On Which Nodes Does GCN Fail? Enhancing GCN From the Node Perspective

Jincheng Huang [1] **Jialie Shen** [2] **Xiaoshuang Shi** [1 3] **Xiaofeng Zhu** [1 4]

Abstract

The label smoothness assumption is at the core of Graph Convolutional Networks (GCNs): nodes in a local region have similar labels. Thus, GCN performs local feature smoothing operation to adhere to this assumption. However, there exist some nodes whose labels obtained by feature smoothing conflict with the label smoothness assumption. We find that the label smoothness assumption and the process of feature smoothing are both problematic on these nodes, and call these nodes out of GCN's control (OOC nodes). In this paper, first, we design the corresponding algorithm to locate the OOC nodes, then we summarize the characteristics of OOC nodes that affect their representation learning, and based on their characteristics, we present DaGCN, an efficient framework that can facilitate the OOC nodes. Extensive experiments verify the superiority of the proposed method and demonstrate that current advanced GCNs are improvements specifically on OOC nodes; the remaining nodes under GCN's control (UC nodes) are already optimally represented by vanilla GCN on most datasets.

1. Introduction

Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) exhibit a remarkable capacity to handle graph data. Therefore, in recent years, many researchers have drawn inspiration from GCNs (Du et al., 2022; Gasteiger et al., 2018; Chen et al., 2020; Feng et al., 2022; Liu et al., 2022a; Huang et al., 2023a; Yang et al., 2024), leading to the development of numerous advanced GCN models. These models not only perform well in the field of graph mining but have also achieved great success in different fields such as healthcare (Li et al., 2020b; Sun et al., 2020b), recommender systems (Fan et al., 2022; 2019), and natural language processing (Wu et al., 2023; Malekzadeh et al., 2021).

The success of GCN is primarily attributed to its smoothing assumption: **Connect nodes tend to have a high probability of sharing the same labels. (*i.e.,* label smoothness assumption)(Zhang et al., 2021)**. Most of the advanced GCNs are built upon this assumption to enhance the capabilities of GCN, which can be roughly divided into two categories: GCN-smoothing based methods and GCN-smoothing enhanced methods. The GCN-smoothing based methods utilize additional components to get a more robust and comprehensive representation based on GCN's smoothing operation. For example, Graph Contrastive Learning methods (Liu et al., 2022b) generally utilizes GCN-smoothing operation as the encoders to learn the invariance of the representation, AM-GCN (Wang et al., 2020) uses multiple GCN-smoothing operation for different channels to enhance the GCN's capability of fusing topological structures and node features substantially. The GCN-smoothing enhanced methods aim to improve the smoothing operation to ensure the node obtains richer and more appropriate information. For example, APPNP (Gasteiger et al., 2018), DAGNN (Liu et al., 2020), and GCNII (Chen et al., 2020) increase the GCN smoothing range while mitigating over-smoothing (Li et al., 2018) for each node through various residual techniques.

Although previous GCNs have achieved promising results by using feature smoothing operation to adhere to label smoothness assumption, a fundamental question remains unexplored: Do all nodes equally benefit from it? Specifically, GCN propagates features on the graph so that the features of connected nodes are similar (*i.e.,* feature smoothing operation). Following this, GCNs utilize known labels to guide the classification process. Given that connected nodes exhibit similar features post feature smoothing, it is implicitly assumed that nodes near those with known labels should share the same labels (*i.e.,* label smoothness assumption). Suppose the labels predicted through the feature smoothing operation align with the label smoothness assumption, such nodes meet the expectations of the GCN framework,

[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China [2]Department of Computer Science, City, University of London, London, United Kingdom [3]Sichuan Artificial Intelligence Research Institute, Yibin, China [4]Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China. Correspondence to: X. Zhu <seanzhuxf@gmail.com>, X. Shi <xsshi2013@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

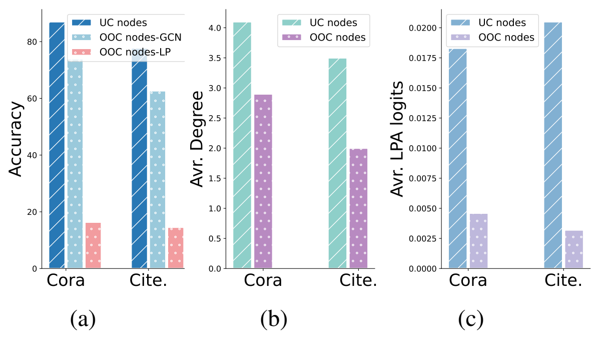


Figure 1. (a) The accuracy of GCN on UC and OOC nodes. (b) The average degree of UC and OOC nodes. (c) Average LPA output logits of UC and OOC nodes. Node with higher average LPA logits indicates closer to labeled nodes and vice versa.

indicating that they are suitable to be represented under the GCN framework and we call these nodes under the control of GCN (*i.e.,* UC nodes). On the other hand, if they cannot be aligned and conflict occurs, these nodes are out of the GCN's control (*i.e.,* OOC nodes). The GCN cannot guarantee that OOC nodes learn the desired representation compared to the UC nodes. Therefore, we consider OOC nodes to be the primary limitation of the GCN frameworks. To locate OOC nodes for further study, one challenge is that the labels under the label smoothness assumption are not clearly defined, so it is difficult to locate OOC nodes by comparing the labels predicted by feature smoothing and the labels under the smoothness assumption. Since GCN performs most effectively when the label distribution of unknown labels adheres to the label smoothness assumption. Theoretically, we found that the label distribution that maximizes GCN's effectiveness is equivalent to the distribution produced by the Label Propagation Algorithm (LPA (Zhu, 2005; Zhou et al., 2003)). Therefore, we can use the labels output by LPA to describe the label distribution under the label smoothness assumption for further analysis. We find that OOC nodes account for a fairly significant proportion of unlabeled nodes, e.g., 39.6%, 53.5%, and 29.2% in the Cora, Citeseer, and Pubmed datasets, respectively.

For OOC nodes, the conflict between the label smoothness assumption and feature smoothing operation makes it challenging for them to learn effective representations within the GCN framework (including the LPA algorithm). We can verify this by observing the results in Figure 1 (a). The accuracy of UC nodes is significantly higher than that of OOC nodes on GCN and LPA algorithms. Therefore, OOC nodes have more potential to be improved. Furthermore, we found that there are at least two reasons for the conflict on OOC nodes: (i) Nodes with few neighbors. Such nodes are less affected by message passing and the nodes lack sufficient neighbor information (*i.e.,* feature and label information) to describe the correct node representation. We further verified it on real-world datasets in Figure 1 (b). (ii) Nodes away from labeled nodes. As they lack known labeled neighbors, the label smoothness assumption cannot encompass these nodes. Additionally, the aggregation process becomes uncorrectable, leading to error accumulation during feature propagation (*i.e.,* feature smoothing) on OOC nodes. From Figure 1 (c), we validate that OOC nodes are typically distant from label nodes.

In this paper, to boost the OOC nodes by addressing the above issues, we propose a simple **D**ual **a**ugmented **GCN** (**DaGCN**). Specifically, we employ the Conditional Variational Auto-encoder to generate virtual neighbors for every node. Simultaneously, we propose the Dual Similarity K-Nearest Neighbor (DSKNN) module to identify potential real neighbors in the original graph. This approach enriches the node's neighbor information from two perspectives, thus tackling the issue (i). Notably, the DSKNN module considers both the original features and the generated virtual neighbor features. To address the issue of being distant from the labeled nodes, the DSKNN module provides a new propagation path, and we show that this problem can be solved as long as the DSKNN is set to a large k value, to tackle the issue (ii). As a result, our method can optimize the representation of OOC nodes by supplementing rich neighbor information for them and increasing the probability of the OOC nodes being closer to labeled nodes. The contributions of this paper are summarized as follows:

- We first investigate the conflicts between GCN's operation and its assumption, then design the corresponding algorithm to locate the OOC nodes. Moreover, we provide a new perspective that can help later work in analyzing and enhancing the performance of GCNs.
- We propose the DaGCN model that can simultaneously solve the two problems existing in OOC nodes, and improve the generalization ability and robustness of the GCN model.
- We empirically verified that our proposed model achieves favorable results on the seven nodes classification benchmark dataset and further validates the robustness and generalization of the model.

2. Related Work

This section briefly reviews the topics related to this work, including Graph Convolutional Networks, Neighbor Generation Methods, and Graph Rewiring Methods.

2.1. Graph Convolutional Networks

The early graph convolution networks mainly focus on the filter design of the eigenvalues of the Laplacian matrix (Bruna et al., 2014). (Defferrard et al., 2016) used Chebyshev polynomials to fit the filter shape, enabling fast convolution of spectral neural networks. Then GCN (Kipf & Welling, 2017) truncates the first two Chebyshev polynomials to further simplify the entire operation and bring improved performance. SGC (Wu et al., 2019a) further decouples the aggregation and mapping process of GCN, which has fewer parameters and training speed while maintaining performance with GCN. APPNP (Gasteiger et al., 2018) decouples prediction and propagation by performing personalized propagation of neural predictions. To make the GCN deeper while avoiding the oversmoothing problem, the GCNII (Chen et al., 2020) utilized the residual technical and self-mapping parameter matrix to design the deeper and more powerful GCN. ScatterGCN (Min et al., 2020) combines low-pass and band-pass filters to make GCN not only consider the similarity but also improve the discrimination of GCN. Graph Gaussian Convolution Networks (G^{2}CN) (Li et al., 2022) develop a new framework *i.e.,* concentration analysis, propose a linear feature smooth method with flexible concentration properties. These methods stacking graph layers to enlarge the receptive field can incorporate multi-hop neighboring information but lead to over-smoothing and over-squashing problems.

2.2. Neighbor Generation Methods

Since feature smooth benefits from the rich feature information in the graph data, thus many works (Zhu et al., 2021; Song et al., 2021; Yang et al., 2023b) naturally utilize structure information to generate more feature information. For example, GRAND (Feng et al., 2020) random drop nodes multiple times and uses feature smooth to generate the new feature of dropped nodes. For generation methods, commonly used in deep learning are based on auto-encoders, and here some methods extend it to graphs (Liu et al., 2022a; Grover et al., 2019; Hou et al., 2022), which are based on variational auto-encoder to generate the neighbor information, they directly use this generation information or combine original information for the downstream task. Beyond that, other types of methods GraphSMOTE (Zhao et al., 2021) and GraphMIXUP (Wu et al., 2021) generate neighbors by SMOTE and mixup techniques and add them directly to the original graph. All of the above methods generate virtual neighbor features and none of them consider mining potential real neighbors from the original graph.

2.3. Graph Rewiring Methods

Graph rewiring methods can discover and connect the potential neighbors from the real existing nodes while disconnecting irrelevant neighbors to get a new graph structure, thus applying the GCN on the proposed DSKNN graph in this paper can be seen as a graph rewiring method. We can divide graph rewiring methods into two categories: the first category of rewiring methods such as SDRF (Topping et al., 2022), SJLR (Giraldo et al., 2023) and BORF (Nguyen et al., 2023) aims to enhance the curvature of the neighborhood by rewiring connecting edges with small curvature. They increase local connectivity in the graph topology indirectly expanding the influence range of labels. However, these methods focus solely on the graph structure, neglecting the utilization of node features and neighboring features. Moreover, they cannot generate connections between distant and disconnected nodes. The other is structure learning methods, such as DHGR (Bi et al., 2022), PTDNet (Luo et al., 2021), SA-SGC (Huang et al., 2023b), BAGCN (Zhang et al., 2024) and Pro-GNN (Jin et al., 2020), these methods can adapt learning the robust and downstream task optimal graph structure. However, these methods involve end-to-end learning of graph structures, resulting in significant computational overhead and failing to address the challenge posed by nodes distant from labeled nodes.

3. Preliminaries

In this section, we first define the common notation, and then introduce the GCN and Label Propagation Algorithm, both of which will be used in subsequent discussions.

