Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 8641220, 12 pages
https://doi.org/10.1155/2022/8641220

Research Article

NC-GNN: Consistent Neighbors of Nodes Help More in Graph Neural Networks

Ming Xu, Baoming Zhang, Hualei Yu, Jinliang Yuan, and Chongjun Wang

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China

Correspondence should be addressed to Chongjun Wang; chjwang@nju.edu.cn

Received 24 March 2022; Accepted 28 May 2022; Published 18 June 2022

Academic Editor: Lei Zhang

Graph neural networks, as the promising methodology in data mining for graph data, currently attract much attention and are broadly applied in graph-based tasks. Existing GNN methods mostly follow the assumption of homophily, where the connected nodes are similar and share the same labels. Most graphs in the real world can satisfy the assumption. However, for the particular nodes, the situation is not always satisfied. The connections between different-labeled nodes will introduce noise in feature aggregation and result in node representation deviating in the wrong direction. In this paper, we focus on the different-labeled neighbors of labeled nodes in the graphs. By regarding aggregation among neighbors as the procedure of node feature reconstruction, we devise a novel metric *neighbor consistency* to measure the difference between nodes and their neighborhoods. In this way, we can evaluate the reliability of nodes after aggregation. Furthermore, we propose a novel method, *Neighbor Consistent Graph Neural Networks* (NC-GNN), to promote the training of graph neural networks by reweighting the influence of labeled nodes based on neighbor consistency scores. Systematic experiments are conducted on benchmark datasets, and the results demonstrate the effectiveness of our method.

1. Introduction

Graphs are widely observed in our daily lives, as graphs can well capture objects' features and the abundant interactions between objects. For instance, following-relations in users form friendship graphs in social networks [1], users' interactions on goods construct user-item graphs in recommendation systems [2], and the communications between mobile phones construct graphs in cellular networks [3]. As an important part of data mining, graph learning attracts much attention to learning latent information in real-world graph data recently. Graph neural networks promote graph learning by introducing deep learning frameworks and achieve great success in most graph mining tasks, like node classification [4, 5], link prediction [6, 7], and graph classification [8–10]. The frameworks are also widely applied in real-life graph tasks, from recommendation [11, 12], social networks [1], text extraction [13, 14], knowledge graphs [15], etc.

The goal of graph neural networks is to encode nodes in the graph into dense and low-dimensional embeddings with node features and graph topology information preserved simultaneously. In this way, nodes or graphs can be represented by the embeddings, and then, we can find out the latent information for downstream tasks. Existing graph neural frameworks mostly follow the manner of message passing neural network (MPNN) [16], namely, updating nodes' representations by aggregating information from neighborhoods. In this way, the representation of nodes is smoothed in each iteration, and the final representation can be used for downstream tasks. The most popular model, GCN [4], simplifies the message passing strategy by using the first-order polynomial, namely, only considering the direct neighbors of nodes. A lot of variants of GCN are then proposed [5, 17].

As graphs in the real world contain abundant nodes, existing graph neural networks primarily focus on the semi-supervised scenario where partial nodes are tagged with labels. MPNN-based GNNs achieve great success by following the assumption of homophily, which assumes that connecting nodes in the graph are similar in features and share the same labels. In this way, beneficial information will be aggregated to nodes and can help models learn better

representations. Most graphs in the real world can satisfy the assumption, while for the particular node in the graph, the situation is not always satisfied. For instance, there exist mistakes when collecting graph data that link nodes of differently labeled nodes. The adversarial attack in graphs is often conducted by connecting nodes of different classes as message passing will amplify noise from different-labeled nodes. Besides, nodes in the boundaries between classes connect to different-labeled nodes in the connected graph. We did simple statistics on benchmark datasets by counting the nodes with different-labeled neighbors. We call the nodes *neighbor inconsistent nodes* (NI-Nodes). The results are shown in Table 1. From the table, we can figure out that NI-Nodes are common even in widely used homophily graphs.

When performing message passing in these nodes, unnecessary information will be aggregated. Consequently, the final node representation will deviate in the wrong direction. Here is a toy example:

In Figure 1, we conducted 1-step aggregation in the sample graph, and different colors indicate different labels. According to the figure, as the node a and node d share the consistent neighbors, respectively, their colors remain the same after aggregation. However, node b and node c change their colors with different-colored neighbors' information propagated, resulting in unreliable final representations. What is worse, the noisy information will influence other nodes with the procedure of iterative aggregation.

Consequently, how to evaluate messages from neighbors becomes crucial for better graph neural networks. The most popular method, GCN, treats every neighbor node equally by mean aggregator without considering noise from the neighborhood. GAT [5] introduces attention scores to evaluate the influence of every neighbor and then reweight messages from the neighborhood. Nevertheless, it still assumes that all neighbors are beneficial for the model training. Some other methods [18, 19] also try to modify the topology structure to help better aggregate information. Besides, some self-supervised learning methods [20–23] try to construct multi-view graphs and apply contrastive learning to alleviating the noise from unreliable neighbors. However, the methods usually train multiple models for different graph views and learn multiple representations for nodes, which is time and space consuming.

In this paper, we focus on different-labeled neighbors in the aggregation of nodes. Instead of modifying models from the structure or neighbor weights, we attempt to consider this problem from the point of labeled nodes as the framework is optimized by the nodes. We argue that considering aggregation in graph neural networks can be seen as the procedure of node feature reconstruction. And we can divide the information captured in aggregation into two parts, node features and context features. Then, we measure the difference between the node feature and the context feature to evaluate the reliability of node representation after aggregation. In particular, we devise a novel metric NC (neighbor consistency) to evaluate the reliability. Furthermore, we propose the method called *Neighbor Consistent Graph Neural Networks* (NC-GNN) to improve the training of graph neural networks by reweighting the influence of labeled nodes. The greater the neighbor consistency is, the more reliable is the node representation after aggregation, which indicates that the node representation can help more in the model training and vice versa. Empirical results for node classification demonstrate the effectiveness of our method. We summarize the main contributions of this paper as follows:

Table 1: Nodes with different-labeled neighbors in the benchmark graphs.

