsymbol{y}}_i$, and $\mathcal{Y}^y$ is the set of labeled nodes. The training process of our GCN-IED model is outlined in Algorithm 2. The convergence criterion in the algorithm is inspired by GAT (Veličković et al., 2018). Specifically, training stops when the loss on the validation set does not decrease for a certain number of consecutive rounds, indicating convergence.

Table 2
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(r)	0.81	0.74	0.8	0.24	0.22	0.22	0.31	0.21	0.11

Algorithm 2 The GCN-IED model.

Input: Feature matrix $\mathbf{X}$, adjacent matrix $\mathbf{A}$.
Parameter: neighborhood range (hops) K, local neighborhood range L, hyperparameter k of kNN graph, threshold ρ.
Prediction: $\widetilde{Z}$.

while not convergence **do**
  Encode feature matrix $\mathbf{X}$ using a fully connected network $f_\theta(X)$.
  Calculate the similarity values between nodes using Eq.(15).
  Get the matrix $\boldsymbol{S}_{kNN}$ from Eq.(16).
  Get the matrix $\boldsymbol{A}_D$ by setting elements in $\hat{\boldsymbol{A}}$ smaller than the threshold ρ to 0.
  Get the adjacent matrix $\boldsymbol{A}_C$ of the updated graph using Eq.(17).
  Get the normalized version $\widetilde{\boldsymbol{A}}_C$ from Eq.(18).
  for $k = 0$ to K **do**
    Get $\boldsymbol{Z}^{(k)}$ through propagating features from the k-hop neighbors using $\widetilde{\boldsymbol{A}}_C$ as expressed in Eq.(19).
  end for
  Aggregate multi-hop neighborhood to get $\widetilde{\boldsymbol{z}}_i$ using Eq.(20).
  Use $\widetilde{\boldsymbol{z}}_i$ to get the predicted results.
  Minimize the prediction loss $\mathcal{L}$ using Eq.(21).
end while

6. Experiments

6.1. Experimental setup

6.1.1. Datasets

We evaluate the performance of our model and other baselines in both semi-supervised and full-supervised node classification tasks. For semi-supervised node classification, we utilize three standard citation network datasets: Cora, Citeseer, and Pubmed (Sen et al., 2008). These datasets exhibit strong homophily. The data is divided into three sets using the standard fixed split method (Yang, Cohen, & Salakhutdinov, 2016): 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for testing.

For full-supervised node classification, we employ 9 graph datasets. The dataset statistics are summarized in Table 2, which include information about the number of nodes, edges, features, classes, and homophily. Following the approach of Geom-GCN (Pei et al., 2020), we perform 10 random splits of nodes per class for each dataset. The training, validation, and testing sets are divided with a 48%/32%/20% split, respectively.

6.1.2. Baselines methods and implementation details

Our GCN-IED model is compared with several baselines in both semi-supervised and full-supervised node classification tasks.

For semi-supervised node classification, we compare GCN-IED with the following baselines: 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). GCN, GAT, and SGC are implemented with their original settings, while GAT uses the sparse version. The propagation step K is set to 10 for APPNP and GPRGNN, and 16 for SSGC. The parameter α is set to 0.1 for APPNP and GPRGNN, and 0.05 for SSGC. In our GCN-IED model, we set the weight decay parameter to 5×10^{-4} and the learning rate to 0.01. The early stopping criterion has a patience of 100 and a maximum of 1500 iterations for all datasets. For aggregation, the hidden layer has 64 hidden units, and a dropout rate of 0.5 is applied to the feature tensor. For updating the graph topology, we use 1-hop neighbors for neighborhood aggregation. We set $k = 2$ for kNN graphs on all datasets and use thresholds $\rho = 0.4, 0.5, 0.5$ for edge deletion in the Cora, Citeseer, and Pubmed datasets, respectively. For aggregation from multi-hop neighbors, we use 10, 4, and 10 propagation steps. After fine-tuning, we set the dropout rate to 0.5 and 0.8 on the input and hidden layers in the Cora dataset, and 0.5 and 0.5 in the Citeseer and Pubmed datasets.