3.1. Problem Formulation

Given a graph $G = (V, E)$, where V is the node set and E is the edge set. Nodes are described by a feature matrix $\mathbf{X} \in \mathbb{R}^{n\times f}$ where $n = |V|$ is the number of nodes and f is the number of features for each node. One node is associated with a class label that is depicted in the label matrix $\mathbf{Y} \in \mathbb{R}^{n\times c}$ with a total of c classes. In this paper, we consider an undirected graph, whose adjacency matrix is represented with a sparse matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$. Let $\mathbf{D} = diag(d_1, d_2, \cdots, d_n)$ be the degree matrix, where $d_i = \sum_{j\in N_i} a_{ij}$ is the degree of node i. The symmetric normalized adjacency matrix is represented as $\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-\frac{1}{2}}\widetilde{\mathbf{A}}\widetilde{\mathbf{D}}^{-\frac{1}{2}}$ where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\mathbf{I}$ is the identity matrix and $\widetilde{\mathbf{D}}$ is the degree matrix of $\widetilde{\mathbf{A}}$. The row normalized adjacency matrix is represented as $\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}$. In this paper, we aim to solve graph-based semi-supervised node classification. Here, we have labels for only a very small fraction of the nodes, and the remainder we call the unknown label set. Our goal is to predict some of the labels in the unknown label set.

3.2. Graph Convolutional Network(GCN)

GCN (Kipf & Welling, 2017) is the most widely used and representative graph neural network, which uses a symmetric normalized adjacency matrix to aggregate information about neighbors. The GCN at layer l can be written as:

$$\mathbf{H}^{(l+1)} = \sigma(\widehat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \tag{1}$$

where σ is the activate function here is ReLU, $\mathbf{H}^{(0)} = \mathbf{X}$ and $\mathbf{W}^{(l)}$ is learnable parameters. By iterating L-layer convolutions, the GCN will make the representations of connected nodes closer and closer (Zhao & Akoglu, 2020; Rong et al., 2019), thus satisfying the "Connect nodes tend to have a high probability of sharing the same labels" assumption.

3.3. Label Propagation Algorithm

Label propagation algorithm(LPA) (Zhu, 2005; Zhou et al., 2003) is a classic semi-supervised learning algorithm that propagates known labels along the graph to unlabeled nodes. It can be formulated as follows:

$$\mathbf{Y}^{(l+1)} = \widehat{\mathbf{A}}\mathbf{Y}^{(l)}, \quad l = 1, 2, \cdots, L \tag{2}$$

where $\mathbf{Y}^{(0)} = \mathbf{Y}$ and $\mathbf{Y} \in \mathbb{R}^{n \times c}$ consists of one-hot label indicator vectors for labeled nodes or zero vectors for unlabeled nodes. In addition, various propagation schemes can be adopted for LP, such as commonly used methods like Personalized PageRank where $\mathbf{Y}^{(l+1)} = (1 - \alpha)\widehat{\mathbf{A}}\mathbf{Y}^{(l)} + \alpha\mathbf{Y}^{(l)}$.

4. Method

4.1. Label-Feature Smoothing Alignment

In this section, we motivate the need to study the nodes that conflict between feature smoothing operation and label smoothness assumption. To do that, we introduce the Complexity Measure to help us understand the GCN-expected label distribution on an unknown label set (*i.e.,* labels under GCN label smoothness assumption). It is the current mainstream method to measure the generalization ability of the model (Neyshabur et al., 2017), which describes the **a lower complexity measure means a better generalization ability**. We follow (Natekar & Sharma, 2020) to adopt Consistency of Representations as our Complexity Measure, which is designed based on the Davies-Bouldin Index (Davies & Bouldin, 1979). Formally, for a given dataset and a given layer of a model, the Davies-Bouldin Index can be written as follows:

$$S_a = \left(\frac{1}{n_a}\sum^{n_a}\left|O_a^{(i)} - \mu_{O_a}\right|^p\right)^{1/p} \quad \text{for } a = 1 \cdots k \tag{3}$$

$$M_{a,b} = \|\mu_{O_a} - \mu_{O_b}\|_p \quad \text{for } a, b = 1 \cdots k, \tag{4}$$

where a, b are two different classes, $O_a^{(i)}$ is the GCN smoothed feature of node i belonging to class a, μ_{O_a} is the cluster centroid of the representations of class a, here we set $p = 2$, thus S_a measures the intra-class distance of class a and $M_{a,b}$ is a measure of inter-class distance between class a and b. Then, we can define complexity measure based on the Davies-Bouldin Index as follows:

$$C = \frac{1}{k}\sum_{i=0}^{k-1}\max_{a \neq b}\frac{S_a + S_b}{M_{a,b}}. \tag{5}$$

With the above basis, we give the following theory.

Theorem 4.1. *For nodes with unknown labels in the graph, the upper bound of the GCN's generalization ability reaches optimal if the true labels of these nodes are equal to the labels generated by the LPA.*

The proof is in Appendix A.2. From the theorem 4.1, we found that the label distribution outputted by the LPA is equivalent to the label distribution that makes GCN most effective in the unknown label set. This is particularly significant since GCN performs most effectively when the labels of the unknown label set adhere to the label smoothness assumption. Thus, we can replace the labels under the label smoothness assumption with those outputted by the LPA to locate the OOC nodes. Consequently, if the labels predicted by feature smoothing are consistent with the labels generated by LPA, these nodes are under GCN's control (*i.e.,* UC nodes). Otherwise, it indicates that on these nodes, the GCN label smoothness assumption conflicts with the feature smoothing operation. Since these nodes operate within the GCN feature smoothing but do not adhere to the label smoothness assumption, we call these nodes out of the GCN's control (*i.e.,* OOC nodes).

Therefore, we propose the Label-Feature Smoothing Alignment Algorithm to find the OOC nodes. To better align the LPA, we rewrite the GCN feature smoothing in the following form:

$$\mathbf{Y}_{fs} = \widehat{\mathbf{A}}^L MLP(\mathbf{X}), \tag{6}$$

where L is the number of the layers and MLP($\cdot$) is Multi-Layer Perceptron. Note that here the MLP has been trained in advance, so each dimension of MLP($\mathbf{X}$)$\in \mathbb{R}^{n \times c}$ is the probability of each class. This feature smoothing form not only aligns with the LPA form but also brings faster training/inference speed and better generalization ability (Yang et al., 2023a). The LPA is then of the following form:

$$\mathbf{Y}_{lp} = \widehat{\mathbf{A}}^L\mathbf{Y}, \tag{7}$$

Then in this paper, the OOC nodes can be defined as:

$$\mathbf{V}_{OOC} = \{\mathbf{V}_i | argmax(\mathbf{Y}_{fs,i}) \neq argmax(\mathbf{Y}_{lp,i}), i \in [n]\}. \tag{8}$$

The UC nodes should be $\mathbf{V}_{UC} = \mathbf{V} - \mathbf{V}_{OOC}$.

As can be seen in Figure 1 (a), the accuracy of $\mathbf{V}_{UC}$ exceeds that of $\mathbf{V}_{OOC}$ by a significant margin. Moreover, it is known from Theory 4.1 that the upper bound of GCN's generalization ability on $\mathbf{V}_{UC}$ nodes is more likely to reach the optimal compared to $\mathbf{V}_{OOC}$ nodes, because the label distribution of UC nodes is closer to the output of LPA. Therefore, $\mathbf{V}_{OOC}$ nodes have more room for improvement. In this paper, we analyze $\mathbf{V}_{OOC}$ mainly and find that some properties of these nodes harm graph representation learning. Specifically, these properties include being distant from labeled nodes and having scarce neighbors. To address these problems, we propose Dual-augmented GCN(DaGCN) to solve them.

4.2. Dual augmented GCN

In this section, to overcome the problem of nodes with few neighbors, we propose two types of neighbor enhancement: virtual neighbor generation and potential real neighbor identification (*i.e.,* DSKNN). Meanwhile, for the problem of being far away from the labeled node, the DSKNN module provides a new message passing path that can increase the probability that the labeled node can influence the OOC nodes.

4.2.1. Virtual Neighbor Generation

To address the issue of nodes having a limited number of neighbors, we can utilize generative methods to create additional local neighbor information, thereby enriching the nodes with more comprehensive neighbor information. Thus, we employ the conditional variational auto-encoder (Kingma & Welling, 2011; Sohn et al., 2015) to condition on the node's own features to learn the distribution of its neighbor's features. Following (Liu et al., 2022a), we use $\mathbf{X}_v(v \in \mathbf{V})$ as a condition, and to learn the distribution of $\mathbf{X}_u(u \in N_v)$. The latent variable $\mathbf{z}$ is generated from the prior distribution $p(\mathbf{z}|\mathbf{X}_v)$ and the output $\mathbf{X}_u$ is generated from the distribution $p(\mathbf{X}_u|\mathbf{X}_v, \mathbf{z})$, which objective function is,

$$L_{ELBO} = -KL(q(\mathbf{z}|\mathbf{X}_u\mathbf{X}_v)||p(\mathbf{z}|\mathbf{X}_v)) + \mathrm{E}_{q(\mathbf{z}|\mathbf{X}_u,\mathbf{X}_v)}(p(\mathbf{X}_u|\mathbf{X}_v, \mathbf{z})). \tag{9}$$

The proof for the objective function is in Appendix A.1. The first term makes the $q(\cdot)$ distribution as close as possible to the $p(\cdot)$ distribution. The second term is the reconstruction loss, which makes the generated samples as close as possible to the true samples. The Conditional Variational Auto-encoder is composed of multiple MLPs.

According to the above two goals, the actual loss function is designed as:

$$L = -\frac{1}{2}(1 + \log \sigma^2 - \sigma^2 - \mu^2) + MSE(\mathbf{X}_u, q(\mathbf{X}_u|p(\mathbf{z}|\mathbf{X}_u, \mathbf{X}_v))), \tag{10}$$

where MSE is mean squared error, $p(\cdot)$ is a standard normal distribution, its μ is 0 and σ is 1, and there μ and σ is belong to $q(\cdot)$ learned by MLPs. Thus, we can utilize the node feature $\mathbf{X}_v$ as the condition and sample a latent variable $\mathbf{z} \sim N(0, 1)$ as input for the decoder. This process allows us to obtain the node v's virtual neighbor feature vector $\overline{\mathbf{X}}_v$.

4.2.2. Potential Real Neighbor Identification

The generated virtual neighbors are based on the nodes' neighbor distribution. However, it cannot generate neighbor information beyond first-order neighbor distribution, such as high-order neighbors and influential nodes. Additionally, virtual neighbors cannot enhance the impact of message passing on OOC nodes. Therefore, beyond virtual neighbors, we posit the existence of potential non-directly connected neighbors in the graph, which can provide the OOC nodes with additional necessary information and augment the message passing of OOC nodes. Specifically, previous studies (Chapelle et al., 2002) show that (i) nodes in the same subspace may be potential neighbors, and (ii) nodes whose neighbors are in the same subspace may be potential neighbors. Thus, we have the following objective function:

$$\min_{\mathbf{S}} \sum_{i,j=0}^{n} (-s_{i,j}\mathbf{X}_i^T\mathbf{X}_j - s_{i,j}\overline{\mathbf{X}}_i^T\overline{\mathbf{X}}_j + s_{i,j}^2). \tag{11}$$

In the above equation, the first and second terms encourage nodes with similar features and neighbor features to be assigned greater weights, the second term avoids the trivial solution. We can easily obtain the closed-form solution $s_{i,j}$ as follows:

$$s_{i,j} = \frac{1}{2}(\mathbf{X}_i^T\mathbf{X}_j + \overline{\mathbf{X}}_i^T\overline{\mathbf{X}}_j) = \frac{1}{2}(\mathbf{X}_i \| \overline{\mathbf{X}}_i)^T(\mathbf{X}_j \| \overline{\mathbf{X}}_j), \tag{12}$$

where $\|$ is concatenate operation. Through Equation 12, we can identify the potential neighbors of any node in the graph. To ensure the sparsity of $\mathbf{S}$, we only consider the k neighbors with the largest weights as potential neighbors. Thus we refer to $\mathbf{S}$ as the Dual Similarity K-Nearest Neighbor (DSKNN) graph adjacency matrix, and denote the symmetric normalized adjacency matrix of the DSKNN graph as $\widehat{\mathbf{S}}$. The construction of DSKNN connects nodes with similar features and neighboring features, making the features of connected nodes in the DSKNN graph highly similar. As a result, GCN (i.e., feature smoothing) is more likely to classify nodes and their neighbors into the same category compared to the original graph. This result is consistent with the results of LPA(i.e., label smoothness) on the DSKNN graph. Hence, it can reduce the probability of OOC nodes appearing and we verified it experimentally in Appendix B.3. Next, we discuss how to use DSKNN-graph to overcome the problem of OOC nodes being far away from labels.