	Nodes	NI-Nodes	Proportion (%)
Cora	2078	932	34.42
Citeseer	3327	1360	40.88
PubMed	19717	8759	35.57

(i) We devise neighbor consistency (NC) to measure the difference between labeled nodes and their neighborhoods. By regarding aggregating information from neighborhoods as node feature reconstruction, NC can evaluate the reliability of labeled nodes after aggregation effectively

(ii) We devise a novel method NC-GNN to promote the training process of graph neural networks. The method can obtain better embeddings from neighbor-consistent nodes by reweighting the influence of labeled nodes according to neighbor consistency scores

(iii) We conduct extensive experiments on node classification, and the results indicate the effectiveness of our method

The remaining part of the paper is organized as follows. Section 2 reviews the related works involving graph neural networks and some modifications of aggregation. In Section 3, we introduce some preliminaries and the framework of graph neural networks. In Section 4, our method is then presented with a detailed description. Extensive experiments are conducted in Section 5 to evaluate the performance of our method. At last, Section 6 concludes the paper with discussions and future works.

2. Related Works

In this section, we briefly review the related works, including graph neural networks and modifications for aggregating neighbor nodes in graph neural networks. Since graph neural network is a very active research area, we only introduce the most relevant models. For more details, we refer readers to some surveys [24, 25].

2.1. Graph Neural Networks. The research of graph neural networks is popular in graph learning. It is aimed at transferring traditional convolutional networks from Euclidean space to graph domain. Graph convolution is first proposed in [26] in graph signal processing, and there are many works to simplify the framework in both spectral and spatial

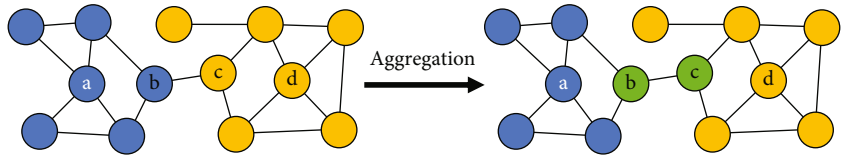


Figure 1: An example of 1-step aggregation in graph neural networks.

domain. For example, [27] introduces the Chebyshev polynomials with orders of K to approximate the eigendecomposition. And Kipf and Welling [4] in GCN simplify the model by using the first-order polynomial, namely, only considering the direct neighbors of nodes. Due to the simpleness and conciseness of GCN, it becomes the baseline and popular in graph learning. Existing graph neural network models usually follow the framework of MPNN (message passing neural network) [16], which aggregates messages from neighbor nodes to update the embeddings of target nodes. For instance, GraphSage [17] applies different strategies to aggregate features from neighbor nodes. GAT [5] evaluates the importance between target nodes and neighbor nodes so that the model can aggregate more related information. GIN [9] develops a simple structure to ensure that the aggregator is injective and the representational power is equal to the power of WL-test. SGCN [28] simplifies GCN through successively removing nonlinearities and collapsing weight matrices between consecutive layers.

The models based on MPNN mostly follow the assumption of homophily, which states that nodes connected by edges are similar and beneficial information can be propagated in the graph. However, the assumption is not always satisfied as there always exists unintentional or intentional noise in real-world graphs. In the following subsection, we will introduce some modifications on aggregating neighbor nodes considering the situation.

2.2. Modifications on Aggregating Neighbor Nodes. As homophily is not always satisfied in the real world, aggregating beneficial neighbors becomes crucial in graph neural networks. To alleviate the influence of different-labeled neighbors, many works are then proposed. For instance, [29] compares the original prediction with the counterfactual prediction calculated by presenting multiple data indicators to assess the trustworthiness of neighbor nodes. Besides, as neighborhood information is preserved in the graph structure, many frameworks are then designed to modify the graph structure so as to conduct aggregation better. Methods like self-enhanced GNN [30] and EGAI [31] add or remove edges based on the predicted neighbor labels learned by the model. Bayesian GCN [32], LDS [18], SimP-GCN [33], and IDGL [34] adopt different strategies to optimize the graph structure and node embeddings simultaneously to make graph structure more suitable for model learning. Some contrasting models try to construct multiviews by modifying neighbor structures [35–37]. In the real-world scenario, some models adopt neighbor aggregation modifications to better fit downstream graph tasks, like modifying graph structure [20, 38] or evaluating the dependencies between nodes [39, 40].

Though the above methods achieve great progress in encoding nodes into better embeddings with modified structure, the modification of graph structure sometimes discards the important interactions between nodes, resulting in information loss.

3. Background

3.1. Notations and Preliminaries. This paper mainly focuses on undirected graphs, but the method can also be used in directed graphs. We present $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ as a graph, where $\mathcal{V}$ consists of the set of nodes in $\mathcal{G}$, with $|\mathcal{V}| = N$. $\mathcal{E}$ is the collection of edges. $\mathcal{X} \in \mathbb{R}^{N \times F}$ denotes node feature matrix, where $\mathbf{x_i} \in \mathbb{R}^F$ represents the attributes of node v_i, and F is the dimension of node features. Adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ is the topological structure of graph $\mathcal{G}$, where $\mathbf{A}_{ij} > 0$ indicates that there is an edge between nodes i and j. Otherwise, $\mathbf{A}_{ij} = 0$.

Given topological structure $\mathbf{A}$ and feature matrix $\mathcal{X}$ as input, our objective is to learn low-dimensional dense node embedding matrix $\mathbf{Z} \in \mathbb{R}^{N \times d}$ with $d \ll F$ without human annotation. The learned node embeddings can well preserve topology and feature information so as to be applied to downstream tasks. In this paper, we focus on semisupervised node classification. $\mathcal{V}_L \subset \mathcal{V}$ is the labeled node set, and we have $|\mathcal{V}_L| \ll |\mathcal{V}|$. $\mathcal{Y} \in \mathbb{R}^{N \times k}$ is the label set for nodes in the graph, $y_i \in \mathbb{R}^k$ is the one-hot label vector of node v_i, and k is the number of classes. Then, we aim to train a classifier $\hat{\mathcal{Y}}_U = g(\cdot)$ by utilizing the learned node embedding $\mathbf{Z}$ as input to predict the labels for unlabeled node set $\mathcal{V}_U = \mathcal{V} - \mathcal{V}_L$.

