BEYOND MESSAGE PASSING PARADIGM: TRAINING GRAPH DATA WITH CONSISTENCY CONSTRAINTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent years have witnessed great success in handling graph-related tasks with Graph Neural Networks (GNNs). However, most existing GNNs are based on powerful message passing to guide feature aggregation among neighbors. Despite their success, there still exist three weaknesses that limit their capacity to train graph data: weak generalization with severely limited labeled data, poor robustness to label noise and structure disturbation, and high computation and memory burden for keeping the entire graph. In this paper, we propose a simple yet effective *Graph Consistency Learning* (GCL) framework, which is based purely on multilayer perceptrons, where structure information is only implicitly incorporated as prior knowledge in the computation of supervision signals but does not explicitly involve the forward. Specifically, the GCL framework is optimized with three well-designed consistency constraints: neighborhood consistency, label consistency, and class-center consistency. More importantly, we provide theoretical analysis on the connections between message passing and consistency constraints. Extensive experiments show that GCL produces truly encouraging performance with better generalization and robustness compared with other leading methods.

1 INTRODUCTION

Recently, the emerging Graph Neural Networks (GNNs) have demonstrated their powerful capability to handle the task of semi-supervised node classification: inferring unknown node labels by using the graph structure and node features with partially known node labels. Despite all these successes, most existing GNNs are mainly based on the message passing paradigm, which consists of two key computations for each node at every layer: (1) AGGREGATE operation: aggregating messages from its neighborhood, and (2) UPDATE operation: updating node representation from its representation in the previous layer and the aggregated messages. Due to the overemphasis on the graph structure, message-passing-based GNNs face several challenges: (1) poor robustness to label noise and structure perturbation; (2) weak generalization when labeled data is severely scarce; (3) not flexible enough, as it requires a large memory space for keeping the entire graph and does not support batch-style training. There has been a large number of work proposed to address the above problems, which leads to some newly born research topics, such as graph adversarial, graph denoising, graph self-supervised learning, large-scale graph learning, etc. However, most of these methods use Graph Convolutional Networks (GCN) (Kipf & Welling, 2016) as the bottleneck encoder, i.e., still following the paradigm of message passing. Therefore, two crucial questions here are: *(1) Is message passing a must for training graph data? (2) Can we remove the message passing and get comparable or even better performance than existing GNNs with a pure MLP-based architecture?*

The answer to *Question (1)* is "Yes", and there have been a number of random-walk-based and factorization-based graph embedding methods that do not require the message passing. For example, DeepWalk (Perozzi et al., 2014) achieves graph embedding directly by deploying a truncated random walk. Instead, LINE (Tang et al., 2015) introduces structural information into the graph regularization term, capturing both first-order and second-order proximities in the graph structure by optimizing a carefully designed objective function. However, since GCN was proposed, its simple design and huge performance advantages have surpassed previous graph embedding methods, making message passing a dominant paradigm for training graph data. Therefore, how to solve the *Problem (2)* still remains to be explored. Recently, there are some attempts to combine sophisticated technologies in computer vision with MLPs to break the paradigm of message passing. For exam-

ple, Graph-MLP (Hu et al., 2021) designs a neighborhood contrastive loss to bridge the gap between GNNs and MLPs by utilizing the adjacency information implicitly. Instead, LinkDist (Luo et al., 2021) directly distills self-knowledge from connected node pairs into MLPs without the need for aggregating messages. Despite their great success, these methods (1) still cannot match the state-of-the-art message-passing-based GNNs in terms of classification performance; and (2) are designed with intuition, and lack sufficient theoretical foundations behind the design.

In this paper, we propose a simple yet effective *Graph Consistency Learning* (GCL) framework to train graph data with three well-designed consistency constraints: neighborhood consistency, label consistency, and class-center consistency. The proposed GCL framework is a pure MLP-based architecture with message passing modules completely removed, and structure information is only implicitly used as a priori knowledge to guide the computation of supervision signals. More importantly, we provide theoretical analysis on the connections between message passing and consistency constraints. Furthermore, an edge sampling strategy is proposed to support batch-style training without the need for keeping the entire graph, which helps to address the limitations of the classical stochastic gradient descent and improves both the effectiveness and efficiency of the inference.

Our contributions are summarized as: (1) break the message passing paradigm, where structure information is only used for loss calculation; (2) propose three consistency constraints to train graph data and achieve performance beyond existing methods; (3) provide theoretical analysis on the connections between message passing and consistency constraints; (4) demonstrate the advantages of GCL over existing methods in terms of generalizability, robustness, and computational efficiency.

2 Related Work

2.1 Graph Neural Networks

Recent years have witnessed the great success of Message-Passing-based Graph Neural Networks (GNNs) in handling graph-related tasks. There are two categories of GNNs: Spectral GNNs and Spatial GNNs. The spectral GNNs define convolution kernels in the spectral domain based on the graph signal processing theory. For example, ChebyNet (Defferrard et al., 2016) uses the polynomial of the Laplacian matrix as the convolution kernel to perform message passing, and GCN is a first-order approximation of ChebyNet with a self-loop mechanism. The spatial GNNs focus on the design of aggregation functions directly. For example, GraphSAGE (Hamilton et al., 2017) employs a generalized induction framework to efficiently generate node embeddings for previously unseen data by aggregating known node features. GAT (Kipf & Welling, 2016), on the other hand, adopts the attention mechanism to calculate the coefficients of neighbors for better information aggregation. We refer interested readers to the recent survey (Wu et al., 2020) for more variants of GNN architectures, such as SGC (Wu et al., 2019), APPNP (Klicpera et al., 2018) and DAGNN (Liu et al., 2020). However, the above methods all share the de facto design that graph structure information is explicitly utilized for message passing to aggregate features.