Analysis of the nodes which distance from labels. As the two-layer GCN makes predictions based on second-order neighbors of nodes, the classification loss does not influence

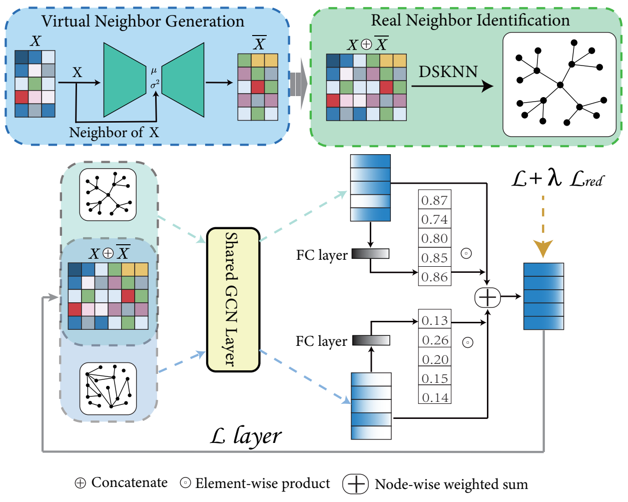


Figure 2. The flowchart of DaGCN. First, generating the virtual neighbor for every node by neighbor distribution condition on themselves. Then, through the original feature matrix and the virtual feature matrix select the top k similar neighbors (*i.e.,* DSKNN graph). Next, we concatenate the virtual features with the original features as input, the model conducts GCN feature smoothing operation on the original adjacency matrix and DSKNN graph adjacency matrix, and afterward, using the adaptive node-level assembling to fuse the two views' information including the output logits layer. Finally, we apply the entropy reduction loss *i.e.,* $\mathcal{L}_{red}$, to reduce the entropy increase caused by fusing the two output logits.

the features of nodes beyond two orders away from the labeled nodes, leaving these nodes unsupervised. Although these unsupervised nodes may not affect the training loss, most nodes predicted during testing rely on them. If their representations are learned without sufficient supervision, erroneous information may accumulate and propagate through GCN's feature smoothing operation, potentially leading to poor predictions at the test stage. This is one of the reasons why the labels obtained by feature smoothing and LPA on OOC nodes cannot be aligned. Intuitively, reducing the number of OOC nodes can be achieved by decreasing the occurrence probability of nodes that are not influenced by labels. We proved the occurrence probability of such nodes in the following Theory.

Theorem 4.2. *Given an undirected graph $G(V, E)$ has n nodes and e edges. Assuming there are q nodes in the graph with labels selected uniformly at random. The occurrence probability of nodes that not affected by labels with a two-layer GCN is equal to* $(1-\frac{q}{n})(1-\frac{q}{n-1})\prod_{i=1}^{q}(1-\frac{2m}{n(n-1)-2i})\prod_{i=q}^{2q}(1-\frac{2(m-1)}{n(n-1)-2i})$

The proof is in the Appendix A.3. From the results of Theory 4.2, it is found that the occurrence probability of unaffected by labeled nodes is negatively correlated with the number of labels and total edges. However, in the semi-supervised case, the exceptionally low number of labeled nodes, combined with the generally sparse graph structure (*i.e.,* low number of edges), leads to the widespread presence of these nodes. Since the number of labels is fixed, we can consider solving the problem by increasing the number of edges. Since the DSKNN graph can ensure better consistency between feature smoothing and label smoothing, while also allowing flexible addition or removal of edges, we just need to make sure that the value of k in constructing the DSKNN graph is much larger than the average degree of the original graph.

4.2.3. Overall Architecture

In our previous analysis, we designed dedicated components for the characteristics of OOC nodes, which logically should be abstracted OOC nodes out and handled separately. However, in our experiments 5.2, we observed that these components do not have a significant impact on UC nodes. Therefore, for implementation simplicity, we applied these components uniformly across all nodes. The flowchart is shown in Figure 2.

Since the generated virtual neighbors do not actually exist in the graph, integrating them into the node aggregation process is complex. Therefore, we directly fuse their information with the corresponding nodes using the concatenation operation,

$$\overline{X} = \mathbf{X} \oplus \overline{\mathbf{X}}. \tag{13}$$

Then, we use GCN to aggregate potential real neighbors on the DSKNN graph, which not only expands the neighbor information of the nodes at the same time reduces the average distance from the OOC nodes to the labeled nodes. Simultaneously, to preserve the topological information of the original graph, we apply the GCN layer on the original graph. Here, to reduce the spatial complexity of the model, we employ the shared parameters GCN layer both on the DSKNN-graph view and the original graph view.

$$\mathbf{H}^{(l)}_{ori} = \hat{\mathbf{A}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)}, \mathbf{H}^{(l)}_{ds} = \hat{\mathbf{S}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)}, \tag{14}$$

where $\mathbf{H}^{(l)}_{ori}$, $\mathbf{H}^{(l)}_{ds}$ are the representations after convolution of the origin graph and DSKNN graph (*i.e.,* two views), respectively, and $\mathbf{H}^{(0)} = \overline{X}$.

Naturally, the information from $\mathbf{H}^{(l)}_{ori}$ and $\mathbf{H}^{(l)}_{ds}$ needs to be fused here. We analyzed common fusion methods, which may be problematic (e.g., concatenate, sum, etc.). Specifically, each node's dependency on the two views is different. Notably, OOC nodes may exhibit a greater dependency on the DSKNN views compared to UC nodes overall. Moreover, the fusion method should vary for each layer because each layer focuses on different information, resulting in nodes in different layers having different dependencies on the two views. To solve the above problems, we design the **adaptive node-level assembling** as follows,

$$\mathbf{H}^{(l)} = diag(\ _0^{(l)})\mathbf{H}_{ori}^{(l)} + diag(\ _1^{(l)})\mathbf{H}_{ds}^{(l)}, \quad _0^{(l)} + \ _1^{(l)} = \mathbf{1}, \tag{15}$$

where $_0^{(l)}$, $_1^{(l)}$ 2 R^n and $diag(\ _0^{(l)})$ represents a diagonal matrix consisting of element of vector $_0^{(l)}$ as its diagonal elements at layer l. The corresponding weights $_0^{(l)}$, $_1^{(l)}$ denote the node-level weights for different views They are computed by two fully connected layers (*i.e.,* $FC_0^{(l)}, FC_1^{(l)}$) with the input being the respective two views node embeddings as follows:

$$_0^{(l)} = \sigma(FC_0^{(l)}(\mathbf{H}_{ori}^{(l)})), \quad _1^{(l)} = \sigma(FC_1^{(l)}(\mathbf{H}_{ds}^{(l)}))$$
$$[\ _0^{(l)}, \ _1^{(l)}] = \frac{[\ _0^{(l)}, \ _1^{(l)}]}{max(\left\|[\ _0^{(l)}, \ _1^{(l)}]\right\|_2, \epsilon)}, \tag{16}$$

where σ is sigmoid activate function and ϵ is a small value to avoid division by zero. In adaptive node-level assembling, the contribution of the representation of two views to the next layer's representation is determined by the node embedding generated by the corresponding view. That is to say, the nodes themselves decide which view is more reliable for downstream tasks. Note that we use adaptive node-level assembling on each layer including the output layer.

4.2.4. Objective function

While assembling logits at the output layer is an intuitively feasible approach, this approach is problematic in semi-supervised learning. A fundamental assumption of semi-supervised learning is that the decision boundary of the classifier should not pass through a high-density region of the marginal data distribution (Berthelot et al., 2019). One way to accomplish this is to require the classifier to output low-entropy predictions on unlabeled data (Grandvalet & Bengio, 2004). When the output layer uses the logits assembling, according to the concavity of entropy (see Lemma 4.3) the entropy will increase beyond a linear combination of the two view output logits' entropy values and it is easy to know that the entropy after the assembly of logits is greater than the entropy of at least one view output logits, we proof it in Appendix A.4. This is obviously moving in the opposite direction of the basic assumptions of semi-supervised learning mentioned above.

Lemma 4.3. ***(concavity of entropy)*** *Given two probability distributions p_1, p_2, with probability $\ 2\ [0,1]$, the following inequality holds.*

$$H(\ p_1 + (1 \quad)p_2) \qquad H(p_1) + (1 \quad)H(p_2) \tag{17}$$

where $H(p_1)$ and $H(p_2)$ denote entropy of p_1 and p_2.

The proof please see the (Ash, 2012). In addition, the assembly of the two view's outputs can increase accuracy but decrease confidence (*i.e.,* high entropy corresponding to low confidence), which, coupled with the general under-confidence exhibited by GNNs (Wang et al., 2021), further undermines the explainability and the credibility of GCNs' prediction results.

To mitigate the issues mentioned above, we propose a simple entropy reduction constraint as follows:

$$L_{red} = \frac{1}{c}\sum_{i=1}^{c}(\mathbf{y}_i \quad \mathbf{y}_i^{\frac{1}{\tau}} \Big/ \sum_j^c \mathbf{y}_j^{\frac{1}{\tau}})^2 + \mathsf{I}(jj\mathbf{y}_{ori} \quad \mathbf{y}_{ds}jj_2), \tag{18}$$

where τ 2 $[0,1]$ is a hyper-parameter, functioning as a slack variable. The lower the value τ is, the lesser the entropy of $\mathbf{y}$ becomes, when the τ $!$ 0, the $\mathbf{y}_i^{\frac{1}{\tau}} \Big/ \sum_j^c \mathbf{y}_j^{\frac{1}{\tau}}$ term will approach a Dirac (*i.e.,* one-hot) distribution and in this time the entropy is minimized, and $\mathbf{y}_{ori}$, $\mathbf{y}_{ds}$, $\mathbf{y}$ denote the outputs of origin graph view, DSKNN-graph view, and whole model(*i.e.,* $\mathbf{y} = \ _0\mathbf{y}_{ori} + \ _1\mathbf{y}_{ds}$), respectively. The second term acts as a counterbalance to the entropy increase by augmenting the similarity between the outputs produced from the two distinct views, where the $\mathsf{I}(\ )$ is the selection function means that we can choose $\mathsf{I}(x) = x$ or $\mathsf{I}(x) = 0$ to control whether to use the second term on different datasets.

Thus, the total objective function of DaGCN is as follows:

$$L = L_{sup} + \beta L_{red}, \tag{19}$$

where L_{sup} is the cross-entropy loss function and β is a hyper-parameter that controls the balance between the two losses. We analyze the model complexity and real running time in Appendix C.1.

5. Empirically Studies

5.1. Experimental Setting

Datasets. We evaluate the effectiveness of the proposed method on seven datasets based on citation, co-authorship, and co-purchase graphs for semi-supervised node classification tasks; including Cora, Citeseer, Pubmed (Sen et al., 2008), Coauthor CS, Coauthor Physics, Amazon Computers and Amazon Photo (Shchur et al., 2018). The statistics and details of these datasets are in Appendix B.1.

Baseline and Setting. The comparison methods include two traditional GNN methods(*i.e.,* GCN (Kipf & Welling, 2017), GAT (Velickovic et al., 2018)), four state-of-the-art GCN promotion methods (*i.e.,* APPNP (Gasteiger et al., 2018), GCN-LPA (Wang & Leskovec, 2021), DAGNN (Liu et al., 2020), AERO-GNN (Lee et al., 2023), wGNN (Ji et al., 2023)). Since wGNN is a plug-and-play approach, we compare its performance on GCN. The details of these baselines are in the Appendix B.4. For all baselines, we use the best parameter settings for the corresponding paper.