For full-supervised node classification, we compare our GCN-IED model with 17 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), dDGM (Kazi et al., 2022), ACM-GCN (Luan et al., 2022) and its variant and MLP. All models except dDGM use 64 hidden units in the hidden layer. dDGM uses the default settings including 32 and 16 hidden units in different hidden layers. And a patience of 100 is set for the early stopping criterion in all models. MLP, GCN, GAT, GraphSAGE, and JKNet use a weight decay parameter of 5×10^{-4}. In particular, we reproduce GraphSAGE using the algorithm from the original paper without node sampling. JKNet uses concatenation for the aggregation layers (5 layers). We use the hyperparameters from the original papers for GPRGNN, GCNII, Geom-GCN, and H2GCN. Furthermore, the hyperparameters of ACM-GCN and its variant are the same as in the original paper. In our GCN-IED model, we use an adjacency matrix with self-loops. After a line search on the hyperparameters, we fine-tune the parameters from the following options: weight decay $(5 \times 10^{-5}, 10^{-4}, 5 \times 10^{-4})$, range of local neighborhood $(0, 1, 2)$, hops $(1, 2, 3, 4, 6, 9)$, dropout rate $(0, 0.2, 0.5)$, top k edges $(1, 2, 4)$, and threshold $(0.4, 0.6, 1)$.

6.2. Performance of semi-supervised node classification

Table 3 presents the classification accuracy along with the mean and standard deviation after 50 runs. The results for MixHop are directly taken from its original paper, as the reproduced results did not match the original values. From Table 3, it can be observed that our GCN-IED model achieves significantly higher accuracy compared to the other models. Furthermore, models that expand the range of the neighborhood, such as SSGC and GPRGNN, also demonstrate high accuracy across all datasets. This indicates that properly expanding the neighborhood in a homophilous graph can enhance the distinguishability of the neighborhood distribution.

To analyze the contributions of different components in GCN-IED, we conducted an ablation study as follows: (1) Using the original graph for aggregation from multi-hop neighbors without graph topology updating (w/o GTU). (2) Using the updated graph topology of two-hop neighbors for aggregation, which is similar to performing graph topology updates on GCN without adaptive multi-hop aggregation (w/o AMA). As shown in Table 3, updating the graph topology plays a crucial role. Similarly, aggregating two-hop neighbors (w/o AMA) improves the average accuracy by nearly 1.8% compared to GCN. The extensible neighborhood aggregation further improves the overall performance.

Table 3
Summary of semi-supervised classification accuracy (%) on Cora, Citeseer, and Pubmed. Results marked with † are reproduced from the original paper. The best results are highlighted in bold.

Method	Cora	Citeseer	Pubmed
GCN	81.4±0.7	70.8±0.7	79.1±0.3
GAT	82.1±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-IED	**85.2 ± 0.6**	**73.1 ± 0.4**	**81.2 ± 0.5**
w/o GTU	85.1±0.4	72.4±0.6	80.1±0.4
w/o AMA	84.2±0.5	72.0±0.5	80.4±0.7

6.3. Performance of full-supervised node classification

We evaluate the performance of GCN-IED using the mean accuracy across 10 splits in the task of full-supervised classification. The results are summarized in Table 4, where the best results are highlighted in bold while underlined letters indicate the second best. The AVG column indicates the average of the model's accuracy over all datasets. It is obvious that GCN-IED matches or achieves state-of-the-art performance on most datasets compared to the other 17 baselines. This indicates that GCN-IED is highly effective especially in handling heterophilous graphs. On the Squirrel dataset, which is a heterophilous graph, GCN-IED achieves an accuracy nearly 11% higher than the best baseline, ACM-GCN. This demonstrates the strong ability of GCN-IED to process heterophilous graphs and extract meaningful node representations. Although dDGM also updates the graph topology, it does not perform well in the experiments. This shows finding a "good" edge distribution is beneficial to neighbor aggregation and enhancing distinguishability of aggregated node feature. On the Cornell, Wisconsin, and Texas datasets, where the node features are distinguishable between classes, we use the 0-hop neighborhood distribution for graph topology updating. This choice further improves the performance of GCN-IED.