3.2. Graph Neural Networks. Graph neural networks are a popular class of graph embedding methodologies that model the graph structure and node features to encode representations for nodes in the graph. Existing GNN frameworks mostly learn node representations by aggregating the features of neighbor nodes. The output of the k-th layer of the framework can be generally expressed as

$$h_i^{(k)} = \sigma\left(h_i^{(k-1)}, \mathrm{AGG}\left(h_j^{(k-1)}\right)\right), j \in \mathcal{N}(i), \tag{1}$$

where $h_i^{(k)}$ is the node representation of node v_i at the k-th layer with $h_i^{(0)} = \mathbf{x}_i$ and $\mathcal{N}(i)$ is the direct neighbors of node v_i. $\sigma(\cdot)$ is the nonlinear method to combine the information from the previous layer to update node representations. $\mathrm{AGG}(\cdot)$ is the method to aggregate information from

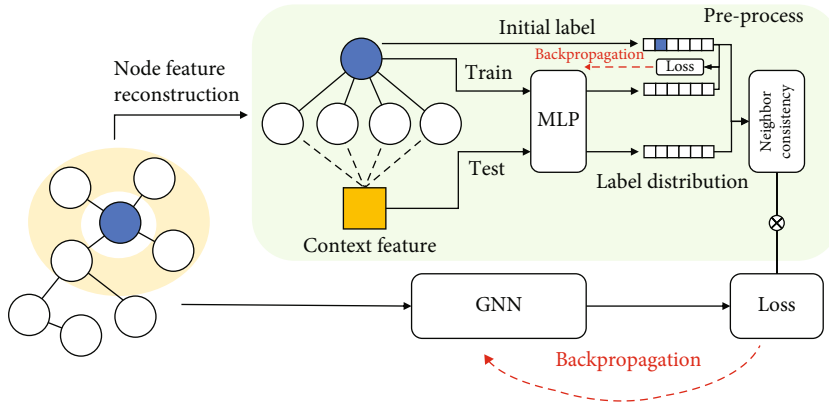


Figure 2: Overview of our proposed NC-GNN. The blue-colored node represents labeled node. We evaluate neighbor consistency of labeled nodes in preprocessing section. After that, we reweight the influence of labeled nodes in the loss function with neighbor consistency scores.

neighbor nodes, which is usually mean, max, sum methods. Different GNN method varies in the formulations of $\sigma(\cdot)$ and AGG$(\cdot)$ methods.

4. Our Approach

We propose a method called Neighbor Consistent Graph Neural Networks, short as NC-GNN, to promote the training of GNNs by evaluating the consistency between nodes and the corresponding neighbors. The overview of the method is shown in Figure 2. The model consists of two components. In the preprocessing procedure, we evaluate neighbor consistency for labeled nodes. After that, with the calculated neighbor consistency scores, the influence of labeled nodes is reweighted in the model training procedure. We will introduce each component in detail as follows.

4.1. Node Feature Reconstruction. Graph neural networks essentially utilize the message passing strategy of aggregating information from neighbor nodes to update node representations. For every node, the desiring situation is that connected neighbor nodes are all similar, namely, the assumption of homophily, so that the final node representation can be more accurate and generalized for node classification. However, the hypothesis is not always satisfied for particular nodes as discussed above. Therefore, the information from neighbor nodes should be evaluated before messaging passing.

Based on the framework of GNNs shown in Section 3.2, the update of the representation for nodes can be divided into node features and context features, which preserve all the neighbor features. As node feature remains unchanged, GNN is trying to transform node representation towards context features. With infinite iterations of aggregation and update, the node representation can be transferred into the mixture feature constructed by neighbors. If node features are blank, the final representation of the node is then decided by the context features thoroughly. Based on the above discussion, we can regard the information aggregation as node feature reconstruction, which represents the nodes with the weighted combination of context features from neighbor nodes and node features.

Therefore, the context features play a critical impact in the final embeddings of nodes after aggregation. In particular, to ensure the final embeddings can well represent the nodes, the context feature constructed by neighbor nodes should be similar to the node features. And the corresponding labels of context features can well capture the information.

In the Euclidian space, we assume a virtual center node exists for every class. Therefore, if neighbor nodes are similar to the target node and share the same label, the nonnegative weighted sum of neighbor features, namely, context features, should always be closer to the class center node than the neighbor node which is furthest away. Otherwise, the label of the representation after aggregation cannot be guaranteed.

Consequently, we devised a simple metric, called neighbor consistency, to better evaluate the consistency between nodes and the corresponding context features by measuring the difference between the labels of nodes and their neighbors.

4.2. Evaluation of Neighbor Consistency. To help GNNs learn better node embeddings, nodes with consistent context representations should be more important. When the consistency is high, the final embedding of the node is representative. Firstly, we calculate the context feature as

$$c_i = \sum \alpha_{i,j} \mathbf{x}_j, j \in \mathcal{N}(i), \tag{2}$$

where $\mathcal{N}(i)$ is the neighbor set of v_i. Actually, the equation can be seen as encoding the neighborhood of v_i to a virtual context node. $\alpha_{i,j}$ is the weight between v_i and v_j. If we focus on the direct neighbors in the graph just as the aggregation in GCN, we can simply set $\alpha_{i,j} = 1/|\mathcal{N}(i)|$. Or we can construct the ego-network of v_i with limited k-order neighbors and then calculate $\alpha_{i,j}$ through Personalized PageRank. In this way, we can reconstruct node features through a wider receptive field.

There exist many methods to measure the difference between features. In this paper, as we focus on labeled nodes, we propose the Multilayer Perception (MLP) to classify context features. In particular, we regard labeled nodes as the training set for the classifier, and then, we can get the

following label distribution for the corresponding context features, namely,

$$\hat{\mathbf{y}}_i^c = \mathrm{MLP}(c_i), \tag{3}$$

where $\hat{\mathbf{y}}_i^c \in \mathbb{R}^k$ is the predicted label distribution for context feature of node v_i and k represents the number of classes.

Compared with other methods like Euclidian distance, MLP$(\cdot)$ trained by labeled nodes can better capture class information among all the training samples, so the MLP$(\cdot)$ is more generalized to calculate robust label distributions for context features.

The learned MLP$(\cdot)$ is overfitting with labeled nodes. So it can well classify the context features. While the model cannot perform well in unlabeled nodes, that is the reason why we do not consider evaluating neighbor consistency for unlabeled nodes in the whole graph.

$\hat{\mathcal{Y}}_L^c$ can then be used to evaluate neighbor consistency by comparing with labels $\mathcal{Y}_L$ of corresponding labeled nodes. When the context feature's predicted label is the same as the corresponding labeled node, we can conclude that the labeled node is neighbor-consistent. The higher confidence in the prediction indicates the greater consistency between the node and the neighborhoods. Otherwise, if the context feature is classified as a different label, the neighbors are inconsistent with the labeled node. In this paper, we utilize prediction confidence to evaluate the consistency between neighbors and nodes. So we define neighbor consistency score (NC) as

$$\mathrm{NC}_i = \begin{cases} \max\left(\left\{\hat{\mathbf{y}}_{ij}^c\right\}\right), & \mathrm{argmax}_j\left(\left\{\hat{\mathbf{y}}_{ij}^c\right\}\right) = l, \mathbf{y}_{il} = 1, \\ -1 * \max\left(\left\{\hat{\mathbf{y}}_{ij}^c\right\}\right), & \text{otherwise}, \end{cases} \tag{4}$$

where max $(\cdot)$ function is used to identify the maximum value in the label distribution. argmax$(\cdot)$ is to figure out the class label of the context feature with the highest probability.