2.2 Graph Embedding

The mainstream graph embedding algorithms can be divided into three categories: (1) factorization-based, (2) random-walk-based, and (3) deep-learning-based. The factorization-based algorithms, such as LLE (Roweis & Saul, 2000) and Laplacian Eigenmaps (Belkin & Niyogi, 2001), represent the connections between nodes in the form of a matrix and factorize this matrix to obtain the embeddings. Instead, the random-walk-based algorithms, such as DeepWalk (Perozzi et al., 2014) and node2vec (Grover & Leskovec, 2016), apply random walks on graphs to directly obtain node representations. With the development of deep learning techniques, deep-learning-based methods have shown better performance than other two types of methods on various graph-related tasks, and *Graph Neural Networks are just one special category of deep-learning-based methods*. For example, Graph-MLP (Hu et al., 2021) designs a neighborhood contrastive loss to bridge the gap between GNNs and MLPs by implicitly utilizing the adjacency information. Instead, LinkDist (Luo et al., 2021) directly distills self-knowledge from connected node pairs into MLPs without the need to aggregate messages. Despite their great success, they (1) still cannot match the state-of-the-art message-passing-based GNNs in terms of performance, and (2) are designed with intuition, and lack sufficient theoretical foundations behind the design. In this paper, we *mainly focus on*

deep-learning-based graph embedding and explore can we remove message passing but still obtain comparable performance with a pure MLP-based architecture?

3 METHODOLOGY

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of N nodes with features $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N] \in \mathbb{R}^{N \times d}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. Each node $v_i \in \mathcal{V}$ is associated with a d-dimensional features vector $\mathbf{x}_i$, and each edge $e_{i,j} \in \mathcal{E}$ denotes a connection between node v_i and node v_j. The graph structure is denoted by an adjacency matrix $\mathbf{A} \in [0,1]^{N \times N}$ with $\mathbf{A}_{i,j} = 1$ if $e_{i,j} \in \mathcal{E}$ and $\mathbf{A}_{i,j} = 0$ if $e_{i,j} \notin \mathcal{E}$. Node classification is a typical node-level task where only a subset of node $\mathcal{V}_L$ with corresponding labels $\mathcal{Y}_L$ are known, and we denote the labeled set as $\mathcal{D}_L = (\mathcal{V}_L, \mathcal{Y}_L)$ and unlabeled set as $\mathcal{D}_U = (\mathcal{V}_U, \mathcal{Y}_U)$, where $\mathcal{V}_U = \mathcal{V} \backslash \mathcal{V}_L$. The task of node classification aims to learn a mapping $\Phi : \mathcal{V} \rightarrow \mathcal{Y}$ on labeled data $\mathcal{D}_L$, so that it can be used to infer the labels $\mathcal{Y}_U$ of unlabeled data $\mathcal{D}_U$.

In this section, we introduce a simple yet effective *Graph Consistency Learning* (GCL) framework, which is based on a pure MLP-based architecture, with each layer composed of a linear transformation, an activation function , a batch normalization, and a dropout function, formulated as:

$$\mathbf{H}^{(l+1)} = \text{Dropout}\left(BN\left(\sigma\left(\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)\right)\right), \quad \mathbf{H}^{(0)} = \mathbf{X}; \qquad 0 \leq l \leq L-1 \tag{1}$$

where $\sigma = \text{ReLu}(\cdot)$ denotes an activation function, $BN(\cdot)$ denotes the batch normalization, and $\text{Dropout}(\cdot)$ is the dropout function. $\mathbf{W}^{(0)} \in \mathbb{R}^{d \times F}$ and $\mathbf{W}^{(l)} \in \mathbb{R}^{F \times F}$ $(1 \leq l \leq L-1)$ are layer-specific weight matrices with the hidden dimension F. The proposed GCL framework trains graph data with three well-designed consistency constraints: neighborhood consistency, label consistency, and class-center consistency. An overview of the proposed GCL framework is shown in Fig. 1.

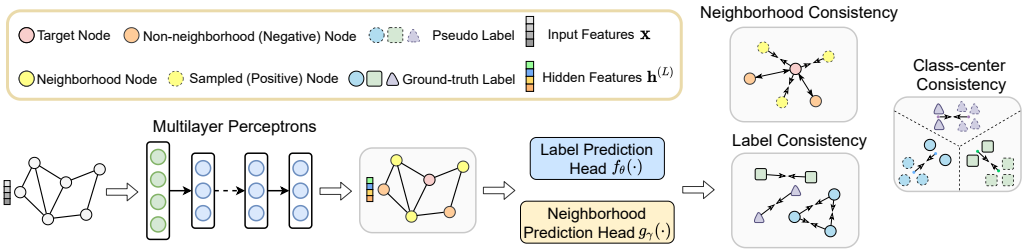

Figure 1: Illustration of the proposed GCL framework, consisting of multilayer perceptrons, label prediction head $f_\theta(\cdot)$, neighborhood prediction head $g_\gamma(\cdot)$, which is mainly optimized by three consistency constraints: Neighborhood Consistency, Label Consistency, and Class-center Lonsistency.