Table 1. Overall classification accuracy (%). The best results are in blod.

Datasets	Cora	Citeseer	Pubmed	Computers	Photo	Physics	CS
GCN	81.5 0.82	70.9 0.71	79.0 0.52	82.6 2.43	91.2 1.21	92.8 1.00	91.1 0.52
GAT	83.0 0.41	71.1 0.51	79.1 0.44	78.0 19.0	85.7 20.3	92.5 0.94	90.5 0.61
APPNP	83.3 0.51	72.5 0.62	79.9 0.32	82.2 2.13	90.8 1.32	93.7 0.69	92.5 0.32
GCN-LPA	83.1 0.73	72.6 0.80	78.6 1.32	83.5 1.41	91.1 0.83	93.6 1.06	91.8 0.42
DAGNN	84.4 0.57	73.3 0.65	80.5 0.53	83.5 1.28	92.0 1.22	94.0 0.62	91.5 0.33
wGCN	83.1 0.31	73.9 0.46	80.8 0.25	83.6 0.86	92.4 0.18	92.8 0.23	89.3 0.14
AERO-GNN	83.9 0.51	73.2 0.68	80.6 0.55	83.3 0.72	91.1 0.83	93.3 0.65	92.0 0.71
Ours	**84.8 0.53**	**75.3 0.41**	**81.7 0.88**	**84.0 1.25**	**92.9 0.56**	**94.3 0.25**	**93.4 0.18**

Table 2. The UC nodes and OOC nodes classification accuracy (%) in test set. The best results of OOC nodes are in bold.

Datasets	Cora		Citeseer		Pubmed		Computers		Photo		Physics		CS	
	UC nodes	OOC nodes	UC nodes	OOC nodes	UC nodes	OOC nodes	UC nodes	OOC nodes	UC nodes	OOC nodes	UC nodes	OOC nodes	UC nodes	OOC nodes
GCN	87.01 0.6	73.95 1.1	77.75 0.5	62.66 0.8	83.66 0.3	67.05 1.0	87.41 0.5	70.13 0.8	96.58 0.7	78.02 1.4	97.01 0.2	86.45 0.3	95.65 0.4	84.53 0.6
APPNP	87.47 0.5	76.21 1.3	78.21 0.6	67.59 0.9	84.36 0.5	67.96 1.1	87.23 0.9	69.84 2.6	95.98 0.8	78.13 1.5	97.13 0.5	89.25 0.9	95.31 0.2	87.01 0.5
DAGNN	87.80 0.5	78.52 1.5	78.33 0.7	68.27 0.93	84.48 0.8	68.32 0.7	88.21 0.7	71.97 1.5	95.36 0.8	80.64 1.2	97.16 0.5	89.98 0.7	94.75 0.2	87.53 0.7
AERO-GNN	87.74 0.3	77.38 0.8	78.14 0.8	68.78 1.0	85.38 0.3	69.79 1.1	88.56 0.8	71.72 1.3	96.34 0.6	77.65 1.0	97.03 0.4	88.65 0.9	95.89 0.6	86.01 1.1
Ours	87.70 0.5	**79.26 0.7**	78.39 0.5	**72.04 1.5**	85.54 0.3	**73.16 1.0**	88.12 0.8	**73.39 1.5**	95.57 0.5	**82.75 0.8**	97.12 0.2	**91.15 0.3**	95.53 0.1	**89.51 0.3**

We conduct all experiments on a server with Nvidia RTX 4090 (24GB memory each) and conduct each experiment on ten random seeds and report the average results. In the proposed method, we optimize all parameters by Adam optimization (Kingma & Ba, 2015) with the learning rate 0.01 except for Pubmed is 0.2, and set the weight decay as 0.0005 for all datasets. Moreover, we set the number of model layers in the range of f2,3,4g, set the dropout in the range of f0.5, 0.6g, and set the size of the hidden unit in the range of f4, 8, 16, 62, 64g. We set τ in from 0 to 1 at intervals of 0.1 and β in from 0 to 1.5 at intervals of 0.1. The best hyperparameters for each dataset can be found in Appendix B.5. To evaluate the effectiveness of the proposed method, We follow the evaluation protocol and split of (Kipf & Welling, 2017) on the Citation Network dataset (*i.e.,* 20 per class for train, 500 nodes for valid, 1000 nodes for test) and follow (Shchur et al., 2018; Liu et al., 2020) on the co-authorship and co-purchase datasets(*i.e.,* 20 per class for train, 30 per class for valid, remain nodes for test).

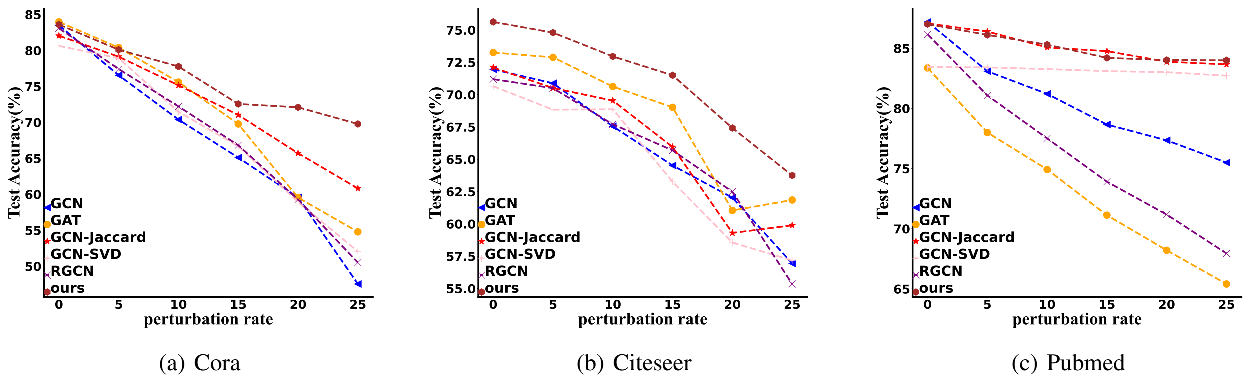
5.2. Overall Results

We first evaluate the effectiveness of our method on the node classification task and report the classification accuracy of all methods on all datasets in Table 1. Obviously, we can observe that the proposed method consistently achieves large-margin outperformance over all baselines across all datasets. Specifically, compared to GCN, the proposed method outperforms GCN by a margin of 3.3%, 4.4%, 2.7%, 1.4%, 1.6%, 1.5%, 2.3% on Cora, Citeseer, Pubmed, Computers, Photo, Physics, and CS datasets, respectively. This is because our proposed method improves the correctness of OOC nodes' representation relative to GCN. When compared to the very recent work AERO-GNN, the proposed model achieves 0.9%, 2.1%, 1.1%, 0.7%, 1.8%, 1.0% and 1.4% improvements.

To better examine the effectiveness of our proposed on OOC nodes, we further evaluate the model's improvement over GCN on these two types of nodes (*i.e.,* OOC nodes and UC nodes) separately in the test set. The results are in Table 2. Firstly, We can observe that almost all methods (including ours) have similar effectiveness on UC nodes, this is because the UC nodes exhibit aligned neighbor label distributions and feature distributions, making them well-suited for GCN. Consequently, GCN has been able to learn the representation of UC nodes very well. Second, we find that the gap in performance between models is mainly centered on the OOC nodes. This suggests that previous work is implicitly improving the effectiveness of OOC nodes, but since they do not capture the key issues of OOC nodes, the improvement is not large relative to what we proposed. In particular, the proposed DaGCN on average improves by 5.9%, compared to the GCN on OOC nodes. Therefore, the study of OOC nodes is particularly important in graph semi-supervised learning.

5.3. Adversarial Robustness

Recent researches point out that graph neural networks are vulnerable to adversarial attacks (Jin et al., 2020; 2021; Li et al., 2020a; Huang et al., 2023a). Specifically, making slight changes to the graph structure that go unnoticed can significantly decrease the performance of graph neural networks. This susceptibility has sparked significant concerns regarding the application of graph neural networks in safety-critical scenarios. Adversarial attacks on graphs are primarily attacks on graph structures, inspired by (Jin et al., 2021) shows that edges with high similarity will be removed



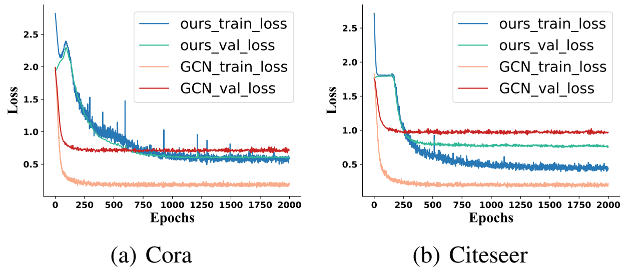
(a) Cora (b) Citeseer (c) Pubmed

Figure 3. Metattack on Cora, Citeseer and Pubmed.

and edges with low similarity will be connected. Recall that DaGCN has the DSKNN component which selects the k neighbors with the highest similarity to connect (*i.e.,* connecting nodes with high similarity and not connecting nodes with low similarity), which can somewhat better recover the clean graph information. Therefore, in this subsection, we are interested in examining its potential benefit on adversarial robustness. We adopt the state-of-the-art nontargeted attack method *metattack* (Sun et al., 2020a) to perturbed graphs. To make a better comparison, we include GCN, GAT, and the state-of-the-art defense models, GCN-Jaccard (Wu et al., 2019b), GCN-SVD (Peng et al., 2022), and RGCN (Zhu et al., 2019) implemented by DeepRobust (Li et al., 2021) as baselines and use the default hyper-parameter settings in the authors' implementations. As we can observe from Figure 3, our proposed method consistently improves the performance of GCN under different perturbation rates of adversarial attack on all three datasets and also achieves the best performance in most cases compared to the defense of GCN. Specifically, DaGCN improves GCN by a larger margin when the perturbation rate is higher. For example, it achieves over 20% improvement over GCN under the 25% perturbation rate on the Cora dataset. These observations suggest the potential robustness of DaGCN against adversarial attacks.

5.4. Analysis Generalization Ability

Section 4.1 mentioned that the presence of OOC nodes affects the generalization ability of GCNs. Since our method solves some of the problems with the OOC nodes, our method is promising to improve the generalization ability of GCNs. To achieve this, we analyze the model's loss on both training and validation sets on Cora and Citeseer. A small gap between the two losses (*i.e.,* , generalization gap (Keskar et al., 2016))indicates a model with good generalization. Figure 4 reports the results for DaGCN and GCN.



(a) Cora (b) Citeseer

Figure 4. Generalization gap analysis.

We can observe that the generalization gap of DaGCN is significantly smaller for GCN, and even on the Cora dataset, the DaGCN generalization gap is almost 0. This observation demonstrates our proposed improve the GCN's generalization ability. More experiments are in Appendix C.

6. Conclusion

In this paper, we revisit the GCN's assumption and its operation and find that the GCN's feature smoothing operation can not completely confirm the label smoothness assumption, thus we design the corresponding algorithm to find the nodes that result in this problem (*i.e.,* OOC nodes), then we proposed the DaGCN around the OOC nodes' properties. Extensive experiments demonstrate that our proposed method, specifically designed for OOC nodes, achieves significant performance in various aspects, including effectiveness, adversarial robustness, and generalizability. Moreover, we arrive at a compelling and insightful conclusion: the actual improvements of current advanced GCNs are predominantly manifested in OOC nodes. Meanwhile, vanilla GCN has been able to achieve high-quality representation learning on UC nodes.

Acknowledgement

Xiaofeng Zhu was supported in part by the National Key Research and Development Program of China under Grant (No.2022YFA1004100), and the Basic and Applied Basic Research Projects of Guangdong Province of China under Grant (No.2024A1515011381). Xiaoshuang Shi was supported by the National Natural Science Foundation of China (No.62276052), and the Sichuan Science and Technology Program (No. 2024YFHZ0268).

