

# Distance-wise Graph Contrastive Learning

Deli Chen<sup>1,2</sup>, Yanyai Lin<sup>2</sup>, Lei Li<sup>1,2</sup>, Xuancheng Ren<sup>1</sup>, Peng Li<sup>2</sup>, Jie Zhou<sup>2</sup>, Xu Sun<sup>1</sup>

<sup>1</sup>MOE Key Lab of Computational Linguistics, School of EECS, Peking University

<sup>2</sup>Pattern Recognition Center, WeChat AI, Tencent Inc., China

{chendeli, lilei\_nlp, renxc, xusun}@pku.edu.cn,  
{yankailin,patrickpli,withtomzhou}@tencent.com

## Abstract

Contrastive learning (CL) has proven highly effective in graph-based semi-supervised learning (SSL), since it can efficiently supplement the limited task information from the annotated nodes in graph. However, existing graph CL (GCL) studies ignore the uneven distribution of task information across graph caused by the graph topology and the selection of annotated nodes. They apply CL to the whole graph evenly, which results in an **incongruous** combination of CL and graph learning. To address this issue, we propose to apply CL in the graph learning adaptively by taking the received task information of each node into consideration. Firstly, we introduce Group PageRank to measure the node information gain from graph and find that CL mainly works for nodes that are topologically far away from the labeled nodes. We then propose our Distance-wise Graph Contrastive Learning (**DwGCL**) method from two views: (1) From the global view of the task information distribution across the graph, we enhance the CL effect on nodes that are topologically far away from labeled nodes; (2) From the personal view of each node's received information, we measure the relative distance between nodes and then we adapt the sampling strategy of GCL accordingly. Extensive experiments on five benchmark graph datasets show that DwGCL can bring a clear improvement over previous GCL methods. Our analysis on eight graph neural network with various types of architecture and three different annotation settings further demonstrates the generalizability of DwGCL.

## 1 Introduction

Graph Neural Networks (GNNs) can effectively utilize the relation information of the nodes contained in the graph topology and have shown powerful modeling capabilities [Yang *et al.*, 2016] in graph-based semi-supervised learning (SSL). Vanilla GNN training [Kipf and Welling, 2017] calculates loss only from the limited labeled nodes, ignoring the information contained in the large amount of unlabeled nodes. To address this problem, recent studies [Hassani and Khasahmadi, 2020;

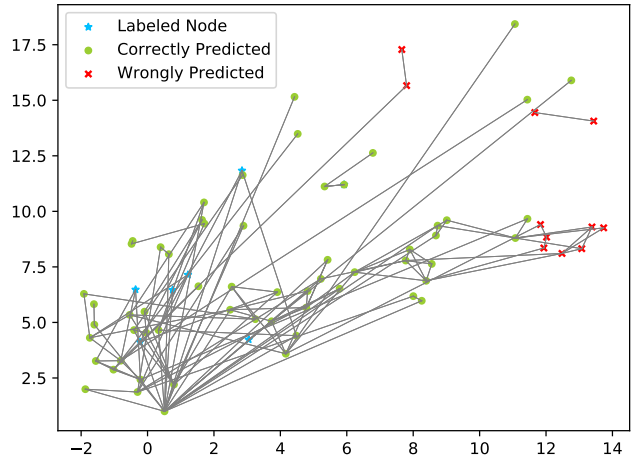


Figure 1: The t-SNE visualization of a sub-graph from the CORA dataset. We display the prediction results of GCN for nodes with the same class. We can find that nodes that are topologically far away from the labeled nodes are more likely to be misclassified, indicating that the GNN inference ability decays when distance from the target node to the labeled nodes increases.

Zhu *et al.*, 2020a; Zhu *et al.*, 2020b; You *et al.*, 2020; Wan *et al.*, 2020; Feng *et al.*, 2020] further introduce Contrastive Learning (CL) [Xie *et al.*, 2019; He *et al.*, 2020] into the graph-based SSL. Generally, graph contrastive learning (GCL) conducts random corruption on graph nodes or topology and learns the node representations by maximizing feature consistency under different graph corruptions. This could serve as a supplement for the task information from the limited annotated nodes.

However, we notice that there is still a common issue in the existing GCL methods: they ignore the **uneven** distribution of task information across graph nodes. When learning GNNs, the task information is propagated from the labeled nodes to the unlabeled ones along the graph edges, leading to an **uneven task information distribution in the graph**. Therefore, the modeling ability embedded in the graph structure decays as the distance to the labeled nodes increases, as shown in Figure 1. All the existing GCL methods ignore this uneven distribution and apply CL evenly in the whole graph, causing an **incongruous** combination of CL and graph learning.