The calculated NC scores can capture the consistency between labeled nodes and their neighbors well, with larger values indicating greater consistency.

4.3. Promoting GNN by Loss Weight Reweighting. In this section, we introduce NC-GNN, a training weight schedule mechanism to promote the training of graph neural networks. As discussed above, graph neural networks are trained through aggregating neighbor features for target nodes to update node representations iteratively, and the model is optimized by calculating the classification loss of labeled nodes. As a result, the labeled nodes with consistent neighborhoods can help more in the training process. Based on the calculation of neighbor consistency, we can figure out the labeled nodes whose neighbors' features may not be helpful in aggregation and even bring extra noise. So we can utilize NC scores to make nodes with consistent neighbors play a more active role in model learning. Specially, we devise a simple method for the calculation of node weights according to neighbor consistency scores,

$$W = \mathrm{softmax}(\mu \cdot \mathrm{NC} + \mathbb{I}(\mathrm{NC}) \cdot b), \tag{5}$$

$$\mathbb{I}(x) = \begin{cases} 1, & x > 0, \\ -1, & x < 0, \end{cases} \tag{6}$$

where $W \in \mathbb{R}^L$ are the weight matrix for labeled nodes and μ and b are the parameters of scaling the neighbor consistency scores and initial weight for nodes, respectively. $\mathbb{I}(\cdot)$ is the indicator function to give the initial weight difference between neighbor-consistent nodes and neighbor-inconsistent nodes. Here we choose softmax$(\cdot)$ to avoid future normalization in calculation.

Then, the training loss for a better graph neural network is computed by the following equations,

$$\hat{\mathcal{Y}} = \mathrm{softmax}(\mathbb{F}(\mathcal{X}, \mathbf{A}, \theta)), \tag{7}$$

$$L_N = -\frac{1}{|\mathcal{L}|}\sum_{i \in \mathcal{L}} w_i \sum_{l=1}^{k} \mathbf{y}_{il} \log \hat{\mathbf{y}}_{il}, \tag{8}$$

where $\mathbb{F}$ denotes any GNN framework, θ is the parameter of $\mathbb{F}$, $\hat{\mathcal{Y}}$ is the GNN output. w_i is the loss weight of labeled node v_i calculated in Equation (8), $\hat{\mathbf{y}}_i$ is the prediction of v_i, and $\mathbf{y}_i$ indicates the original label for node v_i in one-hot embedding. By encouraging positive effects from the aggregation of consistent neighbors which share the same label with target nodes and alleviating the negative effects from the aggregation of neighbors with different labels, our method can promote graph neural networks by reducing noise in model training, so as to represent nodes with robust embeddings.

4.4. Complexity Analysis. In our NC-GNN method, we construct context representations for labeled nodes and predict the corresponding label distribution to evaluate the neighbor consistency between labeled nodes and their neighbors. And the method can be split into two procedures, preprocessing and GNN training.

In the GNN training procedure, we use common GNN frameworks and the time complexity is the same as the frameworks. Here we consider GCN as an example. The time complexity of an L-Layer GCN model is $O(L|\mathbf{A}_0|F + LNF^2)$, where N is the number of nodes, $|\mathbf{A}_0|$ is the number of nonzeros in $\mathbf{A}$, and F is the number of features. In the preprocessing procedure, we first construct the context representations for labeled nodes. For the mean method, the time complexity is $O(d|\mathcal{L}|)$, where d is the average degree of nodes in the graph and $|\mathcal{L}|$ is the number of labeled nodes. For the PPR method, the complexity is $O(d^2|\mathcal{L}|)$. Due to the sparsity of graphs, $d << N$. Then, we use MLP$(\cdot)$ as the classifier to predict the contexts' label distributions. For the L-Layer network, the time complexity is $O(L|\mathcal{L}|F^2)$. And the overall complexity of the procedure is $O(|\mathcal{L}| + L|\mathcal{L}|F^2)$.

For space complexity, GCN needs $O(LF^2)$ memory for storing the weight matrix and $O(LNF)$ for embeddings.

Table 2: Data distribution of benchmark datasets.

	Cora	Citeseer	PubMed	Photo	Computers
Nodes	2078	3327	19717	7487	13381
Edges	5278	4614	44325	119043	245778
Features	1433	3703	500	745	767
Classes	7	6	3	8	10
Training nodes	140	120	60	20 per class	20 per class
Validation nodes	500	500	500	30 per class	30 per class
Test nodes	1000	1000	1000	Rest nodes	Rest nodes

Our method needs additional $O(L|\mathscr{L}|F)$ memory to store the context representations and $O(LF^2)$ memory to store the weight matrix of the classifier.

5. Experiments

In this section, we conduct adequate experiments to validate the effectiveness of our method. We first evaluate whether the calculated neighbor consistency matches the neighbor distribution in the graphs. Then, node classification experiments are conducted to demonstrate the effectiveness of our method. Furthermore, we discuss the relationships between neighbor consistency and the performance of our method and study the parameters' influence on the model.

5.1. Datasets. Following previous works [4, 41], we utilize the widely used Planetoid paper citation datasets(Cora, Citeseer, and PubMed) and the Amazon purchase graphs (Photo and Computers). In the citation datasets, nodes and edges represent documents and citation relations between documents, respectively. Each node is represented by the bag-of-words features extracted from the contents of the document. Each node corresponds to a label with the one-hot encoding of the document category. In Amazon purchase graphs, nodes represent goods on the site, edges indicate that two goods are frequently bought together, node features are bag-of-words encoded product reviews, and class labels are given by the product category. We employ data with *DGL* [42] and *Pytorch-Geometric* [43] module, and the data distribution is shown in Table 2.

5.2. Experimental Settings

5.2.1. Baseline Methods. To evaluate the effectiveness of our method, we compare with the following *state-of-the-art* methods.

(i) *DeepWalk* [44]. It is the typical shallow network embedding model by regarding node as words in documents and utilizing skip-gram models to train embeddings

(ii) *GCN* [4]. It is the baseline of graph neural networks. It generalizes the covolutional operation from deep learning to graph domain, considering aggregating messages from direct neighbors

Table 3: Proportion of NI-Nodes to the predicted nodes using mean method.