Impact Statement

This paper aims to advance the field of Graph Machine Learning. It provides a new perspective on Graph Convolutional Networks (GCNs), which could offer novel approaches to enhance research on graph neural networks. Specifically, we provide an algorithm that classifies graph data nodes into those suitable for GCNs (*i.e.,* UC nodes) and those not suitable for GCNs (*i.e.,* OOC nodes). This classification can guide future research to focus on OOC nodes. Additionally, we summarize the characteristics of OOC nodes, offering a feasible approach to improve their performance. Our findings are expected to improve the accuracy and efficiency of GCNs, thereby providing a more robust framework for future research. Furthermore, our work can promote various applications based on GCNs, opening new avenues for practical implementations and further exploration in the field.

References

Ash, R. B. *Information theory*. Courier Corporation, 2012.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems(NeurIPS)*, 32:5050–5060, 2019.

Bi, W., Du, L., Fu, Q., Wang, Y., Han, S., and Zhang, D. Make heterophily graphs better fit gnn: A graph rewiring approach. *arXiv preprint arXiv:2209.08264*, 2022.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations(ICLR)*, 2014.

Chapelle, O., Weston, J., and Schölkopf, B. Cluster kernels for semi-supervised learning. In *Advances in neural information processing systems (NeurIPS)*, volume 15, 2002.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *International conference on machine learning(ICML)*, pp. 1725–1735. PMLR, 2020.

Davies, D. L. and Bouldin, D. W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence(TPAMI)*, (2):224–227, 1979.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems(NeurIPS)*, volume 29, pp. 3837–3845, 2016.

Du, L., Shi, X., Fu, Q., Ma, X., Liu, H., Han, S., and Zhang, D. Gbk-gnn: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In *Proceedings of the Web Conference(WWW)*, pp. 1550–1558, 2022.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *Proceedings of the Web Conference(WWW)*, pp. 417–426, 2019.

Fan, W., Liu, X., Jin, W., Zhao, X., Tang, J., and Li, Q. Graph trend filtering networks for recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR)*, pp. 112–121, 2022.

Fatemi, B., El Asri, L., and Kazemi, S. M. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems(NeurIPS)*, 34:22667–22681, 2021.

Feng, J., Chen, Y., Li, F., Sarkar, A., and Zhang, M. How powerful are k-hop message passing graph neural networks. *Advances in Neural Information Processing Systems(NeurIPS)*, 35:4776–4790, 2022.

Feng, W., Zhang, J., Dong, Y., Han, Y., Luan, H., Xu, Q., Yang, Q., Kharlamov, E., and Tang, J. Graph random neural networks for semi-supervised learning on graphs. In *Advances in neural information processing systems(NeurIPS)*, volume 33, pp. 22092–22103, 2020.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations, (ICLR)*, 2018.

Giraldo, J. H., Skianis, K., Bouwmans, T., and Malliaros, F. D. On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management(CIKM)*, pp. 566–576, 2023.

Grandvalet, Y. and Bengio, Y. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems(NeurIPS)*, 2004.

Grover, A., Zweig, A., and Ermon, S. Graphite: Iterative generative modeling of graphs. In *International conference on machine learning(ICLR)*, pp. 2434–2444. PMLR, 2019.

Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., and Tang, J. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining(SIGKDD)*, pp. 594–604, 2022.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems(NeurIPS)*, 33:22118–22133, 2020.

Huang, J., Du, L., Chen, X., Fu, Q., Han, S., and Zhang, D. Robust mid-pass filtering graph convolutional networks. In *Proceedings of the Web Conference(WWW)*, pp. 328–338, 2023a.

Huang, J., Li, P., Huang, R., Chen, N., and Zhang, A. Revisiting the role of heterophily in graph representation learning: An edge classification perspective. *ACM Transactions on Knowledge Discovery from Data(TKDD)*, 18: 13:1–13:17, 2023b.

Ji, F., Lee, S. H., Meng, H., Zhao, K., Yang, J., and Tay, W. P. Leveraging label non-uniformity for node classification in graph neural networks. In *International Conference on Machine Learning(ICML)*, pp. 14869–14885, 2023.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (SIGKDD)*, pp. 66–74, 2020.

Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., and Tang, J. Adversarial attacks and defenses on graphs. *ACM SIGKDD Explorations Newsletter*, 22(2):19–34, 2021.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations(ICLR)*, 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *International Conference on Learning Representations(ICLR)*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *International Conference on Learning Representations(ICLR)*, 2011.

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

Lee, S. Y., Bu, F., Yoo, J., and Shin, K. Towards deep attention in graph neural networks: Problems and remedies. In *International Conference on Machine Learning(ICML)*, volume 202, pp. 18774–18795, 2023.

Li, M., Guo, X., Wang, Y., Wang, Y., and Lin, Z. G^2 cn: Graph gaussian convolution networks with concentrated graph filters. In *International Conference on Machine Learning(ICML)*, pp. 12782–12796. PMLR, 2022.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence(AAAI)*, volume 32, 2018.

Li, Y., Bai, S., Zhou, Y., Xie, C., Zhang, Z., and Yuille, A. Learning transferable adversarial examples via ghost networks. In *Proceedings of the AAAI Conference on Artificial Intelligence(AAAI)*, volume 34, pp. 11458–11465, 2020a.

Li, Y., Qian, B., Zhang, X., and Liu, H. Graph neural network-based diagnosis prediction. *Big Data*, 8(5):379–390, 2020b.

Li, Y., Jin, W., Xu, H., and Tang, J. Deeprobust: a platform for adversarial attacks and defenses. In *Proceedings of the AAAI Conference on Artificial Intelligence(AAAI)*, volume 35, pp. 16078–16080, 2021.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining(SIGKDD)*, pp. 338–348, 2020.

Liu, S., Ying, R., Dong, H., Li, L., Xu, T., Rong, Y., Zhao, P., Huang, J., and Wu, D. Local augmentation for graph neural networks. In *International Conference on Machine Learning(ICML)*, pp. 14054–14072. PMLR, 2022a.

Liu, Y., Jin, M., Pan, S., Zhou, C., Zheng, Y., Xia, F., and Philip, S. Y. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering(TKDE)*, 35(6):5879–5900, 2022b.

Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., and Zhang, X. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining(WSDM)*, pp. 779–787, 2021.

Malekzadeh, M., Hajibabaee, P., Heidari, M., Zad, S., Uzuner, O., and Jones, J. H. Review of graph neural network in text classification. In *2021 IEEE 12th annual ubiquitous computing, electronics & mobile communication conference (UEMCON)*, pp. 0084–0091, 2021.

Min, Y., Wenkel, F., and Wolf, G. Scattering gcn: Overcoming oversmoothness in graph convolutional networks. *Advances in neural information processing systems(NeurIPS)*, 33:14498–14508, 2020.

Natekar, P. and Sharma, M. Representation based complexity measures for predicting generalization in deep learning. *arXiv preprint arXiv:2012.02775*, 2020.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. *Advances in neural information processing systems(NeurIPS)*, 30: 5947–5956, 2017.

Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S., and Nguyen, T. M. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International Conference on Machine Learning(ICML)*, volume 202, pp. 25956–25979, 2023.

Peng, S., Sugiyama, K., and Mine, T. Svd-gcn: A simplified graph convolution paradigm for recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management(CIKM)*, pp. 1625–1634, 2022.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations(ICLR)*, 2019.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems(NeurIPS)*, volume 28, pp. 3483–3491, 2015.

Song, R., Giunchiglia, F., Zhao, K., and Xu, H. Topological regularization for graph neural networks augmentation. *arXiv preprint arXiv:2104.02478*, 2021.

Sun, Y., Wang, S., Tang, X., Hsieh, T.-Y., and Honavar, V. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference(WWW)*, pp. 673–683, 2020a.

Sun, Z., Yin, H., Chen, H., Chen, T., Cui, L., and Yang, F. Disease prediction via graph neural networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3): 818–826, 2020b.

Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations(ICLR)*, 2022.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations, (ICLR)*. OpenReview.net, 2018.

Wang, H. and Leskovec, J. Combining graph convolutional neural networks and label propagation. *ACM Transactions on Information Systems (TOIS)*, 40(4):1–27, 2021.

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., and Pei, J. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining(SIGKDD)*, pp. 1243–1253, 2020.

Wang, X., Liu, H., Shi, C., and Yang, C. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems(NeurIPS)*, 34:23768–23779, 2021.

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

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence(IJCAI)*, pp. 4816–4823, 2019b.

Wu, L., Lin, H., Gao, Z., Tan, C., Li, S., et al. Graphmixup: Improving class-imbalanced node classification on graphs by self-supervised context prediction. *arXiv preprint arXiv:2106.11133*, 2021.

Wu, L., Chen, Y., Shen, K., Guo, X., Gao, H., Li, S., Pei, J., Long, B., et al. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328, 2023.

Yang, C., Wu, Q., Wang, J., and Yan, J. Graph neural networks are inherently good generalizers: Insights by bridging gnns and mlps. In *International Conference on Learning Representations(ICLR)*, 2023a.

Yang, X., Tan, C., Liu, Y., Liang, K., Wang, S., Zhou, S., Xia, J., Li, S. Z., Liu, X., and Zhu, E. Convert:

Contrastive graph clustering with reliable augmentation. In *Proceedings of the 31st ACM International Conference on Multimedia (MM)*, pp. 319–327, 2023b.

Yang, X., Wang, Y., Liu, Y., Wen, Y., Meng, L., Zhou, S., Liu, X., and Zhu, E. Mixed graph contrastive network for semi-supervised node classification. *ACM Trans. Knowl. Discov. Data*, 2024. ISSN 1556-4681.

Zhang, A., Huang, J., Li, P., and Zhang, K. Building shortcuts between distant nodes with biaffine mapping for graph convolutional networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 18(6):1–21, 2024.

Zhang, W., Yang, M., Sheng, Z., Li, Y., Ouyang, W., Tao, Y., Yang, Z., and Cui, B. Node dependent local smoothing for scalable graph learning. In *Advances in Neural Information Processing Systems(NeurIPS)*, volume 34, pp. 20321–20332, 2021.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations(ICLR)*, 2020.

Zhao, T., Zhang, X., and Wang, S. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining(WSDM)*, pp. 833–841, 2021.

Zhou, D., Bousquet, O., Lal, T., Weston, J., and Schölkopf, B. Learning with local and global consistency. In *Advances in neural information processing systems(NeurIPS)*, volume 16, pp. 321–328, 2003.

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining(SIGKDD)*, pp. 1399–1407, 2019.

Zhu, D.-H., Dai, X.-Y., and Chen, J.-J. Pre-train and learn: Preserving global information for graph neural networks. *Journal of Computer Science and Technology*, 36:1420–1430, 2021.

Zhu, X. *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.