We also conduct ablation experiments to examine the contributions of the different components in GCN-IED. The two modules, namely w/o GTU (without Graph Topology Update) and w/o AMA (without Adaptive Multi-hop Aggregation), are evaluated. From the results in Table 4, we observe that using an updated graph topology in GCN (w/o AMA) improves its performance on all datasets, particularly on heterophilous graphs. Especially, its AVG accuracy is higher than 17 compared modes, which indicates the importance of a "good" edge distribution. Furthermore, the average performance of GCN-IED decreases when any component is removed, indicating the significance of the designed components in enhancing the performance of GCN-IED.

6.4. Sensitivity analysis

6.4.1. Impact of threshold ρ

The effect of the hyperparameter ρ is examined in Fig. 5. In homophilous graphs, we observe that the accuracy tends to increase and then decrease as the threshold value increases. This suggests that the original graph structure is already well-suited for the task and only requires minor adjustments. In heterophilous graphs, we find that the performance tends to improve as the threshold increases. However, the performance of the Cornell and Wisconsin datasets fluctuates as the threshold increases, while the Chameleon and Squirrel datasets show significant improvements only when the original graph is completely removed. The Actor dataset is less affected by the threshold value and exhibits less sensitivity. The results also indicate that the best performance in reconstructing the edge distribution for heterophilous graphs is achieved with a threshold value of 1. This suggests that a higher threshold value leads to a lower weight for the original graph and a higher weight for the edges in the kNN graph. Moreover, as the number of edges in the kNN graph increases, each edge weight should decrease accordingly.

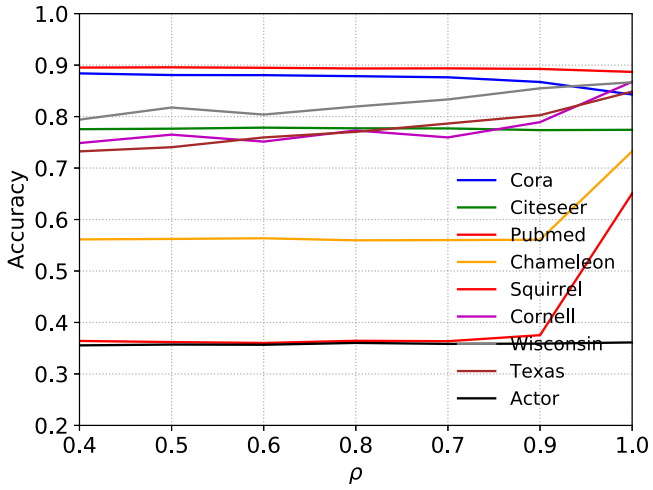


Fig. 5. Sensitivity analysis of the threshold ρ on different datasets, exploring its effect on performance.

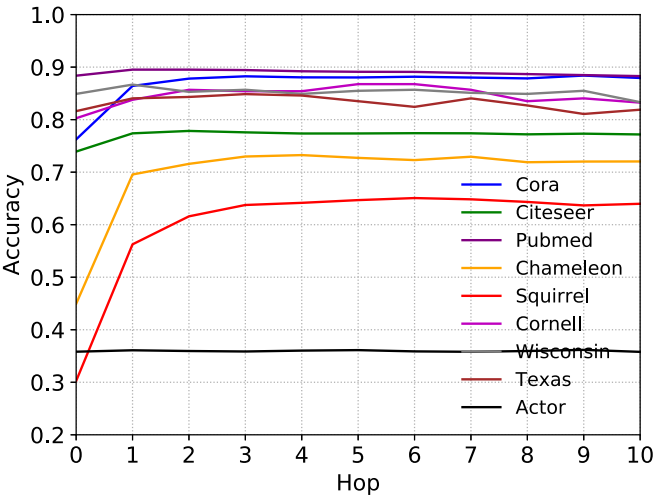


Fig. 6. Impact of neighborhood range in GCN-IED on nine common datasets.