3.1 CONSISTENCY CONSTRAINTS

3.1.1 NEIGHBORHOOD CONSISTENCY

The smoothness assumption indicates that connected nodes should be similar, while disconnected nodes should be far away, which aligns with the idea behind GCNs. With such motivation, we propose a neighborhood consistency loss that enables the MLP-based model to learn the connectivity of graph nodes without explicit message passing modules. To this end, we first define two functions: label prediction head $\mathbf{y}_i = f_\theta(\mathbf{h}_i^{(L)}) \in \mathbb{R}^C$ and neighborhood prediction head $\mathbf{z}_i = g_\gamma(\mathbf{h}_i^{(L)}) \in \mathbb{R}^C$, where C is the number of categories. Specifically, we sample between each node v_i and its neighbors $\mathcal{N}_i$ with *learnable sampling coefficients*, and then take sampled nodes instead of neighboring nodes as positive samples, while other non-neighborhood nodes are considered as negative samples. Finally, the neighborhood consistency loss is defined as

$$\mathcal{L}_n = \sum_{i=1}^{N} \Big(\sum_{j \in \mathcal{N}_i} \left\| \mathbf{y}_i - g_\gamma\left(\beta_{i,j}\mathbf{h}_j^{(L)} + (1-\beta_{i,j})\mathbf{h}_i^{(L)}\right) \right\|_2^2 - \alpha \mathbb{E}_{v_k \sim P_k(v)} \left\| \mathbf{y}_i - g_\gamma(\mathbf{h}_k^{(L)}) \right\|_2^2 \Big) \tag{2}$$

where $\beta_{i,j} = \text{sigmoid}\left(\mathbf{a}^T \left[\mathbf{h}_i^{(L)} \| \mathbf{h}_j^{(L)}\right]\right)$ is defined as learnable sampling coefficients with the shared attention weight $\mathbf{a}$. Besides, $P_k(v)$ denotes the distribution that generates negative samples, and for

each node v_i, we specify $P_k(v_i) = \frac{d_i}{|\mathcal{E}|}$. It can be seen that the neighborhood consistency loss $\mathcal{L}_n$ encourages positive samples to be closer and pushes negative samples away. In this paper, we specify two independent prediction heads $f_\theta(\cdot)$ and $g_\gamma(\cdot)$ for the two contrastive node pairs, which is mainly inspired by the recent success experience of visual self-supervised learning (He et al., 2020; Chen et al., 2020). In the experimental section, we have also demonstrated the benefits of using two independent prediction heads compared to one shared prediction head.

3.1.2 LABEL CONSISTENCY

The label consistency is proposed to encourage intra-class compactness and inter-class distinguishability. Specifically, the query node v_i and key node v_j are encoded by two prediction heads $f_\theta(\cdot)$ and $g_\gamma(\cdot)$ to obtain two label distributions $\widetilde{\mathbf{y}}_i = \mathrm{softmax}\big(f_\theta(\mathbf{h}_i^{(L)})\big)$ and $\widetilde{\mathbf{z}}_j = \mathrm{softmax}\big(g_\gamma(\mathbf{h}_j^{(L)})\big)$, respectively. If node v_i and node v_j have the same label, they are considered to be the positive pair. Then, the label consistency loss can be formulated as

$$\mathcal{L}_l = \sum_{i=1}^{N} \sum_{j \neq i} \mathbb{I}_{c_j = c_i} \cdot \left\| \widetilde{\mathbf{y}}_i - \widetilde{\mathbf{z}}_j \right\|_2^2 \tag{3}$$

where c_i and c_j are the labels of node v_i and v_j, and $\mathbb{I}_{c_j=c_i}$ is an indicator function to determine whether the label of node v_i is the same as node v_j. Inspired by the success of graph contrastive learning (Ren et al., 2021; Qiu et al., 2020; Wu et al., 2021), the label consistency can essentially be considered as a self-supervised auxiliary task that encourages intra-class compactness with more distinguishable inter-lcass boundaries, which helps to improve model robustness.

3.1.3 CLASS-CENTER CONSISTENCY

The basic idea behind this consistency is that the class center of the ground-truth labels and pseudo labels are expected to be mapped nearby, which helps to mitigate the negative impact of false pseudo labels. The class-center consistency loss can be defined as

$$\mathcal{L}_c = \sum_{i=0}^{C-1} \left\| \mathbf{C}_i(\mathbf{Y}) - \widetilde{\mathbf{C}}_i(\mathbf{Y}) \right\|_2^2 \tag{4}$$

where $\mathbf{C}_i(\mathbf{Y}) = \frac{1}{|\mathcal{C}_i|} \sum_{j \in \mathcal{C}_i} \mathbf{y}_j$ and $\widetilde{\mathbf{C}}_i(\mathbf{Y}) = \frac{1}{|\widetilde{\mathcal{C}}_i|} \sum_{j \in \widetilde{\mathcal{C}}_i} \mathbf{y}_j$ represents the centroid of node features with the ground-truth label $\mathcal{C}_i$ and pseudo label $\widetilde{\mathcal{C}}_i$, respectively. The benefits of center consistency have been shown in (Yang et al., 2021) and will be supported by our experiment in Sec. 4.3 as well.

3.2 CONNECTIONS BETWEEN MESSAGE PASSING AND NEIGHBORHOOD CONSISTENCY

In this section, we theoretically analyze the connections between message passing and neighborhood consistency. Without loss of generality, we take classical GCN as an example to analyze the properties of message passing, which can be easily generalized to other message-passing-based GNNs.

Theorem 1 *The neighborhood consistency loss is lower bounded by structure smoothing imposed by message passing (taking GCN as an example), i.e. minimizing the neighborhood consistency loss is equivalent to performing message passing among neighbors, if the following conditions hold:*

(a) $\alpha = 0$*, that is to remove the negative sampling module;*

(b) $\mathcal{N}_i = d, \forall v_i \in \mathcal{V}$*, which means that all nodes in the graph have the same degree* d*;*

(c) $f_\theta(\cdot) = g_\gamma(\cdot)$ *and* $\beta_{i,j} = 1, \forall i, j \in \mathcal{V}$.