A. Theoretical Proof

A.1. Proof for Loss of Conditional Variational Auto-Encoder

Proof. The specific conditional variational auto-encoder loss function is derived as follows,

$$\begin{aligned}
\log p(X_u \, j \, X_v) &= \int q(\mathbf{z} \, j \, X_u, X_v) \log p(X_v \, j \, X_v) \, \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, X_v)}{p(X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, X_v)\, p(X_v, X_v, \mathbf{z})}{p(X_v)\, p(X_v, X_v, \mathbf{z})} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_v, X_v) \log \frac{p(X_u, X_v, \mathbf{z})}{p(X_v)} \frac{1}{\frac{p(\boldsymbol{X}_u; \boldsymbol{X}_v; \mathbf{z})}{p(\boldsymbol{X}_u; \boldsymbol{X}_v)}} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, \mathbf{z} \, j \, X_v)}{p(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, \mathbf{z} \, j \, X_v)}{p(\mathbf{z} \, j \, X_u, X_v)} \frac{q(\mathbf{z} \, j \, X_u, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \left(\log \frac{p(X_u, \mathbf{z} \, j \, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} + \log \frac{q(\mathbf{z} \, j \, X_u, X_v)}{p(\mathbf{z} \, j \, X_u, X_v)} \right) \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, \mathbf{z} \, j \, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz} + KL(q(\mathbf{z} \, j \, X_u, X_v) k p(\mathbf{z} \, j \, X_u, X_v)) \\
& \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, \mathbf{z} \, j \, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz}
\end{aligned} \tag{20}$$

$$\begin{aligned}
L_{ELBO} &= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, \mathbf{z} \, j \, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u, X_v, \mathbf{z})}{q(\mathbf{z} \, j \, X_u, X_v)\, p(X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u \, j \, X_v, \mathbf{z})\, p(X_v, \mathbf{z})}{q(\mathbf{z} \, j \, X_u, X_v)\, p(X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(X_u \, j \, X_v, \mathbf{z})\, p(\mathbf{z} \, j \, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz} \\
&= \int q(\mathbf{z} \, j \, X_u, X_v) \log \frac{p(\mathbf{z} \, j \, X_v)}{q(\mathbf{z} \, j \, X_u, X_v)} \mathrm{dz} + \int q(\mathbf{z} \, j \, X_u, X_v) \log p(X_u \, j \, X_v, \mathbf{z}) \, \mathrm{dz} \\
&= \quad KL(q(\mathbf{z} \, j \, X_u, X_v) jj p(\mathbf{z} \, j \, X_v)) + \mathrm{E}_{q(\mathbf{z} j \boldsymbol{X}_u; \boldsymbol{X}_v)}(p(X_u \, j \, X_v, \mathbf{z}))
\end{aligned} \tag{21}$$

□

A.2. Proof for Theorem 4.1

Theorem A.1. *For nodes with unknown labels in the graph, the upper bound of the GCN's generalization ability reaches optimal if the true labels of these nodes are equal to the labels generated by the LPA.*

Proof. To facilitate a clear description, we first consider the binary classification task. We prove the theorem by analyzing the exact lower bound of the complexity measure of the graph convolutional layer. Based on the Consistency of Representations and Fisher discriminant analysis, which all use the ratio of the inter-class variance to the intra-class variance as an indicator, we rewrite the Consistency of Representations for the convenience of theoretical analysis:

$$C = \frac{S_0 + S_1}{M_{0;1}}. \tag{22}$$

We define P_0 as the probability that a node's neighbor belongs to the '0-th' class, and I_0 as the probability that the node itself belongs to the '0-th' class. Thus, we can calculate the cluster centroid after GCN smoothed features:

$$\begin{aligned} \mu_{O_0} &= \mathrm{E}[O_0^i] = \mathrm{E}[\mathbf{W} \sum_{j \in N_i} \frac{1}{d_i} \mathbf{X}^j] \\ &= \mathbf{W}(I_0 P_0 \mu_{X_0} + I_0(1 - P_0)\mu_{X_1}), \end{aligned} \tag{23}$$

where $\mathbf{X}^j$ is the 'j-th' node feature and μ_{X_i} is the cluster centroid of the node features of class i. Likewise, we have:

$$\mu_{O_1} = \mathbf{W}(I_1 P_1 \mu_{X_1} + I_1(1 - P_1)\mu_{X_0}). \tag{24}$$

Then, the $M_{0,1}$ can be computed by:

$$\begin{aligned} M_{0,1} &= \|\mu_{O_a} - \mu_{O_b}\| \\ &= \|\mathbf{W}(I_0 P_0 \mu_{X_0} + I_0(1 - P_0)\mu_{X_1} - (I_1 P_1 \mu_{X_1} + I_1(1 - P_1)\mu_{X_0}))\| \\ &= \|\mathbf{W}(I_0 P_0 \mu_{X_0} + I_0 \mu_{X_1} - I_0 P_0 \mu_{X_1} - I_1 P_1 \mu_{X_1} - I_1 \mu_{X_0} + I_1 P_1 \mu_{X_0})\| \\ &= (I_0 P_0 + I_1 P_1)\|\mathbf{W}(\mu_{X_0} - \mu_{X_1})\| + \|I_0 \mu_{X_1} - I_1 \mu_{X_0}\| \\ &\leq (I_0 P_0 + I_1 P_1)\|\mathbf{W}(\mu_{X_0} - \mu_{X_1})\| + \|\mu_{X_1}\| + \|\mu_{X_0}\|. \end{aligned} \tag{25}$$

Then S_0^2 is calculated by:

$$\begin{aligned} S_0^2 &= \mathrm{E}\left[\left\|O_0^{(i)} - \mu_{O_0}\right\|^2\right] = \mathrm{E}\left[< O_0^{(i)} - \mu_{O_0}, O_0^{(i)} - \mu_{O_0} >\right] \\ &= \mathrm{E}[(I_0 P_0)(I_0 P_0 (X_0 - \mu_{X_0})^T \mathbf{W}^T \mathbf{W}(X_0 - \mu_{X_0}))] + \mathrm{E}[I_0(1 - P_0) I_0(1 - P_0)(X_1 - \mu_{X_1})^T \mathbf{W}^T \mathbf{W}(X_1 - \mu_{X_1}))] \\ &= I_0^2 P_0^2 \mathrm{E}[\|W(X_0 - \mu_{X_0})\|] + I_0^2(1 - P_0)^2 \mathrm{E}[\|W(X_1 - \mu_{X_1})\|]. \end{aligned} \tag{26}$$

Similarly, we have:

$$\begin{aligned} S_1^2 &= \mathrm{E}\left[\left\|O_1^{(i)} - \mu_{O_1}\right\|^2\right] = \mathrm{E}\left[< O_1^{(i)} - \mu_{O_1}, O_1^{(i)} - \mu_{O_1} >\right] \\ &= \mathrm{E}[(I_1 P_1)(I_1 P_1 (X_1 - \mu_{X_1})^T \mathbf{W}^T \mathbf{W}(X_1 - \mu_{X_1}))] + \mathrm{E}[I_1(1 - P_1) I_1(1 - P_1)(X_0 - \mu_{X_0})^T \mathbf{W}^T \mathbf{W}(X_0 - \mu_{X_0}))] \\ &= I_1^2 P_1^2 \mathrm{E}[\|W(X_1 - \mu_{X_1})\|] + I_1^2(1 - P_1)^2 \mathrm{E}[\|W(X_0 - \mu_{X_0})\|], \end{aligned} \tag{27}$$

where $< \cdot, \cdot >$ is inner production. For simplicity, let $\sigma_0^2 = \mathrm{E}[\|W(X_0 - \mu_{X_0})\|]$ and $\sigma_1^2 = \mathrm{E}[\|W(X_1 - \mu_{X_1})\|]$, then the above equation can then be simplified to:

$$S_0^2 = (I_0 P_0)^2 \sigma_0^2 + (I_0(1 - P_0))^2 \sigma_1^2 - I_0^2 \frac{\sigma_0^2 \sigma_1^2}{\sigma_0^2 + \sigma_1^2}. \tag{28}$$

Similarly, we have:

$$S_1^2 = (I_1 P_1)^2 \sigma_1^2 + (I_1(1 - P_1))^2 \sigma_1^2 - I_1^2 \frac{\sigma_0^2 \sigma_1^2}{\sigma_0^2 + \sigma_1^2}. \tag{29}$$

Then the complexity measure can be represented as:

$$C = \frac{\sqrt{S_0^2 + S_1^2 + 2 S_0 \cdot S_1}}{M_{0,1}} \geq \frac{2\sigma_0 \sigma_1 (I_0 + I_1)^2}{\sqrt{\sigma_0^2 + \sigma_1^2} \cdot ((I_0 P_0 + I_1 P_1)\|\mathbf{W}(\mu_{X_0} - \mu_{X_1})\| + \|\mu_{X_1}\| + \|\mu_{X_0}\|)}. \tag{30}$$

Thus, we obtain a lower bound of complexity measure. Also this is the upper bound of the generalization ability. Notice that σ_0 and σ_1 could not be zero, otherwise the classification problem is meaningless. We observe the above equation for nodes with unknown labels and analysis the relationship between distribution of label I_0, I_1 and lower bound of complexity measure, we find that the probability of their own label (*i.e.,* I_0 or I_1) and the probability of their neighbors' labels (*i.e.,* P_0 or P_1) affect the upper bound on their generalization ability. Since $I_0 + I_1 = 1$, we analyze term $(I_0P_0 + I_1P_1)$,

$$(I_0P_0 + I_1P_1) = \frac{1}{n}\sum_{i}^{n} I_{0;i}P_{0;i} + I_{1;i}P_{1;i} \tag{31}$$

where $I_{0;i} \in \{0, 1\}$ is the binary probability that the 'i-th' node label belongs to class 0 where $I_{1;i} = 1 - I_{0;i}$ and $P_{0;i}$ is the probability that the 'i-th' node whose neighbor belongs to class 0. In order to minimize the lower bound of complexity measure, *i.e.,* to maximize the upper bound of generalization ability, it is necessary to maximize $(I_0P_0 + I_1P_1)$ here. Obviously, the maximum $(I_0P_0 + I_1P_1)$ is obtained at $I_{0;i} = argmax(P_{1;i}P_{0;i})$.

Let's look at Label Propagation Algorithm(LPA). For nodes with unknown labels,

$$\widehat{y_i} = \frac{1}{d_i}\sum_{j \in N_i} y_j . \tag{32}$$

Then the probability that the LPA predicts that the 'i-th' node belongs to class 0 can be obtained:

$$\widehat{I_{0;i}} = argmax(\frac{1}{d_i}\sum_{j \in N_i} y_i == 1, \frac{1}{d_i}\sum_{j \in N_i} y_i == 0) = argmax(P_{1;i}P_{0;i}). \tag{33}$$

Similarly, the probability of predicting the 'i-th' node to belong to class 1 is:

$$\widehat{I_{1;i}} = argmax(\frac{1}{d_i}\sum_{j \in N_i} y_i == 0, \frac{1}{d_i}\sum_{j \in N_i} y_i == 1) = argmax(P_{0;i}P_{1;i}). \tag{34}$$

Thus for binary classification, the upper bound on the generalization ability is maximized when the labels of the unknown label set are distributed as LPA-generated labels.

Next, we consider the multi-classification case, to facilitate the calculation, the complexity measure can be write as:

$$C = \frac{1}{k}\sum_{i=0}^{k-1} \max_{i \neq j} \frac{(S_i + S_j)}{M_{i;j}} = \frac{1}{k}\sum_{i=0}^{k-1} \max_{i \neq j} C_{i;j} \tag{35}$$

where $i, j \in 0, 1, \quad , c$ and $i \neq j$. To simplify the equation, we can rewrite $C_{i;j}$ in equation 30 as:

$$C_{i;j} \quad \frac{{}_{1;ij}}{{}_{2;ij} \quad ((I_iP_i + I_jP_j) \ {}_{3;ij} + \ {}_{4;ij})}, \tag{36}$$

where ${}_{1;ij} = 2\sigma_i\sigma_j$, ${}_{2;ij} = \sqrt{\sigma_i^2 + \sigma_j^2}$, ${}_{3;ij} = \left\|\mathbf{W}(\mu_{X_i} - \mu_{X_j})\right\|$ and ${}_{4;ij} = \|\mu_{X_i}\| + \left\|\mu_{X_j}\right\|$. The complexity measure for multi-classification is:

$$\begin{aligned} C \quad & \frac{1}{k}\sum_{i=0}^{k-1} \max_{i \neq j} C_{i;j} = \frac{1}{k}\sum_{i=0}^{k-1} \max_{i \neq j} \frac{{}_{1;ij}}{{}_{2;ij} \quad ((I_iP_i + I_jP_j) \ {}_{3;ij} + \ {}_{4;ij})} \\ &= \frac{1}{k}\sum_{i=0}^{k-1} \max_{i \neq j} \frac{{}_{1;ij}}{{}_{2;ij} \quad ((\frac{1}{n}\sum_r^n I_{i;r}P_{i;r} + I_{j;r}P_{j;r}) \ {}_{3;ij} + \ {}_{4;ij})}. \\ &= \frac{1}{k}\sum_{i=0}^{k-1} \max_{i \neq j} \frac{{}_{1;ij}}{{}_{2;ij}} \frac{1}{\frac{1}{n}\sum_r^n((I_{i;r}P_{i;r} + I_{j;r}P_{j;r}) \ {}_{3;ij} + n \ {}_{4;ij})}. \end{aligned} \tag{37}$$