	Predicted nodes	NI-Nodes	Percentage
Cora	36	20	55.6%
Citeseer	46	37	80.4%
PubMed	7	6	85.7%
Photo	37	31	83.8%
Computers	67	52	77.6%

Table 4: Proportion of NI-Nodes to the predicted nodes using Personalized PageRank method.

	Predicted nodes	NI-Nodes	Percentage
Cora	28	20	71.4%
Citeseer	38	32	84.2%
PubMed	7	6	85.7%
Photo	34	28	82.4%
Computers	69	58	84.1%

(iii) *GraphSage* [17]. It is extending the mean aggregator of GCN to perform multiaggregation and performing a sampling strategy before aggregation

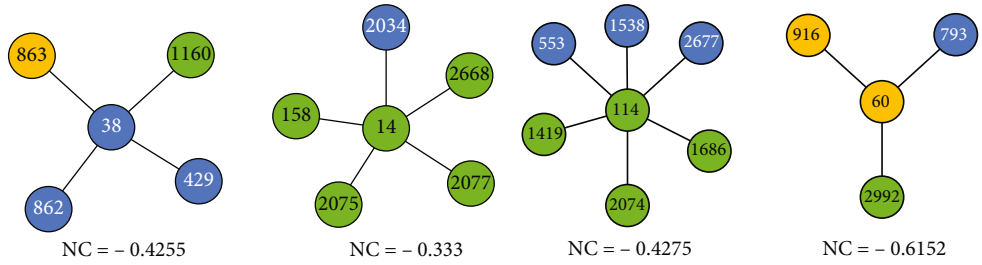
(iv) *GAT* [5]. It is considering weighting the neighbors in aggregation by introducing attention mechanism to GCN and assigning different weights to neighbor nodes according to attention scores

(v) *DropEdge* [45]. It is considering modifying the structure by randomly removing a certain number of edges at each epoch to improve the generalization capacity of GCN

(vi) *SimP-GCN* [33]. It is considering modifying the structure by combining kNN-graph calculated by node features and original graph to preserve node feature similarity with updated structure and improve the homophily in the graph

(vii) *NC-GNN*. This is our method, and we choose the GCN and GAT as our baseline methods

5.2.2. Parameter Settings. In parameter settings, we designed 2-layer graph neural networks with the same hidden layer dimension and the same output dimension simultaneously for every method. For baseline methods like DeepWalk,



(a) Samples in Cora (b) Samples in Citeseer

Figure 3: Sampled predicted low neighbor consistency nodes in datasets. Numbers included are the node index, and different color indicates different class.

GCN, GAT, and DropEdge, we follow the instruction of original codes in Github published by the authors. For GraphSage, we only consider the situation with the mean aggregator, and the model is implemented the same as the authors' guidance. With our methods, we set the parameters of MLP almost the same as GCN, with the same hidden layers, and the same dropout rate. Besides, for the NC-GNN models, we follow most settings the same as the base methods except that we use an early stopping strategy the same as GAT with patience of 100 epochs in NC-GCN.

In data splitting, we follow the same data split as previous works in Planetoid citation datasets. For Amazon copurchasing datasets, as there is no existing split setting, we randomly sample 20 nodes per class as the training set, 30 nodes per class as the validating set, and the rest nodes as the testing set, which is consistent with previous works.

5.3. Neighbor Consistency Evaluation. We first evaluate the neighbor consistency measured by the difference between nodes and the corresponding neighbors. According to the discussion in Section 4.2, we can figure out that when the value of the NC score is negative, the node is more likely to connect to different-labeled neighbor nodes. So we conduct the experiments to determine whether the predicted neighbor-inconsistent nodes are connecting to different-labeled nodes. Besides, we compare the mean and Personalized PageRank method in constructing context features. The results are shown in Tables 3 and 4.

From Tables 3 and 4, we can find out that most nodes with negative NC scores are neighbor-inconsistent, which may bring unnecessary information for aggregation in model training. The results prove that NC scores can well capture the neighbor consistency of nodes. Compared Tables 3 and 4, in most cases, Personalized PageRank method performs better than mean method in finding neighbor-consistent nodes, as the wider receptive field can provide more neighbor information. Therefore, we utilize Personalized PageRank method as the base method for constructing context features in the following experiments.

Visualization. We sampled some predicted nodes by our method in the datasets to observe whether NC scores can figure out neighbor-inconsistent nodes. The results are shown in Figure 3.

From Figure 3, we can conclude that NC scores can discover the neighbor-inconsistent nodes in the graph well. As there exist different-labeled nodes in the neighborhood, the aggregated information can be noise for the central node sometimes. Besides, in the right part of Figure 3(a), we can find that only one noisy node in the neighborhood sometimes can bring a considerable negative impact on aggregation. The results indicate that attention to neighbor consistency is essential to train GNN models.

Table 5: The results of node classification accuracy (%) on the datasets.

Data	Cora	Citeseer	PubMed	Photo	Computers
DeepWalk	67.2	43.2	65.3	-	-
GCN	81.6	70.5	78.7	89.6	76.5
GraphSage	77.4	67.0	76.6	86.5	74.5
GAT	82.6	70.3	77.5	91.3	79.3
DropEdge	79.6	67.6	73.4	87.6	78.2
SimP-GCN	82.5	72.5	**80.9**	89.4	78.2
NC-GCN	83.0	**74.2**	79.3	90.9	83.3
NC-GAT	**83.5**	73.0	79.1	**92.1**	**84.5**

5.4. Node Classification Comparison. To verify the effectiveness of our proposed NC-GNN by reweighting train weights of labeled nodes, we conducted extensive experiments on node classification compared with baselines in benchmark datasets. The results are shown in Table 5. From the table, we can find the following observations:

Our method NC-GNN achieves the best or second-best performance compared with baseline methods in all datasets. The promising results validate the effectiveness of our method which reweighs the labeled nodes with calculated neighbor consistency scores. Compared with NC-GCN, NC-GAT shows fewer improvements compared with corresponding baselines as GAT utilizes an attention mechanism to measure the weights for neighborhoods. The attention scores can alleviate the noise passed from different-labeled neighbors.

DropEdge randomly removes a certain percentage of edges in the graph to improve the generalization performance of graph neural networks, which can be seen as removing different-labeled neighbors randomly. However, the randomicity sometimes discards the important interactions between nodes, resulting in unsatisfying performance.