We first start from the analysis of how node representations are learned in GCN. According to the definition of GCN, we can formulate l-th $(0 \leq l \leq L-1)$ layer of GCN as follows

$$\mathbf{H}^{(l+1)} = \sigma\left(\widetilde{\mathbf{D}}^{-\frac{1}{2}} \widetilde{\mathbf{A}} \widetilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{(l)} \mathbf{W}^{(l)}\right) \tag{5}$$

where $\widetilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the adjacency matrix with added self-connections, $\mathbf{I}_N$ is the identity matrix for N nodes, $\widetilde{\mathbf{D}}$ is a diagonal degree matrix with $\widetilde{\mathbf{D}}_{i,i} = \sum_j \widetilde{\mathbf{A}}_{i,j}$. GCN has been proved to be a special form of Laplacian smoothing (Li et al., 2018). As the GCN model goes deeper with more convolution layers, the representations in Eq. (5) reaches a termination condition $\mathbf{H}_t$ as

$$\mathbf{H}^t = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^t\mathbf{W}^t\right) \tag{6}$$

where $\mathbf{W}^t$ is the weight matrix on the last layer of GCN. Following the implementation made by SGC (Wu et al., 2019), nonlinearities and collapsing weight matrices between consecutive layers can be ignored for the sake of simplification. Thus, an approximate solution of Eq. (6) is written as

$$\mathbf{H}^t = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^t \tag{7}$$

More specifically, for each node v_i, the approximation of the corresponding final representation is

$$\mathbf{h}_i^t = \sum_{j\in\mathcal{N}_i} \frac{1}{\sqrt{(d_i+1)\,(d_j+1)}}\mathbf{h}_j^t + \frac{1}{d_i+1}\mathbf{h}_i^t \tag{8}$$

where $d_i = |\mathcal{N}_i|$ is the degree of node v_i, from Eq. (8) which we have

$$\mathbf{h}_i^t = \sum_{j\in\mathcal{N}_i} \frac{1}{d_i}\sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t \tag{9}$$

Based on the above analysis, the objective of the approximate GCN model is two-fold: (1) classify nodes with partially known labels; and (2) model the structural smoothness of the graph convolution. Specifically, the structural smoothing loss l_s imposed by message passing can be defined as

$$l_s = \sum_{i=1}^{N} Dis\Big(\mathbf{h}_i^t, \sum_{j\in\mathcal{N}_i} \frac{1}{d_i}\sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t\Big) \tag{10}$$

where $Dis(\cdot)$ is a metric function. Here, we adopt the Mean Square Error (MSE), and obtain

$$l_s = \sum_{i=1}^{N}\Big\|\mathbf{h}_i^t - \sum_{j\in\mathcal{N}_i}\frac{1}{d_i}\sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t\Big\|_2^2 = \sum_{i=1}^{N}\frac{1}{d_i}\Big\|\sum_{j\in\mathcal{N}_i}(\mathbf{h}_i^t - \sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t)\Big\|_2^2 \tag{11}$$

According to the power-mean inequality $\left(\frac{\sum_{i=1}^n a_i^\beta}{n}\right)^{\frac{1}{\beta}} \le \left(\frac{\sum_{i=1}^n a_i^\alpha}{n}\right)^{\frac{1}{\alpha}}$ ($\beta = 1, \alpha = 2$), we have

$$l_s = \sum_{i=1}^{N}\frac{1}{d_i}\Big\|\sum_{j\in\mathcal{N}_i}(\mathbf{h}_i^t - \sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t)\Big\|_2^2 \le \sum_{i=1}^{N}\frac{1}{d_i}\cdot d_i\sum_{j\in\mathcal{N}_i}\Big\|(\mathbf{h}_i^t - \sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t)\Big\|_2^2 \tag{12}$$

With *condition (b)*, we have $\sqrt{\frac{d_i+1}{d_j+1}} = 1$. Moreover, let's define $\mathbf{y}_i = f_\theta(\mathbf{h}_i^{(L)}) = \mathbf{h}_i^t$, and follwing *condition (c)*, we have $g_\gamma\big(\beta_{i,j}\mathbf{h}_j^{(L)} + (1-\beta_{i,j})\mathbf{h}_i^{(L)}\big) = g_\gamma(\mathbf{h}_j^{(L)}) = f_\theta(\mathbf{h}_j^{(L)}) = \mathbf{h}_j^t = \sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t$. Finally, according to *condition (a)*, i.e., removing the negative samples, we can rewrite Eq. (12) as

$$l_s \le \sum_{i=1}^{N}\frac{1}{d_i}\cdot d_i\sum_{j\in\mathcal{N}_i}\Big\|(\mathbf{h}_i^t - \sqrt{\frac{d_i+1}{d_j+1}}\mathbf{h}_j^t)\Big\|_2^2 = \sum_{i=1}^{N}\sum_{j\in\mathcal{N}_i}\big\|(\mathbf{y}_i - g_\gamma\big(\beta_{i,j}\mathbf{h}_j^{(L)} + (1-\beta_{i,j})\mathbf{h}_i^{(L)}\big))\big\|_2^2 = \mathcal{L}_n$$

Remark 1 In practical applications, real-world graphs usually have different node degrees, so *condition (b)* may be an overly stringent assumption. Recalling Eq. (12), the role of node degree can be approximated as a weighting on neighboring nodes $\mathcal{N}_i$. Therefore, if we modify *condition (c)* by setting $g_\gamma(\cdot) \neq f_\theta(\cdot)$ and sampling with learnable sampling coefficients $\{\beta\}_{i,j=1}^N$, it enables modeling the effect of node degree on the representations of neighboring nodes $\mathcal{N}_i$, which in turn helps mitigate the impact of degree inhomogeneity. In the experiment part, we have demonstrated the benefits of both (1) using two independent prediction heads and (2) training with learnable sampling coefficients. Moreover, while *condition (a)* theoretically helps to ensure equivariant neighborhood consistency with message passing, it may limit the expressive power of the model. Extensive recent work on visual representation learning (Chen et al., 2020) suggests that negative samples help to learn more distinguishable class boundaries, which is also supported by our experiments in Sec. 4.3.