where in multi-classification case, $I_{i;r} \in \{0,1\}$ is the binary probability and if $I_{i;r} = 1$ then any j on node r have $I_{j;r} = 0$, because a node can only belong to one class. Then, we analyze under what circumstances the label of node v should belong to l (*i.e.*, $I_{l;v} = 1$, $I_{i;v} = 0$, s.t., $i \in [0, 1, \cdots, c] - [l]$) can minimizing Complexity measure's lower bound. The above equation can be:

$$\begin{aligned}
C &\geq \frac{n}{k}\sum_{i=0}^{k-1} \max_{i \neq j} \frac{\Delta_{1;ij}}{\Delta_{2;ij}} \frac{1}{(I_{i;v}P_{i;v} + I_{j;v}P_{j;v})\Delta_{3;ij} + \sum_{r \neq v}^{n}((I_{i;r}P_{i;r} + I_{j;r}P_{j;r})\Delta_{3;ij} + n\Delta_{4;ij})} \\
&= \frac{n}{k}\Big(\sum^{i=l|j=l} \max_{i \neq j} \frac{\Delta_{1;lj}}{\Delta_{2;lj}} \frac{1}{(I_{l;v}P_{l;v})\Delta_{3;lj} + \sum_{r \neq v}^{n}((I_{l;r}P_{l;r} + I_{j;r}P_{j;r})\Delta_{3;lj} + n\Delta_{4;lj})} \\
&+ \sum^{i \neq l \& j \neq l} \max_{i \neq j} \frac{\Delta_{1;ij}}{\Delta_{2;ij}} \frac{1}{0 + \sum_{r \neq v}^{n}((I_{i;r}P_{i;r} + I_{j;r}P_{j;r})\Delta_{3;ij} + n\Delta_{4;ij})}\Big).
\end{aligned} \tag{38}$$

Since $(I_{l;v}P_{l;v})\Delta_{3;lj} \geq 0$, obviously, we have:

$$\sum^{i=l|j=l} \max_{i \neq j} \frac{1}{(I_{l,v}P_{l,v})\Delta_{3,lj} + \sum_{r \neq v}^{n}((I_{l,r}P_{l,r} + I_{j,r}P_{j,r})\Delta_{3,lj} + n\Delta_{4,lj})} \leq \sum^{i \neq l \& j \neq l} \max_{i \neq j} \frac{\Delta_{1,ij}}{\Delta_{2,ij}} \frac{1}{0 + \sum_{r \neq v}^{n}((I_{i,r}P_{i,r} + I_{j,r}P_{j,r})\Delta_{3,ij} + n\Delta_{4,ij})}. \tag{39}$$

For node v is only relevant to the first term and we can easy to know that when $I_{l;v}P_{l;v}$ is maximized, the first term is minimum. Thus, for any node v, the condition for minimizing Complexity Measure when its label is l is satisfied if and only if the probability of its neighbor labels are belong to l is maximized.

This conclusion is the same as in the case of binary classification: *the Complexity Measure is minimized when the node's own category and the maximum probability category of its neighbors are the same. i.e.,* $I_{i;v} = int(argmax(P_{0;v}, P_{1;v}, \cdots, P_{c;v}) == i)$. This is consistent with the multi-class LPA.

Therefore, for nodes with unknown label, the predictions of LPA happen to coincide with the conditions that make the upper bound on the generalization ability of GCN reaches optimal, so the proof is complete. □

A.3. Proof for Theorem 4.2

Theorem A.2. *Given a undirected graph* $G(V, E)$ *has* n *nodes and* e *edges. Assuming there are* q *nodes in the graph with labels selected uniformly at random. The occurrence probability of nodes that not affected by labels with a two-layer GCN is equal to* $(1-\frac{q}{n})(1-\frac{q}{n-1})\prod_{i=1}^{q}(1-\frac{2m}{n(n-1)-2i})\prod_{i=q}^{2q}(1-\frac{2(m-1)}{n(n-1)-2i})$

Proof. We follow the proof idea of (Fatemi et al., 2021). For simplicity, we refer to nodes that are not influenced by labels as 'starved nodes'. To compute the probability of a node being a starved node, we first compute the probability of the node v being unlabeled and not connected to any labeled nodes then compute the probability of it's neighbor u not being connected to any labeled nodes.

With n nodes and q labels, the probability of a node being labeled is $\frac{q}{n}$, then we have $\Pr(l_v \in l^U) = (1 - \frac{q}{n})$.

In the undirected graph G, there are $\frac{1}{2}n(n-1)$ pairs of nodes that can potentially have an edge between them. Therefore, the probability that node v is not connected to one of the labeled node is $(1 - \frac{m}{\frac{1}{2}n(n-1)-1}) = (1 - \frac{2m}{n(n-1)-2})$. If the node v not being connected to one of the labeled nodes, then total number of edges still remains m and the number of potential connected pair nodes number is $\frac{1}{2}n(n-1)-1$, so we can compute probability that node v being disconnected to the second labeled node is $(1 - \frac{m}{\frac{1}{2}n(n-1)-2}) = (1 - \frac{2m}{n(n-1)-4})$. With the similar reasoning, the probability of node v being disconnected to the $i-st$ labeled node given that it is disconnected from the first $i-1$ labeled nodes is $(1 - \frac{2m}{n(n-1)-2i})$. Therefore, the probability of the node v being unlabeled and not connected to any labeled nodes is equal to $(1 - \frac{q}{n})\prod_{i=1}^{q}(1 - \frac{2m}{n(n-1)-2i})$.

For the node v, given that we know node v is unlabeled, the probability of its neighbor u being unlabeled is $\Pr(l_u \in l^U | l_v \in l^U) = (1 - \frac{q}{n-1})$. Therefore, $\Pr(l_v \in l^U, l_u \in l^U) = (1 - \frac{q}{n})(1 - \frac{q}{n-1})$.

Follow the similar reasoning with node v, the probability that node u being disconnected to first one of the labeled node given that node v is disconnected from all labeled nodes is $(1 - \frac{m}{\frac{1}{2}n(n-1)-q-1}) = (1 - \frac{2(m-1)}{n(n-1)-2(q+1)})$. This is because there are $m \quad 1$ remaining edges, excluding the one connecting v and u, and $\frac{1}{2}n(n \quad 1) \quad q \quad 1$ pairs of nodes that can potentially be connected. Similarly, we can also compute the probability of node u being disconnected to the $i \quad st$ labeled node given that it is disconnected from the first $i \quad 1$ labeled nodes and that v is disconnected from all labeled nodes is $(1 \quad \frac{2m}{n(n \quad 1) \quad 2(q+i)})$. Thus, we can obtain the probability of u being disconnected to any labeled nodes given v being unlabeled and not connected to any labeled nodes is $\prod_{i=q}^{2q}(1 \quad \frac{2(m \quad 1)}{n(n \quad 1) \quad 2i})$.

Finally, we can compute the joint probability of them to get the probability of a node being a starved node, and it is equal to $(1 \quad \frac{q}{n})(1 \quad \frac{q}{n \quad 1})\prod_{i=1}^{q}(1 \quad \frac{2m}{n(n \quad 1) \quad 2i})\prod_{i=q}^{2q}(1 \quad \frac{2(m \quad 1)}{n(n \quad 1) \quad 2i})$.

□

A.4. Proof for The Entropy After the Assembly of Logits Is Greater Than the Entropy of at Least One Perspective Output Logit

Proof. We already know that $H(\quad p_1 + (1 \quad)p_2) \quad H(p_1) + (1 \quad)H(p_2)$ We utilize the counterfactual. Suppose that both $H(p_1)$ and $H(p_2)$ are greater than $H(\quad p_1 + (1 \quad)p_2)$, we have,

$$\begin{aligned} H(\quad p_1 + (1 \quad)p_2) & \quad H(p_1) + (1 \quad)H(p_2) \\ & = \quad H(p_1) + H(p_2) \quad H(p_2) \\ & = (H(p_1) \quad H(p_2)) \quad + H(p_2) \end{aligned} \tag{40}$$

It is very easy to know that $(H(p_1) \quad H(p_2)) \quad + H(p_2)$ is monotone with respect to , so its minimum value should be at both ends, so you can find its minimum value is $min(H(p_1), H(p_2))$ when $\quad = 0$ and $\quad = 1$. Conclusions can be drawn $H(\quad p_1 + (1 \quad)p_2) \quad min(H(p_1), H(p_2))$, which is a contradiction of the hypothesis. Therefore the proof is complete. □

B. Experimental Details

Table 3. The statistics of the datasets

Datasets	Nodes	Edges	Train/Valid/Test Nodes	Features	Classes
Cora	2,708	5,429	140/500/1000	1,433	7
Citeseer	3,327	4,732	120/500/1,000	3,703	6
Pubmed	19,717	44,338	60/500/1,000	500	3
Amazon Photo	7,487	119,043	160/240/7,084	745	8
Amazon Computers	13,381	245,778	200/300/12,881	767	10
Coauthor CS	18,333	81,894	300/450/17,583	6,805	15
Coauthor Physics	34,493	247,962	360/540/33,593	100	18

B.1. Dataset

We use seven benchmark datasets widely used in node classification tasks including three standard citation networks (Kipf & Welling, 2017), namely, Cora, Citeseer, and Pubmed, co-authorship networks (Shchur et al., 2018) namely, Coauthor CS, Coauthor Physics, and co-purchase networks (Shchur et al., 2018) namely, Amazon Computers and Amazon Photo. Table 3 summarizes the statistics of those benchmark datasets.

1. *Citation networks* include Cora, Citeseer and Pubmed. They are composed of papers as nodes and their relationships such as citation relationships, common authoring. Node feature is a one-hot vector that indicates whether a word is present in that paper. Words with frequency less than 10 are removed.

2. *Coauthor CS and Coauthor Physics* are co-authorship graphs based on the Microsoft Academic Graph from the KDD Cup 2016 challenge 3. Here, nodes are authors, that are connected by an edge if they co-authored a paper; node features represent paper keywords for each author's papers, and class labels indicate the most active fields of study for each author.

Following the setting of prior work (Kipf & Welling, 2017), we apply the standard fixed training/validation/testing split on Cora, Citeseer , and Pubmed. For the Coauthor CS and Coauthor Physics datasets, we follow (Shchur et al., 2018; Liu et al., 2020) utilize 20 labeled nodes per class for training, 30 labeled nodes for validation, and the rest for the testing set.

B.2. Percentage of OOC Nodes

We statistics the number and percentage of OOC nodes on all datasets in Table 5, we can observe that the percentage of OOC nodes is significant.

Table 4. The number and percentage of OOC nodes of whole datasets.

Datasets	Cora	Citeseer	Pubmed	Computers	Photo	Physics	CS
Numberof UC nodes	1552	1491	13915	10250	6006	23301	11080
Number of OOC nodes	1016	1716	5742	3302	1484	11092	6953
Percentage of OOC nodes	39.60%	53.50%	29.20%	25.00%	20.20%	32.50%	38.60%

B.3. Number of UC nodes on the DSKNN Graph

We list the number of UC nodes in the test set on the DSKNN graph in the following table, where the ”Combined Graph” in the table is the union of UC nodes from the original graph and the DSKNN graph.

Table 5. The number and percentage of OOC nodes of whole datasets.

Number of UC nodes	Cora	Citeseer	Pubmed
Original Graph	633	485	708
DSKNN Graph	660	711	720
Combine Graph	833	840	879
Improve Ratio	31.6%	73.2%	24.2%

Based on the above information, we have the following observations:

First, on all three datasets, the number of UC nodes in the DSKNN graph is greater than that in the original graph, especially on the Citeseer dataset. This demonstrates that the DSKNN graph can increase the occurrence of UC nodes.