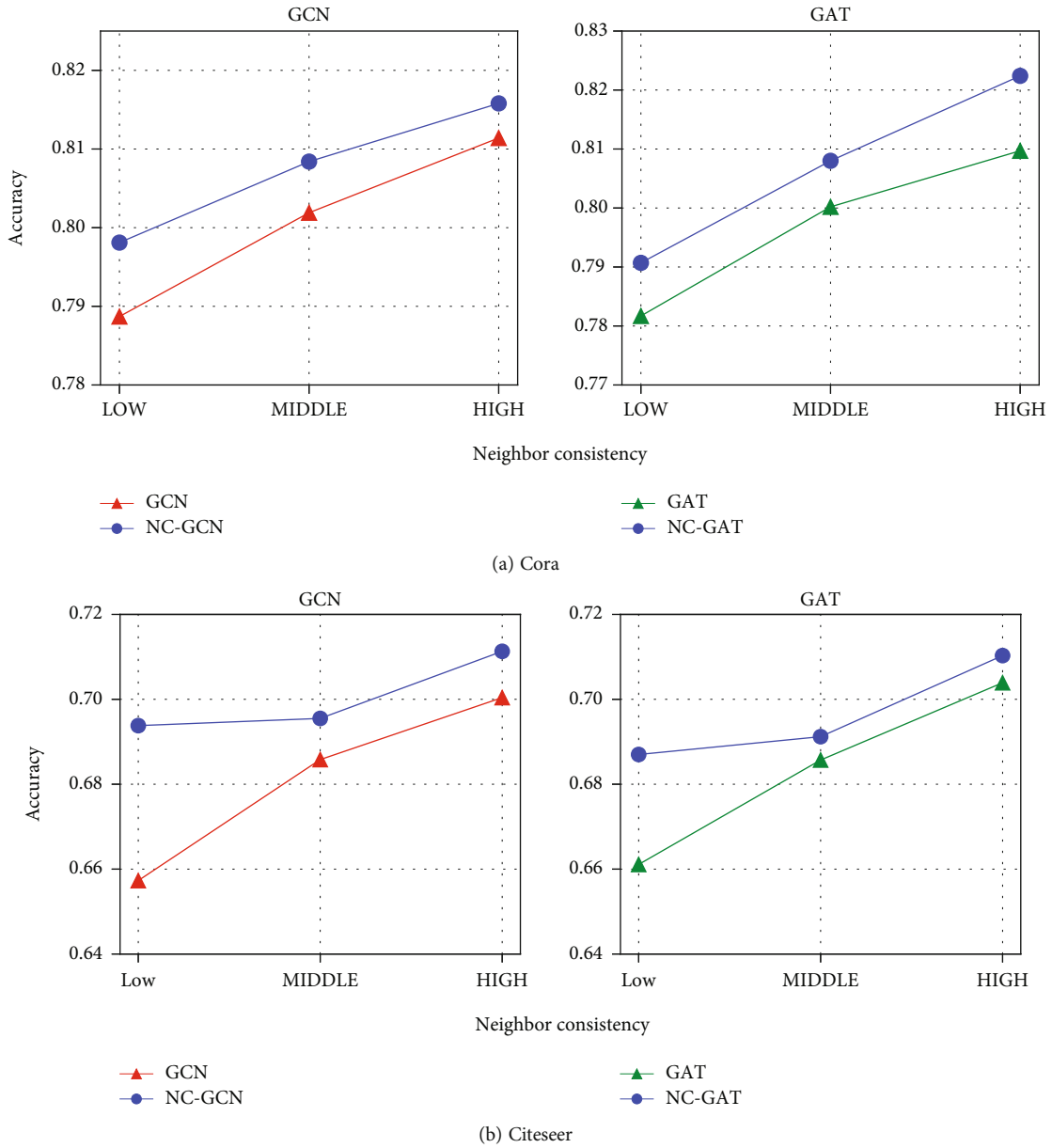


Figure 4: The results of classification accuracy in datasets with different neighbor consistency. Low, middle, and high indicate different probabilities of neighbor-inconsistent nodes in the dataset.

Evaluating the neighbor consistency in our method can figure out the different-labeled neighbors without losing the structure information, which is easier to control and stable. SimP-GCN updates the graph structure by combining with kNN-graph calculated by node features' similarity, thus connecting similar nodes and improving homophily in the graph. However, it ignores the different-labeled neighbors which pass unnecessary information in aggregation.

GAT outperforms other baselines as they can weigh neighbor nodes with attention scores so as to prevent the information aggregated from different-labeled neighbors. However, GAT still assumes that all neighbors are beneficial no matter the neighbors' labels. Therefore, the node remains unreliable after aggregating information from different-labeled neighbors. In contrast, our method improves the model by reducing the impact of unreliable nodes in the training procedure. In this way, we can better capture the beneficial information to train the model. The results also show that our method can learn better node embeddings.

Considering the specific dataset, we can find that our method shows more improvements in Citeseer than Cora compared with baseline methods. According to Table 4, we can conclude that we find more neighbor-inconsistent nodes in Citeseer; thus, our method contributes more to reducing the negative impact of different-labeled neighbors and enhancing the performance of the framework. As for PubMed, NC-GNN only finds 6 neighbor-inconsistent nodes, so our method can bring few improvements on baselines.

As for the baseline methods, GCN model performs better than DeepWalk as graph convolution can capture node

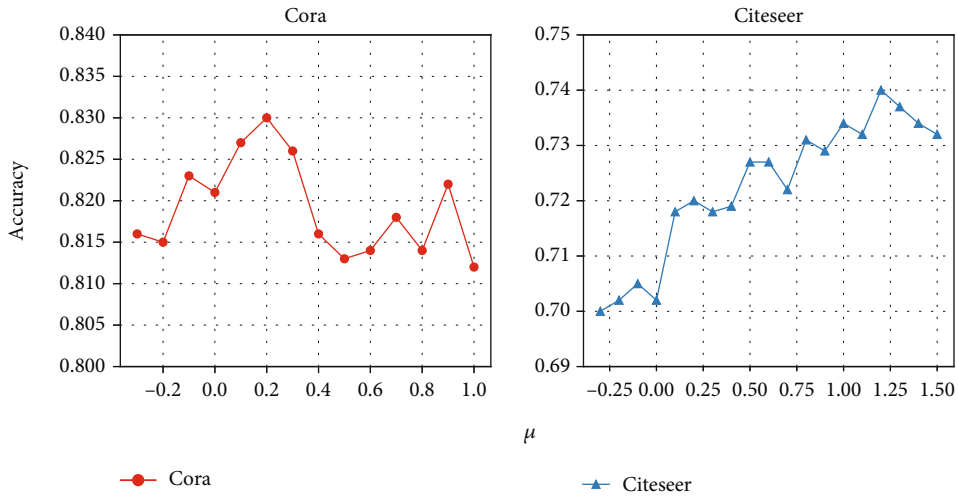


Figure 5: Classification accuracy with different μ in Cora and Citeseer.