3.3 EDGE SAMPLING FOR BATCH-STYLE TRAINING

Optimizing objective Eq. (2)(3) is computationally expensive, as it performs the summation over the entire set of nodes, i.e, requiring a large memory space for keeping the entire graph. To address this problem, we adopt the edge sampling strategy (Mikolov et al., 2013; Tang et al., 2015) for batch-style training. More specifically, we first sample a mini-batch of edges from the entire edge set $\mathcal{E}$ to construct a mini-batch $\mathcal{E}_b \in \mathcal{E}$. Then we randomly sample a negative node v_k for each edge $e_{i,j}$ in $\mathcal{E}_b$, which generates two virtual negative edges $e_{i,k}$ and $e_{j,k}$, and thus constructs a new triple set, denoted as $\mathcal{T}_b = \{(i,j,k)|e_{i,j} \in \mathcal{E}_b, v_k \sim P_k(v)\}$. Finally, we can rewrite Eq. (2) as

$$\mathcal{L}_n^b = \frac{1}{B}\sum_{b=1}^{B}\sum_{(i,j,k)\in\mathcal{T}_b}\Big(\big\|\mathbf{y}_i - \widehat{\mathbf{z}}_j\big\|_2^2 + \big\|\mathbf{y}_j - \widehat{\mathbf{z}}_i\big\|_2^2 - \alpha\big(\big\|\mathbf{y}_i - \mathbf{z}_k\big\|_2^2 + \big\|\mathbf{y}_j - \mathbf{z}_k\big\|_2^2\big)\Big) \tag{13}$$

where B is the batch size, $\widehat{\mathbf{z}}_j = g_\gamma\big(\beta_{i,j}\mathbf{h}_j^{(L)} + (1-\beta_{i,j})\mathbf{h}_i^{(L)}\big)$, and $\widehat{\mathbf{z}}_i = g_\gamma\big(\beta_{j,i}\mathbf{h}_i^{(L)} + (1-\beta_{j,i})\mathbf{h}_j^{(L)}\big)$. In a similar way, we can rewrite Eq. (3) in a mini-batch from as

$$\mathcal{L}_l^b = \frac{1}{B}\sum_{b=1}^{B}\sum_{i\in\mathcal{V}_b}\sum_{j\in\mathcal{V}_b, j\neq i}\mathbb{I}_{c_j=c_i}\cdot\big\|\widetilde{\mathbf{y}}_i - \widetilde{\mathbf{z}}_j\big\|_2^2 \tag{14}$$

where $\mathcal{V}_b$ is all sampled nodes in $\mathcal{E}_b$, that is, $\mathcal{V}_b = \{v_i, v_j|e_{i,j} \in \mathcal{E}_b\}$. Moreover, we can formulate the cross-entropy loss on the labeled set $\mathcal{D}_L$ as a classification loss, as follows

$$\mathcal{L}_{cla}^b = \frac{1}{B}\sum_{b=1}^{B}\sum_{i\in\mathcal{V}_L, i\in\mathcal{V}_b}\sum_{j=0}^{C-1} s_{i,j}\log\widehat{\mathbf{y}}_{i,j} \tag{15}$$

where $\widehat{\mathbf{y}}_i = \text{softmax}(\mathbf{y}_i) \in \mathbb{R}^C$, and $s_{i,j} = 1$ if the label of node v_i is c_j, otherwise $s_{i,j} = 0$. Finally, the total loss for training GCL in a mini-batch is defined as:

$$\mathcal{L}_{total} = \mathcal{L}_n^b + \mathcal{L}_l^b + \mathcal{L}_{cal}^b \tag{16}$$

The pseudo-code of the proposed GCL framework is summarized in Algorithm. 1.

Algorithm 1 Algorithm for the proposed *Graph Consistency Learning* (GCL) framework

Input: Feature Matrix: $\mathbf{X}$; Edge Set: $\mathcal{E}$; Number of Batches: B; Number of Epochs: E.
Output: Predicted Labels $\mathcal{Y}_U$, model parameters $\{\mathbf{W}^l\}_{l=0}^{L-1}$ and label prediction head $f_\theta(\cdot)$.

```
Initialize parameter matrices {W^l}_{l=0}^{L-1} and two prediction heads f_θ(·), g_γ(·).
for epoch ∈ {0,1,···,E−1} do
    for batch ∈ {0,1,···,B−1} do
        Sample a mini-batch of edges E_B from E as well as corresponding node set V_B and triple set T_B;
        Compute loss L_n^B, L_l^B and L_cla^B by Eq. (13)(14)(15) and sum up as total loss L_total by Eq. (16);
        Update parameters {W^l}_{l=0}^{L-1}, f_θ(·), and g_γ(·) by back propagation with loss L_total.
    end for
    Compute the class-center consistency loss L_c on the entire node set V by Eq. (4);
    Update parameters {W^l}_{l=0}^{L-1} and f_θ(·) by back propagation with loss L_c.
end for
Predict labels Y_U for those unlabeled nodes V_U.
return Predicted labels Y_U, model parameters {W^l}_{l=0}^{L-1} and label prediction head f_θ(·).
```

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUPS

The experiments include the following five aspects: (1) node classification performance compared to existing methods on various real-world graph datasets; (2) generalization to severely limited labeled data; (3) robustness to label noise and structure disturbation; (4) ablation study and analysis on some of its components and consistency losses; and (5) inference time or running efficiency.

Datasets. The experiments are conducted on five widely used real-world datasets, namely Cora, Citeseer, Actor, Coauthor-CS, and Coauthor-Phy. For each dataset, we randomly select 20 nodes per class to construct a training set, 500 nodes for validation, and 1000 nodes for testing.