To remedy this issue, we propose to apply GCL more adaptively by taking the task information distribution into consideration. Firstly, we design an indicator named **Group PageRank** to measure the effectiveness of task information that nodes receive from graph. Group PageRank algorithm starts a random walk from the group of nodes with the same annotated task information (e.g., labeled nodes of the same class), and we regard the random walk stop probability for each node as the influence intensity from this group. We calculate Group PageRank for each class independently, and the effectiveness of the node's received information is measured by both the significance and confusion of different class's values. With the support of Group PageRank, we find that **the effect of GCL mainly comes from the improvement of the distant nodes**.

Motivated by this observation, we propose the Distance-wise Graph Contrastive Learning (**DwGCL**) method to enhance the existing GCL methods from two aspects: (1) from the global view of the task information distribution across the graph nodes, we further enhance the GCL effect on nodes that are topologically far away from the information source (labeled nodes), by designing a novel distance-wise graph corruption mechanism and loss schedule; and (2) from the personal view of each node's received task information, we measure the relative distance between nodes by mainly considering the KL divergence of their Group PageRank distribution; then we sample the closest nodes as positive pairs and semi-difficult nodes as negative pairs for each node individually.

Extensive experiments on five benchmark datasets have shown that our DwGCL method can bring larger improvements to GNN training than existing GCL methods without distance-wise mechanisms. Further analysis in various graph SSL scenarios, including eight different types of GNN architecture and three different annotation sizes, demonstrates the generalizability of the proposed DwGCL.

## 2 Inspections of Graph Contrastive Learning

Owing to their promising performance of graph contrastive learning methods, we attempt to probe its working mechanism in this section. We firstly introduce the Group PageRank algorithm which is designed to measure the node received information from the labeled nodes, and then study the joint effect of graph and contrastive learning.

### 2.1 Task Formalization

We select the fundamental semi-supervised node classification task [Kipf and Welling, 2017; Yang *et al.*, 2016; Hamilton *et al.*, 2017] to conduct experiments. Formally, given an undirected graph with node embedding  $\mathbf{X} \in \mathbb{R}^{n \times h}$  and adjacency matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  ( $n$  denotes the node size of the graph and  $h$  denotes the dimension of the initial node embedding), the node classification task aims to train a classifier  $\mathcal{F}$  to predict the node class label  $\mathbf{y}$ . We denote the labeled and unlabeled node set as  $\mathbf{L}$  and  $\mathbf{U}$ , respectively.

### 2.2 Group PageRank: Measuring the Supervision Information from Labeled Nodes

In this part, we introduce an indicator designed to measure the node received information from labeled nodes. Graph-based SSL takes both node and adjacency features as input;

we focus on the adjacency feature and mask the node feature to **disentangle** the effect of graph topology from that of node embedding. Specifically, we modify the PageRank algorithm with grouping and vectorization, to extend it into the graph-based SSL scene.

PageRank algorithm [Page *et al.*, 1999] is a very popular algorithm for measuring the importance of graph nodes. The original PageRank ( $\pi_{pr}$ ) is calculated via:

$$\pi_{pr} = (1 - \alpha)\mathbf{A}'\pi_{pr} + \alpha\mathbf{I} \quad (1)$$

with  $\mathbf{A}' = \mathbf{A}\mathbf{D}^{-1}$ ;  $\mathbf{D}$  is the degree matrix;  $\mathbf{I} \in \mathbb{R}^n$  is the **teleport vector** and filled with value  $1/n$ ;  $\alpha \in (0, 1]$  represents the **random walk restart** probability. Thereafter, Personal PageRank [Haveliwala, 2003] modifies PageRank by restarting from one node to measure the influence from this node's view, and has been recently applied in GNN training [Klicpera *et al.*, 2019].

In our scenario, we want to measure the influence from a group of nodes, i.e., the labeled nodes with the same class, so we make the random walk restart at a random node in the group with probability  $\alpha$ , which is named **Group PageRank** ( $\pi_{gpr}$ ):

$$\pi_{gpr}(c) = (1 - \alpha)\mathbf{A}'\pi_{gpr} + \alpha\mathbf{I}_c \quad (2)$$