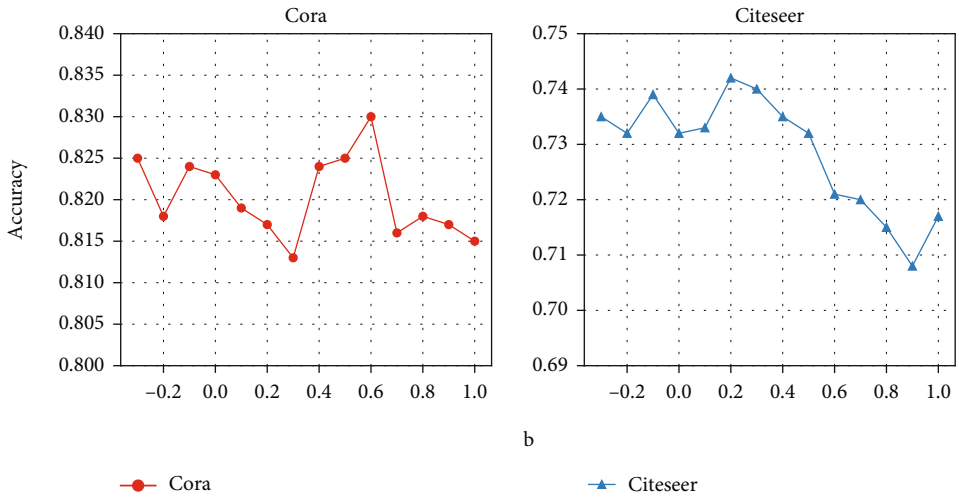


Figure 6: Classification accuracy with different b in Cora and Citeseer.

Table 6: Running time(s) of the models on the datasets.

Data	Cora	Citeseer	PubMed	Photo	Computers
GCN	1.71	1.87	6.17	3.76	7.75
NC-GCN	2.62(+53.2%)	2.98(+53.0%)	7.04(14.1%)	6.57(+48.1%)	9.58(+23.6%)
GAT	19.50	26.91	89.78	84.14	144.67
NC-GAT	20.39(+4.6%)	28.01(+4.1%)	90.67(+1.0%)	85.95(+2.2%)	146.48(+1.2%)

features and topology information simultaneously. GAT outperforms GCN in some cases as GAT introduces attention to graph convolution to decide the more important neighborhoods. The results are consistent with those in previous works.

5.5. The Influence of Neighbor Consistency. Our method focuses on improving GNN frameworks with the neighbor consistency of labeled nodes. Therefore, the neighbor consistency of the labeled node set plays a considerable influence on the performance of our method. We conduct experiments to discuss the influence of different neighbor consistency probabilities. In particular, we randomly sampled three labeled node sets of 20%, 50%, and 80% neighbor-consistent nodes with the same setting of 20 labeled nodes per class in the previous works. And the probabilities correspond to high, middle, and low neighbor consistency, respectively. We conduct the experiments on Cora and Citeseer with our variants of GCN and GAT. The results are shown in Figure 4.

In Figure 4, we can figure out that as the neighbor consistency grows, all models perform increasingly better, which indicates that neighbor consistency is crucial for model performance. In most situations, our method leads

to more improvements on baseline methods when neighbor consistency of labeled nodes is low, while in the high neighbor consistency situation, the gap between our method and baseline decreases as there are fewer neighbor-inconsistent nodes.

5.6. Parameter Study. In this section, we consider the parameters in calculating training weights for labeled nodes in the model, including μ for scaling the node neighbor consistency scores and b for giving the initial training weights. To verify the effects of the corresponding parameters, the experiment results of NC-GCN in Cora and Citeseer datasets with different parameter settings are presented in Figures 5 and 6.

μ tries to scale the computed neighbor consistency scores. A larger μ will highlight the neighbor consistency differences, while a smaller value tries to mitigate the neighbor consistency differences. From Figure 5, we can figure out that NC-GCN performs best in both datasets when the value of μ is positive, which indicates that the labeled nodes with consistent neighbors contribute more in the training process. We can conclude that valuing the neighbor consistency is beneficial for node classification with graph neural networks.

b assigns an initial training weight to the labeled nodes with the corresponding sign given by the indicator method, thus distinguishing between the neighbor-consistent nodes and the neighbor-inconsistent nodes. A positive b indicates that the nodes are vital in the training process, while a negative value weakens the influence. Figure 6 presents that the model achieves the best results in both datasets when b is positive. We can conclude that labeled nodes with consistent neighbors should be more noticed than neighbor-inconsistent nodes in the training process. Besides, the performance of NC-GCN degrades when the value of b is too large, as the model can hardly capture the information of different neighbor consistency scores. So we recommend a small b in model training.

5.7. Running Time Analysis. We report the running time of our method NC-GCN and NC-GAT compared with corresponding baselines in Table 6 (the experiments are conducted in the machine with an Intel(R) Core(TM) i9-10900 K @ 3.70 GHz CPU and a Nvidia GeForce GTX 3090 GPU).

The table shows that our method brings some time cost when the baseline model is simple, whereas there is little running time gain when the baseline is complex, such as GAT. The observation indicates that our method can be applied to massive graphs when choosing a suitable baseline framework. Besides, the time consumption of our method increases less compared with baseline methods when the graph is larger and contains more nodes. This mainly lies in that only partial nodes are labeled in real-world graph tasks.

6. Conclusion

Existing graph neural network methods mostly follow the assumption of homophily, which states that connected nodes are similar and share the same labels. However, the assumption is not always satisfied in real-world graphs. In this paper, we focus on labeled nodes and try to evaluate the consistency between the nodes and corresponding neighborhoods. In particular, we regard information aggregation in graph neural networks as node feature reconstruction and represent the neighborhoods as context features. Then, we design a novel metric neighbor consistency to evaluate the difference between node features and the corresponding context features so as to measure the reliability of labeled nodes after aggregation. Furthermore, we propose the method called *Neighbor Consistent Graph Neural Networks* (NC-GNN) to promote the training of graph neural networks by reweighting the influence of labeled nodes. In this way, the labeled nodes with consistent neighborhoods can contribute more to the model training. Extensive experiments are conducted on benchmark datasets, and outstanding performance indicates the effectiveness of our method.