Baseline. We consider the following seven classical baselines: MLP, GCN, GAT, GraphSAGE, SGC, APPNP, and DAGNN. Besides, we compare GCL with two state-of-the-art graph embedding methods, including Graph-MLP and LinkDist, both based on pure MLP architectures. Moreover, we use the same data split on each dataset to provide a fair comparison among different models.

Hyperparameters. The following hyperparameters are set the same for all datasets: Adam optimizer with learning rate lr = 0.01 and weight decay $decay$ = 5e-4; Epoch E = 200; Layer number L = 2; Loss weights α = 1.0. The other dataset-specific hyperparameters are determined by an AutoML toolkit NNI with the hyperparameter search sapces as: hidden dimension $F = \{128, 256, 512, 1024\}$; batch size $B = \{256, 512, 1024, 4096\}$. Each set of experiments is run five times with different random seeds, and the average performance are reported as metrics.

4.2 Results and Analysis

Performance for Node Classification. Table. 1 summarizes the mode properties, i.e., whether the adjacency matrix $\mathbf{A}$ is available (or involved) in the forward propagation, as well as the model performance. It can be seen that while Graph-MLP and LinkDist can achieve comparable performance with GCN on a few datasets, they still lag far behind the state-of-the-art message-passing-based GNN model - DAGNN, and cannot even match the performance of GraphSAGE and GAT. Instead, the GCL framework consistently achieves the best overall performance on all five datasets.

Table 1: Classification accuracy $\pm$ std (%) on five real-world datasets. The "Available Data" refers to data that involves the forward propagation. The best results are marked by **bold** on each dataset.

Method	Available Data	Cora	Citeseer	Actor	Coauthor-CS	Coauthor-Physics
MLP	$\boldsymbol{X}$,$\boldsymbol{A}$	61.86±0.43	59.76±0.51	21.18±0.64	83.12±0.53	86.24±0.66
GCN	$\boldsymbol{X}$,$\boldsymbol{A}$	81.28±0.42	71.06±0.44	24.84±0.56	88.66±0.48	92.14±0.34
GAT	$\boldsymbol{X}$,$\boldsymbol{A}$	83.02±0.45	72.56±0.51	26.28±0.45	89.28±0.63	92.40±0.52
GraphSAGE	$\boldsymbol{X}$,$\boldsymbol{A}$	82.22±0.80	71.22±0.58	26.54±0.70	89.18±0.45	91.54±0.54
SGC	$\boldsymbol{X}$,$\boldsymbol{A}$	80.88±0.47	71.84±0.72	25.24±0.55	88.56±0.60	90.92±0.62
APPNP	$\boldsymbol{X}$,$\boldsymbol{A}$	83.28±0.33	71.74±0.27	27.82±1.02	89.72±0.59	92.54±0.59
DAGNN	$\boldsymbol{X}$,$\boldsymbol{A}$	84.30±0.51	73.14±0.62	28.98±0.86	90.20±0.61	93.02±0.72
Graph-MLP	$\boldsymbol{X}$	81.45±0.52	72.87±0.70	25.40±0.49	89.80±0.68	91.85±0.49
LinkDist	$\boldsymbol{X}$	76.70±0.47	65.19±0.55	23.96±0.65	89.56±0.58	92.36±0.70
GCL (ours)	$\boldsymbol{X}$	**84.80±0.42**	**73.72±0.67**	**31.86±0.72**	**91.22±0.41**	**94.46±0.56**

Performance with Severely Limited Labels. The performance on Cora and Citeseer datasets with 1, 3, 5, and 10 training labels per class are shown in Table. 2. When only a limited number of labels are provided, GCL outperforms other baselines at various label rates. Moreover, the performance of GCL is greatly improved compared to other methods when there is only 1 label per class.

Table 2: Classification accuracy $\pm$ std (%) with severely limited labeled nodes.

Method	Cora				Citeseer			
	1 label	3 labels	5 labels	10 labels	1 label	3 labels	5 labels	10 labels
GCN	43.64±0.79	62.96±0.78	74.03±1.03	74.98±0.80	30.75±0.75	55.42±0.77	62.40±0.64	66.82±0.57
GAT	51.60±0.60	67.15±0.70	76.35±0.95	77.94±0.83	43.66±0.77	58.14±0.64	63.36±0.66	68.56±0.81
GraphSAGE	47.57±0.65	66.70±0.62	74.52±0.79	76.55±0.68	38.24±0.81	56.07±0.59	63.67±0.70	67.47±0.84
SGC	45.15±1.08	63.84±1.22	73.31±1.10	75.58±0.98	35.05±0.92	54.03±0.88	59.40±0.76	65.30±0.90
APPNP	49.53±0.54	70.20±0.71	77.95±0.88	79.19±0.75	44.87±0.67	61.10±0.85	64.16±0.58	69.03±0.71
DAGNN	**66.71±0.78**	72.66±0.59	78.50±0.93	80.66±0.67	59.06±0.83	63.26±0.72	67.84±0.84	71.76±0.73
GCL (ours)	65.14±0.57	**73.78±0.41**	**79.52±0.90**	**81.62±0.60**	**59.82±0.64**	**68.94±0.62**	**71.58±0.46**	**72.74±0.50**

Performance with Corrupted Structures. The classification performance under different structure perturbation ratios r is reported in Table. 3, where the corrupted structures are obtained by randomly removing and adding $r \cdot |\mathcal{E}|$ edges for training. Experiments are conducted on the Cora and Citeseer datasets by varying structure perturbation ratios r as $\{5\%, 10\%, 20\%, 30\%\}$. It can be seen that our model is more robust than other methods under various structure perturbation ratios, especially

under severe structure perturbations, e.g., $r = 20\%$ or 30%. For example, when $r = 30\%$, GCL outperforms GAT by 3.42% and 4.97% on the Cora and Citeseer datasets, respectively.