To make a balance between homophilous and heterophilous graphs, we set $\lambda = \frac{1}{k+1/(\rho-0.3)}$ as the coefficient for edges of the kNN graph. The value of λ is calculated based on the k value of the kNN graph and the threshold ρ ($\rho > 0.3$). This formulation ensures that as the threshold value increases, the weight of the original graph decreases, and the weight of the edges in the kNN graph increases. Additionally, as the number of edges in the kNN graph increases, each edge weight decreases.

6.4.2. Analysis of neighborhood range

From Fig. 6, we observe that the performance of the aggregated features generally surpasses that of the original features (0-hop) in both homophilous and heterophilous graphs, indicating the effectiveness of updating the graph topology. Additionally, GCN-IED demonstrates the capability to effectively expand the range of neighborhood aggregation. Notably, the performance of GCN-IED stabilizes as the neighborhood range increases, reaching its peak accuracy within a 10-hop neighborhood. These results suggest that expanding the neighborhood on the updated graph does not introduce excessive noise or irrelevant information, and an appropriate neighborhood range contributes to improving

Table 4
The mean classification accuracy (%) of full-supervised node classification under 10 runs. The best results are highlighted in bold while underlined letters indicate the second best.

Method	Chameleon	Squirrel	Actor	Cornell	Wisconsin	Texas	Cora	Citeseer	Pubmed	AVG
GCN	59.98	38.75	27.16	61.35	49.41	55.95	86.96	76.40	87.03	60.33
GAT	55.75	35.24	27.18	60.54	50.00	56.22	87.67	76.09	85.48	59.35
GraphSAGE	62.24	44.25	34.1	62.24	77.25	71.89	86.88	76.72	88.71	67.14
JKNet	60.39	45.38	25.91	58.92	48.63	58.38	84.79	71.79	85.92	60.01
APPNP	54.39	35.11	26.53	58.65	45.69	58.92	87.36	75.29	86.64	58.73
GPRGNN	66.48	48.86	35.27	84.71	83.53	83.53	87.78	76.62	87.85	72.74
GCNII	58.99	38.87	33.76	72.43	72.75	71.62	88.21	76.96	89.38	67.00
GCNII*	63.20	41.46	34.88	77.03	81.18	76.22	88.01	76.94	**90.27**	69.91
Geom-GCN-I	60.37	33.14	29.10	56.75	58.63	57.57	85.25	**78.05**	89.99	60.98
Geom-GCN-P	60.92	38.09	31.65	59.45	64.51	68.38	84.83	75.40	88.08	63.48
Geom-GCN-S	60.19	36.14	30.32	55.67	56.86	60.26	85.27	74.90	84.70	60.48
H2GCN-1	52.96	36.42	34.31	79.46	83.14	83.24	86.21	77.11	89.43	69.14
H2GCN-2	58.38	32.33	34.49	78.11	84.31	80.00	87.93	77.06	89.55	69.13
dDGM	45.61	33.04	31.58	54.05	60.78	64.86	82.09	66.81	85.78	58.29
ACM-GCN	66.47	54.38	36.12	84.86	86.47	84.05	87.81	77.09	89.03	74.03
ACMII-GCN	66.82	51.72	**36.37**	81.89	85.49	**86.22**	87.87	77.06	88.59	73.56
MLP	45.33	28.98	34.20	79.19	85.49	81.08	75.49	73.23	86.17	65.46
GCN-IED	**73.25**	**65.07**	36.12	**86.76**	**86.67**	84.86	**88.39**	77.85	89.57	**76.50**
w/o GTU	56.21	40.06	31.86	62.16	72.16	57.84	88.15	77.32	89.50	63.92
w/o AMA	72.98	65.00	35.86	85.41	85.49	83.24	87.53	76.70	88.71	75.65

Table 5
Impact of local neighborhood range on performance.