Second, the overlap of the UC nodes between the original graph and the DSKNN graph is low. This indicates that integration information from both the original graph and the DSKNN graph can increase the likelihood of UC node occurrences.

Therefore, the integration in the DSKNN graph can significantly reduce the impact of OOC nodes.

B.4. Baseline

The comparison methods include two traditional GNN methods and four state-of-the-art GCN promotion methods:

- **GCN** (Kipf & Welling, 2017) is a semi-supervised graph convolutional network model which learns node representations by aggregating information from neighbors.
- **GAT** (Velickovic et al., 2018) is a graph neural network model using the attention mechanism as the weights to smooth node features.
- **APPNP** (Gasteiger et al., 2018) decouples prediction and propagation with performing personalized propagation of neural predictions.
- **DAGNN** (Liu et al., 2020) adaptively incorporate information from large receptive felds.
- w**GCN** (Ji et al., 2023) select the large label non-uniformity nodes join into the train set and random drop the small label non-uniformity nodes’s edge. Then use GCN as the backbone.

- **AERO-GNN** (Lee et al., 2023) implements a deep graph attention network using symmetric normalized attention weights and node-level HOP attention.

B.5. Best Hyper-Parameters

Table 6 reports the detailed hyperparameters of DaGCN. Note that we only performed simple tuning. Regarding the selection of the k value, we found it to be insensitive, and the average degree of all datasets is less than 20. So it was set to 20 for all datasets.

Table 6. Hyperparameters of DaGCN for reproducibility.

Hyperparameter	Cora	Citeseer	Pubmed	Photo	Computers	CS	Physics
λ	1.3	1.1	0.1	1.5	1.3	1.3	1.3
τ	0.5	0.5	0.6	0.1	0.4	0.5	0.5
Learning rate	0.01	0.01	0.02	0.01	0.01	0.01	0.01
L2 weight decay	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4	5e-4
Dropout rate	0.6	0.6	0.5	0.6	0.5	0.6	0.6
Hidden layer size	8	8	4	64	64	128	128
k	20	20	20	20	20	20	20
I( )	True	True	False	False	False	False	False

C. Additional Experiments

C.1. Model Complexity and Running Time

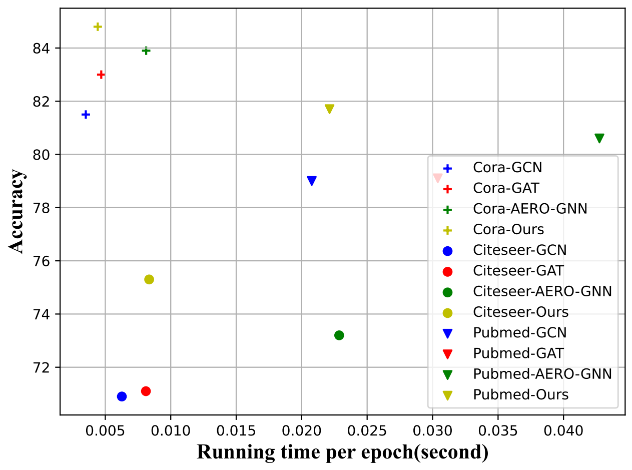
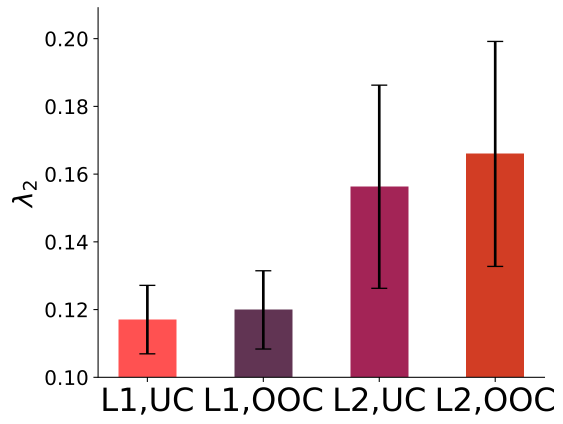


Figure 5. Accuracy v.s. running time.

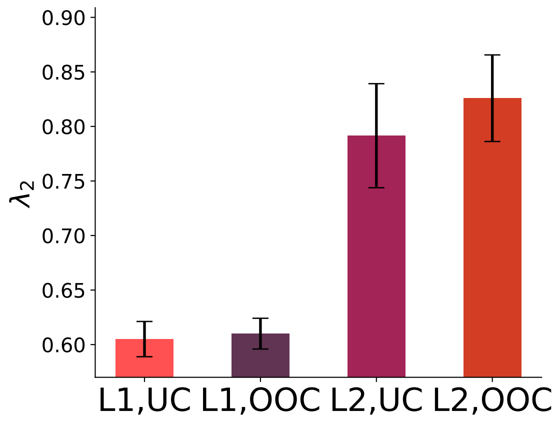
The complexity of feature smoothing on the origin graph is $O(jEjfh)$, where jEj is the number of edges, h is the number of hidden units, and smoothing on the DSKNN graph is $O(kjVjfh)$ where jVj is the number of nodes. The adaptive node-level assembling module consists of a fully connected layer and output dimension is 1, thus which complexity is $O(jVjf)$. Since we set $kjVj > jEj$, so running the DaGCN layer takes $O(kjVjfh)$ computational time, which is slightly above GCN $O(jEjfh^0)$. Actually, we use a very small number of hidden units, the actually running time and accuracy are shown in Figure 5. We can observe that, our proposal obtains an excellent balance between accuracy and running time.

C.2. Analysis Adaptive Node-level Assembling

To study how the weights are learned by adaptive node-level assembling mechanism. We visualize the learned value on DSKNN sides in Fig 6.



(a) Cora



(b) Pubmed

Figure 6. Visualization of average node-level adaptive weights of UC nodes and OOC nodes on each layer on Cora and Pubmed.

First, the learned values vary in a range, indicating that the designed aggregation scheme can adapt the information differently for individual nodes. In particular, the OOC nodes learn that the average weights of the weights for the DSKNN side are heavier than those of the UC nodes, especially in the second layer. This indicates that OOC nodes benefit more from the DSKNN side relative to UC nodes.

Second, the weights learned by each layer are differentiated, and we can observe that the weights of the second layer tend to be larger, which suggests that our adaptive node-level assembling mechanism can adaptively learn different information from each layer.

C.3. Ablation Study

To get a better understanding of how different components affect the model performance, we conduct ablation studies on citation networks. Specifically, we build the following ablations:

- **Without Virtual Neighbor Generation(VNG):** The input only utilizes the original features, which do not concatenate the $\overline{\mathbf{X}}$.
- **Without Real Neighbor Generation (RNG):** We replace the DSKNN-graph with the origin graph.
- **Without Entropy Reduction Loss(ERL):** We only use the cross entropy loss, *i.e.*, $\beta = 0$.

In Table 7, we have two observations. First, all DaGCN variants with some components removed witness clear performance drops when compared to the full model, suggesting that each of the designed components contributes to the success of DaGCN. Second, DaGCN without any component still outperforms GCN, demonstrating that even incomplete DaGCN can perform well in semi-supervised learning.

Table 7. Ablation study results (%).

Ablation	Cora	Citeseer	Pubmed
DaGCN	84.8 $_{0.53}$	75.3 $_{0.41}$	81.7 $_{0.88}$
- w/o VNG	84.2 $_{0.96}$	74.5 $_{0.66}$	81.2 $_{1.00}$
- w/o RNG	83.6 $_{0.46}$	73.6 $_{0.37}$	80.7 $_{0.62}$
- w/o ERL	84.0 $_{0.56}$	73.8 $_{0.72}$	81.3 $_{0.75}$
GCN	81.5 $_{0.82}$	70.9 $_{0.71}$	79.0 $_{0.52}$

C.4. Analysis Parameter Sensitivity

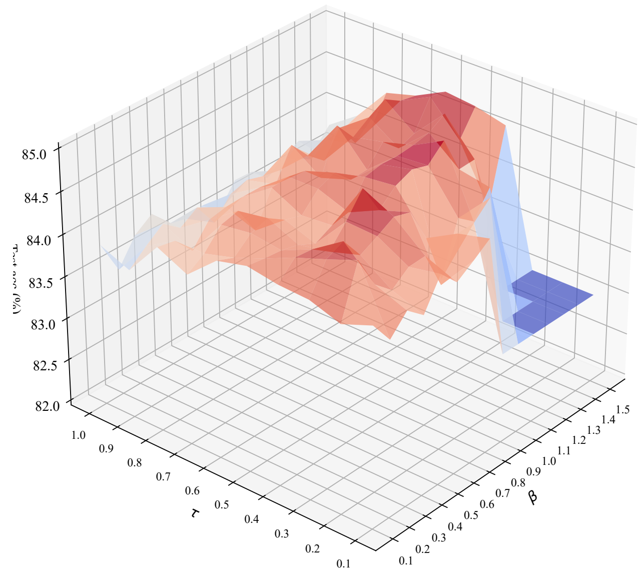


Figure 7. The classification performance of the proposed method at different parameter settings (*i.e.,* τ, β) on the Cora dataset.

Analyze τ, β: In the proposed method, we employ the non-negative parameters(*i.e.,* τ, β) to achieve the temperature control of entropy reduction and a trade-off between each term of the entropy reduction loss and cross-entropy loss. To investigate the impact of τ and β with different settings, we conduct the node classification on the Cora dataset by varying the value of parameters in the range of [0.0, 1.0] for τ and [0.0, 1.5] for β, then reporting the results in Figure 7.

From Figure 7, we have the following observations: First, the temperature parameter τ is significantly important, since when τ is in the interval [0.4, 0.8], the model performance maintains an excellent level. This is because when the τ is closer to 0, the distribution will approach a Dirac (i.e., one-hot) distribution, then the logits confidence of some false prediction are also increased too much, thus making the model's decision boundaries away from the misclassified samples, in extreme cases causing the model to have an undesired situation where all samples are on the same side of the decision boundary (*i.e.,* $\tau = 0$ and β is large). Second, if we ensure that τ is in a suitable range, the selection of β is not sensitive.

Analyze k: As shown in Figure 8. Our method achieves the best performance with k = 20. It is greater than the average degree of these three datasets.

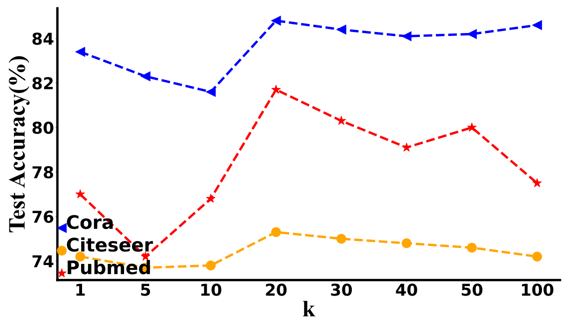


Figure 8. The classification performance of the proposed method under various k on the Cora, Citeseer, and Pubmed datasets.

Table 8. Test accuracy (%) on ogbn-arxiv.

Model	Ogbn-arxiv
GCN	71.7 0.29
GraphSAGE	71.5 0.27
DAGNN	72.1 0.19
DaGCN	**72.4 0.21**

C.5. Validation on Large-scale Graph

We further validate the proposed method on the large-scale graph, *i.e.,* ogbn-arxiv (Hu et al., 2020), which has 139,343 nodes and 1,166,243 edges. The results are shown in Table 8, we can see that our proposed method still maintains relatively good results on graph datasets of such size.

D. Discuss on heterophily

Currently, heterophily is a hot research topic, and it seems that heterophily may lead to the emergence of OOC nodes, so we experimentally observed the heterophily ratio of UC nodes and OOC nodes, as shown in Table 9. We find that node-level heterophily is not one of the factors leading to the existence of OOC nodes.

Table 9. Homophily ratio on representative datasets.

Homophily ratio	Cora	Citeseer
Avr. all nodes	0.81	0.71
Avr. OOC nodes	0.78	0.71