Table 3: Classification accuracy ± std (%) with different structure perturbation ratios.

Method	Cora				Citeseer			
	5%	10%	20%	30%	5%	10%	20%	30%
GCN	76.01±0.55	74.19±0.96	68.69±0.67	63.82±0.54	69.82±0.62	64.84±0.69	62.87±0.93	60.51±0.73
GAT	77.77±0.74	75.01±0.97	69.62±0.51	64.76±0.74	68.27±0.47	63.35±0.89	61.81±0.96	58.57±1.23
GraphSAGE	77.35±0.64	74.72±0.69	69.02±0.50	64.14±0.94	68.98±0.58	63.38±0.67	62.80±0.66	59.54±0.77
SGC	74.80±0.87	73.84±0.83	67.52±0.67	63.77±0.98	66.57±0.56	61.50±0.64	60.76±0.71	57.00±0.98
APPNP	77.92±0.73	74.36±0.71	70.02±0.93	64.90±1.05	67.42±0.44	63.88±0.93	61.56±1.08	58.32±0.69
DAGNN	78.96±0.58	75.12±0.85	70.41±0.84	65.74±0.82	68.73±0.82	64.74±0.65	61.92±0.91	58.96±1.19
GCL (ours)	**80.68±0.52**	**76.42±0.98**	**72.28±0.48**	**68.18±0.99**	**72.16±0.42**	**69.02±0.60**	**66.10±0.90**	**63.54±0.76**

Performance with Noisy Labels. The performance with noisy labels is reported in Table. 4 at various noise ratios $r \in \{5\%, 10\%, 20\%, 30\%\}$ for two types of label noise: symmetric and asymmetric. The symmetric noise means that label c_i $(0 \leq i \leq C-1)$ of each training sample changes independently with probability $\frac{r}{|C|-1}$ to another class c_j $(j \neq i)$, but with probability $1-r$ preserved as label c_i; the asymmetric noise means that label c_i flips independently with probability r to another (fixed) class c_j $(j = (i+1)\%C)$, but with probability $1-r$ preserved as label c_i. It can be seen that our model is more robust than other methods with various label noise types and ratios, especially with the asymmetric noise and severe noise ratios. For example, with $r = 30\%$ asymmetric noise, GCL outperforms DAGNN by 3.61% and 5.99% on the Cora and Citeseer datasets, respectively.

Table 4: Classification accuracy ± std (%) with different label noise ratios.

Dataset	Flipping-Rate	GCN	GAT	GraphSAGE	SGC	APPNP	DAGNN	GCL (ours)
Cora	symmetric 20%	77.77±0.62	79.75±0.92	78.38±0.68	74.82±0.82	79.53±0.56	80.76±0.59	**81.50±0.80**
	symmetric 40%	69.39±0.70	72.69±0.78	71.33±0.81	68.43±1.08	74.38±0.76	75.69±0.90	**76.54±0.98**
	symmetric 60%	52.17±0.93	55.16±0.81	53.99±0.89	49.04±0.90	58.87±1.04	63.10±0.88	**65.86±1.15**
	asymmetric 20%	71.97±0.97	73.63±0.83	73.58±1.05	70.30±0.77	74.53±0.69	76.50±0.97	**78.74±0.77**
	asymmetric 40%	64.07±0.58	64.24±0.78	63.86±0.68	62.41±0.80	65.99±0.69	67.18±0.66	**68.76±0.67**
	asymmetric 60%	38.47±0.95	39.38±0.99	39.49±0.78	37.02±0.82	40.39±1.01	42.61±0.81	**46.22±0.80**
Citeseer	symmetric 20%	66.91±0.58	67.61±0.59	67.34±0.43	65.19±0.62	68.20±0.48	71.25±0.61	**72.54±0.38**
	symmetric 40%	61.65±0.59	63.88±0.46	62.21±0.58	57.66±0.67	65.61±0.58	69.32±0.77	**71.64±0.60**
	symmetric 60%	54.83±0.63	55.26±0.90	54.20±0.58	53.63±0.70	55.84±0.56	59.36±0.77	**63.38±0.59**
	asymmetric 20%	65.38±0.89	66.62±0.85	66.52±0.70	64.50±1.07	68.17±0.96	68.61±0.89	**71.90±0.95**
	asymmetric 40%	55.70±1.07	56.42±0.81	56.60±0.99	53.91±0.91	57.63±0.79	60.39±0.86	**65.38±0.67**
	asymmetric 60%	41.90±0.98	43.70±1.15	42.65±0.58	41.61±0.76	45.15±0.91	46.05±0.87	**52.04±1.06**

Inference Time. With the removal of message passing, the inference time complexity of GCL can be reduced from $\mathcal{O}(|\mathcal{V}|dF + |\mathcal{E}|F)$ to $\mathcal{O}(|\mathcal{V}|dF)$ compared to GCN, where d and F are the input and hidden dimensions. The running time averaged over 30 sets of runs is reported in Table. 5, where all methods use $L = 2$ layers (except $L = 1$ for SGC) and hidden dimension $F = 16$. Besides, all baselines are implemented based on the standard implementation in the DGL library (Wang et al., 2019). While the inference speed of SGC is the fastest, it involves only one layer of linear transformation. In a fair comparison (without considering SGC and marking it as gray), GCL achieves the fastest inference speed on all datasets compared to other message-passing-base GNNs.

Table 5: Comparison of the inference time (ms) for various methods.