Range	Chameleon	Squirrel	Actor	Cornell	Wisconsin	Texas	Cora	Citeseer	Pubmed
0-hop	43.9	29.17	36.12	86.76	86.67	84.86	87.16	76.46	89.45
1-hop	65.9	60.35	32.54	73.78	78.01	71.35	88.39	77.85	89.57
2-hop	73.25	65.07	31.69	71.08	75.69	67.03	88.01	77.34	89.51

the edge distribution and, consequently, the neighborhood distribution. However, the impact of the neighborhood range on different datasets varies; for instance, the Squirrel dataset is more sensitive to the neighborhood range, while the Citeseer dataset shows less sensitivity.

6.4.3. Comparison of homophily

We compare the homophily of the input graphs (with self-loops) to the mean homophily of the output graphs (with self-loops) after 10 runs. Fig. 7 illustrates the changes in homophily resulting from the graph topology updates in GCN-IED. While the impact is less pronounced in homophilous graphs, we observe a significant improvement in homophily across all datasets for heterophilous graphs. This finding suggests that leveraging local neighborhood information aids in identifying corresponding neighbors, thereby enhancing performance on heterophilous graphs. Conversely, the fluctuations in homophily help explain the limited improvements observed in homophilous graphs.

6.4.4. Impact of different hop local neighbors

We investigate the effect of different hop local neighbors on graph topology update, and the results are summarized in Table 5. On homophilous graphs, using 1-hop neighbors tends to yield better results. However, on heterophilous graphs such as Chameleon and Squirrel, the original edge distribution plays a positive role, and aggregating neighbor information improves the updating of graph topology. On other heterophilous graphs, expanding the range of aggregated neighbors has a negative impact, indicating that the node itself provides the best features.

These findings highlight the varying distinguishability of neighborhood distributions on different datasets. Some datasets, like Cornell, have easily distinguishable node features, and the edge distribution disrupts the aggregation of node features, making it challenging to find neighbors from the same class. This phenomenon is commonly observed in heterophilous graphs. Other datasets, like Chameleon, have less distinguishable node features, but a "good" edge distribution enhances the distinguishability of aggregated node features and facilitates the identification of neighbors from the same class.

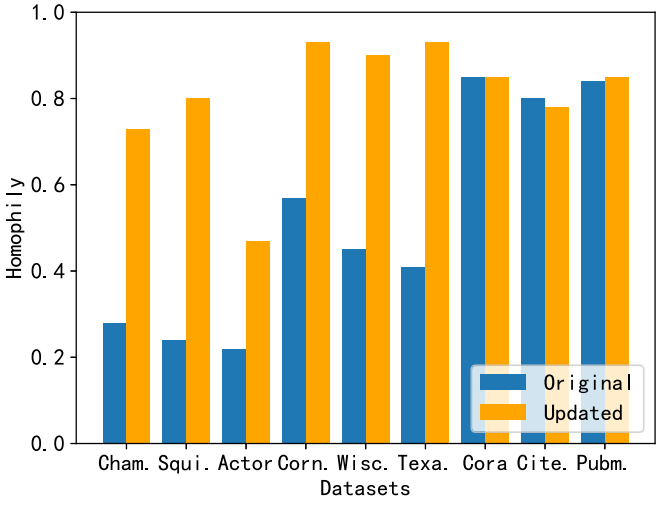


Fig. 7. The comparison of homophily before and after the graph topology update.