In the future, we aim to extend the neighbor consistency from labeled nodes to all nodes in the graph to improve aggregation in graph neural networks.

Data Availability

The data used in this paper is available upon the request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by the National Key Research and Development Program of China (Grant No. 2018YFB1403400), the National Natural Science Foundation of China (Grant No. 61876080), the Key Research and Development Program of Jiangsu (Grant No. BE2019105), and the Collaborative Innovation Center of Novel Software Technology and Industrialization at Nanjing University.

References

[1] W. Fan, Y. Ma, Q. Li et al., "Graph neural networks for social recommendation," in *Proceedings of the World Wide Web Conference*, pp. 417–426, San Francisco, California, USA, 2019.

[2] C. Gao, X. Wang, X. He, and Y. Li, "Graph neural networks for recommender system," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 1623–1625, Virtual Event / Tempe, Arizona, USA, 2022.

[3] W. Jiang, "Graph-based deep learning for communication networks: a survey," *Computer Communications*, vol. 185, pp. 40–54, 2022.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2017.

[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, Virtual, OpenReview.net, 2018.

[6] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 5165–5175, Montreal, Canada, 2018.

[7] M. Zhang, P. Li, Y. Xia, K. Wang, and L. Jin, "Revisiting graph neural networks for link prediction,," 2020, https://arxiv.org/abs/2010.16103.

[8] H. Gao and S. Ji, "Graph u-nets," in *Proceedings of the International Conference on Machine Learning*, pp. 2083–2092, Long Beach, California, USA, 2019.

[9] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2019.

[10] H. Yuan and S. Ji, "Structpool: structured graph pooling via conditional random fields," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2020.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: simplifying and powering graph convolution network for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, Xi'an, China, 2020.

[12] J. Sun, Y. Zhang, W. Guo et al., "Neighbor interaction aware graph convolution networks for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1289–1298, Xi'an, China, 2020.

[13] A. Benamira, B. Devillers, E. Lesot, A. K. Ray, M. Saadi, and F. D. Malliaros, "Semi-supervised learning and graph neural networks for fake news detection," in *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 568-569, Marriott Downtown, Vancouver, Canada, 2019.

[14] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 7370–7377, Honolulu, Hawaii, USA, 2019.

[15] S. Arora, "A survey on graph neural networks for knowledge graph completion," 2020, https://arxiv.org/abs/2007.12374.

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the International Conference on Machine Learning*, pp. 1263–1272, Sydney, Australia, 2017.

[17] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the International Conference on Neural Information Processing Systems*, pp. 1025–1035, Long Beach, California, USA, 2017.

[18] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *Proceedings of the International Conference on Machine Learning*, pp. 1972–1982, Long Beach, California, USA, 2019.

[19] C. Zheng, B. Zong, W. Cheng et al., "Robust graph representation learning via neural sparsification," in *Proceedings of the International Conference on Machine Learning*, pp. 11458–11468, Virtual, 2020.

[20] J. Wu, X. Wang, F. Feng et al., "Self-supervised graph learning for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 726–735, Virtual, 2021.

[21] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. X. Huang, "Hypergraph contrastive collaborative filtering," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, Madrid, Spain, 2022.

[22] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and N. Q. V. Hung, "Are graph augmentations necessary? Simple graph contrastive learning for recommendation," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, Madrid, Spain, 2022.

[23] J. Zhang, M. Gao, J. Yu, L. Guo, J. Li, and H. Yin, "Double-scale self-supervised hypergraph learning for group recommendation," in *Proceedings of the ACM International Conference on Information & Knowledge Management*, pp. 2557–2567, Queensland, Australia, 2021.

[24] W. L. Hamilton, "Graph representation learning, Synthesis Lectures on Artifical Intelligence and Machine," *Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[25] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2022.

[26] J. Bruna, W. Zaremba, A. D. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2014, https://arxiv.org/abs/1312.6203.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3837–3845, Barcelona, Spain, 2016.

[28] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the International Conference on Machine Learning*, pp. 6861–6871, Long Beach, California, USA, 2019.

[29] F. Feng, W. Huang, X. Xin, X. He, and T.-S. Chua, "Should graph convolution trust neighbors? A simple causal inference method," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1208–1218, Paris, France, 2021.

[30] H. Yang, X. Yan, X. Dai, Y. Chen, and J. Cheng, "Self-enhanced gnn: improving graph neural networks using model outputs," 2020, https://arxiv.org/abs/2002.07518.

[31] C. Liu, J. Wu, W. Liu, and W. Hu, "Enhancing graph neural networks by a high-quality aggregation of beneficial information," *Neural Networks*, vol. 142, pp. 20–33, 2021.

[32] Y. Zhang, S. Pal, M. Coates, and D. Ustebay, "Bayesian graph convolutional neural networks for semi-supervised classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 5829–5836, Honolulu, Hawaii, USA, 2019.

[33] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 148–156, Virtual, 2021.

[34] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: better and robust node embeddings," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19314–19326, 2020.

[35] X. Chen, Y. Zhang, I. Tsang, and Y. Pan, "Learning robust node representations on graphs," 2020, https://arxiv.org/abs/2008.11416.

[36] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5812–5823, 2020.

[37] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference*, pp. 2069–2080, Virtual, 2021.

[38] M. Luo, X. Chang, L. Nie, Y. Yang, A. G. Hauptmann, and Q. Zheng, "An adaptive semisupervised feature analysis for video semantic recognition," *IEEE Transactions on Cybernetics*, vol. 48, no. 2, pp. 648–660, 2018.

[39] D. Zhang, L. Yao, K. Chen, S. Wang, X. Chang, and Y. Liu, "Making sense of spatio-temporal preserving representations for EEG-based human intention recognition," *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3033–3044, 2020.

[40] K. Chen, L. Yao, D. Zhang, X. Wang, X. Chang, and F. Nie, "A semisupervised recurrent convolutional attention model for human activity recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 5, pp. 1747–1756, 2020.

[41] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, https://arxiv.org/abs/1811.05868.

[42] M. Wang, D. Zheng, Z. Ye et al., "Deep graph library: a graph-centric, highly-performant package for graph neural networks," 2019, https://arxiv.org/abs/1909.01315.

[43] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, New York, USA, 2014.

[45] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: towards deep graph convolutional networks on node classification," in *Proceedings of the International Conference on Learning Representations*, Virtual, OpenReview.net, 2020.