Method	Cora	Citeseer	Actor	Coauthor-CS	Coauthor-Phy
GCN	22.41	23.37	23.01	40.43	66.51
GAT	29.65	34.91	32.77	53.19	84.26
GraphSAGE	13.89	13.10	13.03	15.86	22.85
SGC	4.63	4.21	3.86	3.52	3.16
APPNP	54.04	51.91	59.21	85.57	129.38
DAGNN	45.03	55.53	47.04	47.47	56.17
GCL (ours)	**5.22**	**5.08**	**4.77**	**8.64**	**14.04**

4.3 ABLATION STUDY

Effects of Three Consistency Constraints. This evaluates the effectiveness of three consistency constraints through four sets of experiments: the model without (A) Neighborhood Consistency (*w/o* $\mathcal{L}_n^b$); (B) Label Consistency (*w/o* $\mathcal{L}_l^b$); (C) Class-center Consistency (*w/o* $\mathcal{L}_c$), and (D) the full model. After analyzing the results in Fig. 2, we can conclude: (1) The neighborhood consistency is *the most important factor* for excellent performance, the lack of which leads to unsatisfactory classification. For example, the removal of the neighborhood consistency will lead to a sharp deterioration in performance, e.g., 6.1% and 8.7% on the Coauthor-CS and Coauthor-Phy datasets. Moreover, both Label Consistency and Class-center Consistency contribute to improving performance. More importantly, applying these two constraints together can further improve performance on top of each.

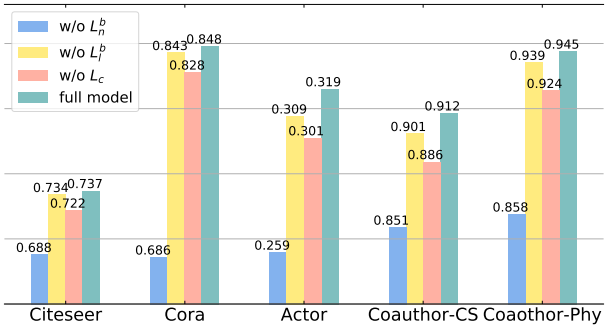

Figure 2: Ablation study on three consistency constraints.

Analysis on Negative Samples, Prediction Heads, and Sampling Coefficients. (1) *Effects of Negative Samples.* The removal of negative samples, i.e., setting α to 0, will lead to performance drops as shown in Table. 6, e.g., 2.2% and 3.2% on the Coauthor-CS and Coauthor-Phy datasets, because contrasting with negative samples helps to push away those possible non-similar samples and enhances the inter-class distinguishability. (2) *Two independent prediction heads* ***vs.*** *One shared prediction head.* Training with one shared prediction head, i.e., $f_\theta(\cdot) = g_\gamma(\cdot)$, leads to huge performance degradation, which indicates mapping neighboring nodes separately using two independent heads helps to better model the relationship between nodes, somewhat similar to learning multiple prediction heads separately for different data augmentations in visual self-supervised learning (You et al., 2021). (3) *Effects of Learnable Sampling Coefficients.* Compared with directly taking neighboring nodes as negative samples, i.e., setting $\beta_{i,j} = 1$, the use of learnable sampling coefficients draws on the success of Mixup (Zhang et al., 2017) and helps to learn distinguishable inter-class boundaries, which explains why setting $\beta_{i,j} = 1$ causes severe performance drops in Table. 6.

Table 6: Ablation study on three key model components.

Scheme	Cora	Citeseer	Actor	Coauthor-CS	Coauthor-Phy
GCL	84.8	73.7	31.9	91.2	94.5
+ $\alpha = 0$	83.1(↓1.7)	71.5(↓2.2)	29.5(↓2.4)	89.0(↓2.2)	91.3(↓3.2)
+ $f_\theta(\cdot)$=$g_\gamma(\cdot)$	82.7(↓2.1)	73.1(↓0.6)	29.1(↓2.8)	88.2(↓3.0)	90.7(↓3.8)
+ $\beta_{i,j} = 1$	83.5(↓1.3)	72.9(↓0.8)	29.9(↓2.0)	89.3(↓1.9)	92.6(↓1.9)

5 CONCLUSIONS

In this paper, we propose a simple yet effective *Graph Consistency Learning* (GCL) framework to train graph data with three well-designed consistency constraints. The GCL framework is based purely on multilayer perceptrons, where structural information is only implicitly incorporated as prior knowledge in the computation of supervision signals, but does not explicitly involve the forward propagation. More importantly, we prove theoretically that minimizing the neighborhood consistency loss is equivalent to performing message passing under certain conditions. The GCL framework demonstrates that message passing is not a must to achieve excellent classification performance and is expected to replace the current dominant GCN-style models as a new paradigm.

REFERENCES

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pp. 585–591, 2001.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Yang Hu, Haoxuan You, Zhecan Wang, Zhicheng Wang, Erjin Zhou, and Yue Gao. Graph-mlp: Node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020.

Yi Luo, Aiguo Chen, Ke Yan, and Ling Tian. Distilling self-knowledge from contrastive links to classify graph nodes without passing messages. *arXiv preprint arXiv:2106.08541*, 2021.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1150–1160, 2020.

Yuxiang Ren, Jiyang Bai, and Jiawei Zhang. Label contrastive coding based graph neural network for graph classification. *arXiv preprint arXiv:2101.05486*, 2021.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Lirong Wu, Haitao Lin, Zhangyang Gao, Cheng Tan, Stan Li, et al. Self-supervised on graphs: Contrastive, generative, or predictive. *arXiv preprint arXiv:2105.07342*, 2021.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

Xu Yang, Cheng Deng, Zhiyuan Dang, Kun Wei, and Junchi Yan. Selfsagcn: Self-supervised semantic alignment for graph convolution network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16775–16784, 2021.

Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. *arXiv preprint arXiv:2106.07594*, 2021.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.