6.4.5. Sensitivity analysis of k values for kNN graphs

We conducted a sensitivity analysis of the k values in the kNN graphs on nine datasets, as shown in Fig. 8. The experimental results indicate that the k values have little effect on the majority of the datasets when $k \leq 5$. This suggests that the neighbors found on most of the datasets are reliable when k is not too large. However, on the Squirrel dataset, the best performance was observed when $k \leq 2$, after which the performance decreased significantly due to an increase in the number of unreliable neighbors.

6.5. Applicability of our model on large graph data

Our model, GCN-IED, has demonstrated successful performance in experimental evaluations on various datasets, particularly those involving heterogeneous graphs. To assess the effectiveness of GCN-IED on

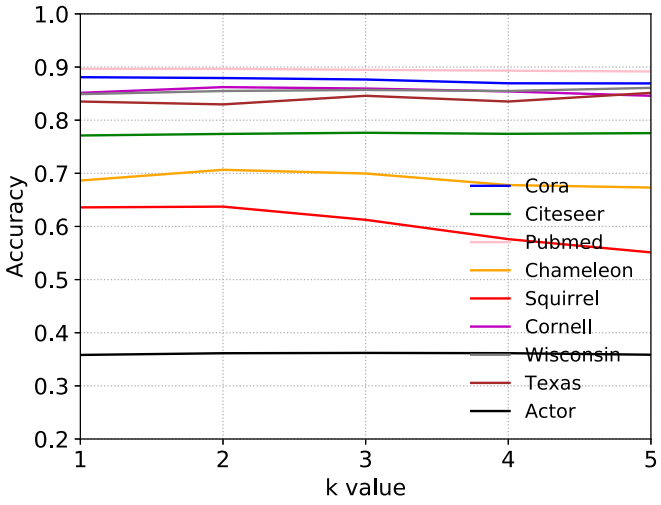


Fig. 8. Sensitivity analysis of k values for kNN graphs on nine datasets.

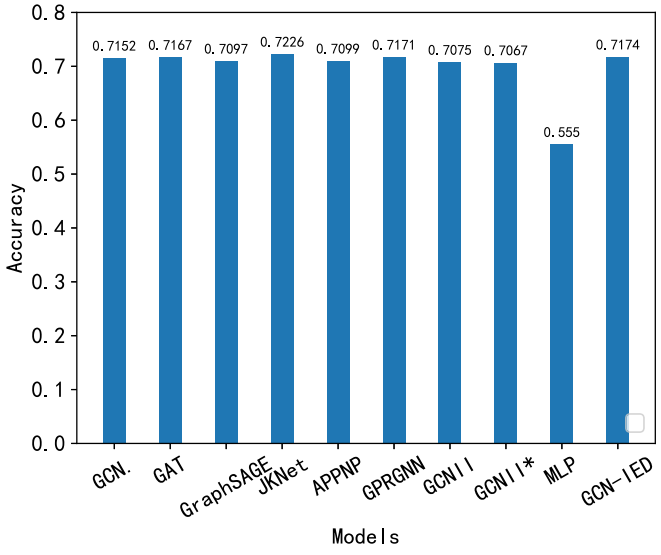


Fig. 9. Comparison of model performances using the OGB dataset.

large-scale datasets, we conducted tests using the ogbn-arxiv dataset from OGB (Hu et al., 2020). In our experimental setup, we utilized 256 hidden units and a dropout rate of 0.5 for all models. MLP, GCN, GAT, and GraphSAGE were configured with two layers, while JKNet utilized five layers. For APPNP, GCNII, GPRGNN, and GCN-IED, we set 10 propagation steps. Additionally, we established weight decay as 10^{-4}, a local neighborhood range of 1, a dropout rate of 0.6, a top-k edges value of 6, and a threshold of 0.5 for our model.

The outcomes of various models are displayed in Fig. 9. However, it is important to note that models specifically designed for heterogeneous graphs, such as Geom-GCN and H2GCN, are not included in Fig. 9. These models require the exploration of global neighbors, but their results could not be included in the figure due to memory limitations. It is worth emphasizing that our model also requires the exploration of neighbors. In order to overcome memory limitations during training, we implemented a sampling technique where 15,000 nodes were selected for global neighbor identification and aggregation at each epoch. This sampling strategy helps conserve memory but does result in a slight reduction in prediction accuracy. As a result, the performance of our model, as shown in Fig. 9, does not stand out significantly. However, our algorithm focuses on the impact of edge distribution and demonstrates significant improvements in heterogeneous graphs. Moving forward, we will explore different approaches to enhance our model's adaptability to large-scale graph datasets.

7. Conclusion

In conclusion, we have conducted a thorough investigation into the role of edge distribution in GCNs and its impact on network performance. Our theoretical and empirical analyses have revealed the significance of edge distribution in achieving optimal results in graph representation learning. To address this, we have proposed a novel framework called GCN-IED (Graph Convolution Network with Improved Edge Distribution) that enhances edge distribution through graph topology updating and extensible neighborhood aggregation. By incorporating both direct and hidden edges, our model achieves superior performance, particularly on heterophilous graphs. Extensive experiments on various datasets validate the effectiveness of GCN-IED in different scenarios.

Our study provides valuable insights into the improvement of graph convolutional networks by considering the influence of edge distribution. By optimizing the edge distribution, we contribute to the advancement of graph representation learning, especially for heterophilous graphs. The proposed GCN-IED framework paves the way for further research and advancements in graph-based machine learning. It holds great potential for applications in a wide range of fields, including social networks, biological networks, and recommendation systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by National Key Research and Development Program of China (No. 2021ZD0113303), the National Natural Science Foundation of China (Nos. 62022052, 62276159).

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Harutyunyan, H., Alipourfard, N., Lerman, K., et al. (2019). MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *International conference on machine learning* (pp. 21–29).

Andersen, R., Chung, F., & Lang, K. (2006). Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE symposium on foundations of computer science* (pp. 475–486). IEEE.

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in neural information processing systems* (pp. 1993–2001).

Brody, S., Alon, U., & Yahav, E. (2022). How attentive are graph attention networks? In *International conference on learning representations*.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral networks and deep locally connected networks on graphs. In *International conference on learning representations*.

Chamberlain, B., Rowbottom, J., Gorinova, M., Bronstein, M., Webb, S., & Rossi, E. (2021). GRAND: Graph neural diffusion. In *International conference on machine learning* (pp. 1407–1418).

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34, no. 04* (pp. 3438–3445).

Chen, Y., Wu, L., & Zaki, M. J. (2020). Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in neural information processing systems* (pp. 19314–19326).

Chien, E., Peng, J., Li, P., & Milenkovic, O. (2021). Adaptive universal generalized PageRank graph neural network. In *International conference on learning representations*.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Neural information processing systems* (pp. 3844–3852).

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., et al. (2020). Graph random neural networks for semi-supervised learning on graphs. In *Advances in neural information processing systems* (pp. 22092–22103).

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., & Pontil, M. (2018). Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning* (pp. 1568–1577). PMLR.

Franceschi, L., Niepert, M., Pontil, M., & He, X. (2019). Learning discrete structures for graph neural networks. In *International conference on machine learning* (pp. 1972–1982).

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263–1272).

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Neural information processing systems* (pp. 1025–1035).

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., et al. (2020). Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, *33*, 22118–22133.

Javaloy, A., Martin, P. S., Levi, A., & Valera, I. (2023). Learnable graph convolutional attention networks. In *International conference on learning representations*.

Jin, D., Yu, Z., Huo, C., Wang, R., Wang, X., He, D., et al. (2021). Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, *34*.

Kazi, A., Cosmo, L., Ahmadi, S.-A., Navab, N., & Bronstein, M. M. (2022). Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(2), 1606–1617.

Keriven, N. (2022). Not too little, not too much: a theoretical analysis of graph (over) smoothing. In *NeurIPS 2022-36th conference on neural information processing systems*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Klicpera, J., Bojchevski, A., & Günnemann, S. (2019). Predict then propagate: Graph neural networks meet personalized PageRank. In *International conference on learning representations*.

Kloster, K., & Gleich, D. F. (2014). Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1386–1395).

Kondor, R. I., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning, vol. 2002* (pp. 315–322).

Leman, A., & Weisfeiler, B. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsiya*, *2*(9), 12–16.

Li, Y., Zemel, R., Brockschmidt, M., & Tarlow, D. (2016). Gated graph sequence neural networks. In *International conference on learning representations*.

Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., et al. (2022). Revisiting heterophily for graph neural networks. arXiv preprint arXiv:2210.07606.

Luzhnica, E., Day, B., & Lio, P. (2019). Clique pooling for graph classification. arXiv preprint arXiv:1904.00374.

Ma, Y., Liu, X., Shah, N., & Tang, J. (2021). Is homophily a necessity for graph neural networks? arXiv preprint arXiv:2106.06134.

Ma, Y., Wang, S., Aggarwal, C. C., & Tang, J. (2019). Graph convolutional networks with eigenpooling. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 723–731).

McPherson, M., Smith-Lovin, L., & Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, *27*(1), 415–444.

Molaei, S., Bousejin, N. G., Zare, H., Jalili, M., & Pan, S. (2021). Learning graph representations with maximal cliques. *IEEE Transactions on Neural Networks and Learning Systems*.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., et al. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence, vol. 33, no. 01* (pp. 4602–4609).

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web*: *Technical report*, Stanford InfoLab.

Pandit, S., Chau, D. H., Wang, S., & Faloutsos, C. (2007). 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).

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., & Yang, B. (2020). Geom-GCN: Geometric graph convolutional networks. In *International conference on learning representations*.

Rusch, T. K., Bronstein, M. M., & Mishra, S. (2023). A survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *Ai Magazine*, *29*(3), 93–106.

Spinelli, I., Scardapane, S., Hussain, A., & Uncini, A. (2021). Fairdrop: Biased edge dropout for enhancing fairness in graph representation learning. *IEEE Transactions on Artificial Intelligence*, *3*(3), 344–354.

Sun, K., Lin, Z., & Zhu, Z. (2021). AdaGCN: Adaboosting graph convolutional networks into deep models. In *International conference on learning representations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. In *Neural information processing systems* (pp. 5998–6008).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In *International conference on learning representations*.

Wu, F., Zhang, T., de Souza, A. H., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In *International conference on machine learning* (pp. 6861–6871).

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks. In *International conference on learning representations*.

Xu, K., Li, C., Tian, Y., Sonobe, T., ichi Kawarabayashi, K., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning* (pp. 5449–5458).

Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning* (pp. 40–48).

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in Neural Information Processing Systems*, *31*.

You, J., Ying, R., & Leskovec, J. (2019). Position-aware graph neural networks. In *International conference on machine learning* (pp. 7134–7143). PMLR.

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems*, *31*.

Zhang, C., Song, D., Huang, C., Swami, A., & Chawla, N. V. (2019). Heterogeneous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 793–803).

Zhao, J., Dong, Y., Ding, M., Kharlamov, E., & Tang, J. (2021). Adaptive diffusion in graph neural networks. *Advances in Neural Information Processing Systems*, *34*.

Zhong, Z., Gonzalez, G., Grattarola, D., & Pang, J. (2022). Unsupervised network embedding beyond homophily. *Transactions on Machine Learning Research*.

Zhu, H., & Koniusz, P. (2021). Simple spectral graph convolution. In *International conference on learning representations*.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, *33*, 7793–7804.