$c \in [0, k)$  is the class index and  $k$  is the category size;  $\mathbf{I}_c \in \mathbb{R}^n$  is the teleport vector:

$$\mathbf{I}_c^i = \begin{cases} \frac{1}{|\mathbf{L}_c|}, & \text{if the } i\text{-th node is a labeled node of class } c \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$\mathbf{L}_c$  denotes the set of labeled nodes with class  $c$ . We calculate the Group PageRank for each group set individually and then concatenate all the Group PageRank vectors to form the final Group PageRank matrix  $\mathbf{Z} \in \mathbb{R}^{n \times k}$ .  $\mathbf{Z}_{i,j}$  represents the supervision influence of category  $i$  on node  $j$ . Practically, we can compute the Group PageRank matrix  $\mathbf{Z}$  parallelly:

$$\mathbf{Z} = \alpha(\mathbf{E} - (1 - \alpha)\mathbf{A}')^{-1}\mathbf{I}^* \quad (4)$$

$\mathbf{E}$  is the unit matrix;  $\mathbf{I}^* \in \mathbb{R}^{n \times k}$  is the concatenation of  $\mathbf{I}_c$ .

### 2.3 The Joint Effect of Graph and Contrastive Learning

With Group PageRank matrix  $\mathbf{Z}$ , we can clearly tell the impact of labeled nodes from different categories on the unlabeled nodes in the graph. Furthermore, we want to find how this information helps the downstream task.

We take the naming convention **Topology Information Gain** (**TIG**) to describe the task information effectiveness that the node obtains from information source (labeled nodes) along the graph topology. Ideally, the received information for each node should be concentrated on one category, which means that the supervision information is strong and clear. So we regard the maximum item of vector  $\mathbf{Z}_i$  to be the most valid information type from graph for node  $i$  and the other items as the confusing information. The TIG value for the  $i$ -th node ( $\mathbf{T}_i$ ) is calculated in the following equations:

$$\mathbf{Z}_{i,c}^* = \frac{1}{2}\mathbf{Z}_{i,c}(1 + \frac{\mathbf{X}_i\mathcal{P}_c^T}{\|\mathbf{X}_i\|\|\mathcal{P}_c\|}), \mathcal{P}_c = \frac{1}{|\mathbf{L}_c|} \sum_{i \in \mathbf{L}_c} \mathbf{X}_i \quad (5)$$

$$\mathbf{T}_i = \max(\mathbf{Z}_i^*) - \lambda \frac{(\sum_{c=0}^{k-1} \mathbf{Z}_{i,c}^*) - \max(\mathbf{Z}_i^*)}{k - 1} \quad (6)$$

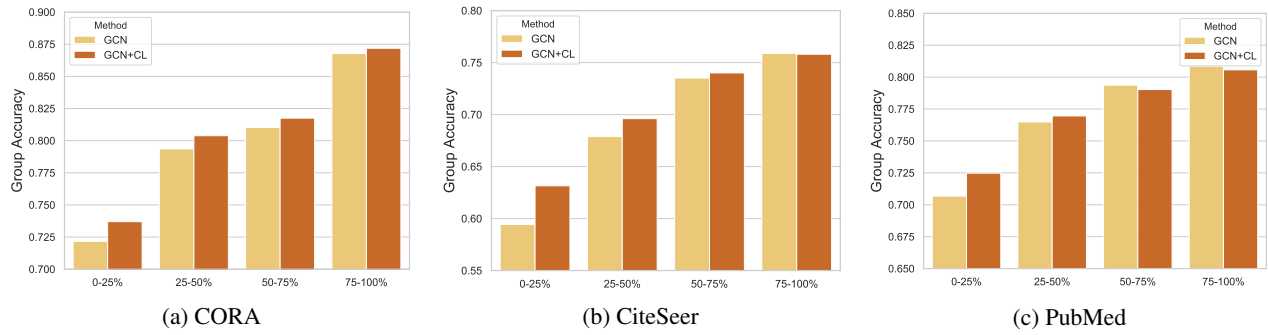


Figure 2: The CL effect on node groups with topological distance to the information source (annotated nodes) from the farthest to the nearest. We can obviously find that: (1) GNN performs better on nodes that are close to the information source where nodes can naturally obtain sufficient task information along graph edges; and (2) CL mainly works on nodes that are topologically far away from the information source where nodes receive insufficient task information from the labeled nodes.

$\max(\cdot)$  is a maximum function and  $\lambda$  is the punishment factor of confusing information from the other types.  $\mathcal{P}_c$  is the prototype embedding of class  $c$  by averaging the embedding of all the annotated nodes in class  $c$ . We assume that each unlabeled node own its special preference for information from different categories, and adopt the cosine similarity between node embedding  $X_i$  and  $\mathcal{P}_c$  to adjust the original Group PageRank value  $Z_i$  from the node feature view.

With the TIG value, we can now probe the CL working mechanism on graph. We conduct a probing experiment on CORA dataset [Sen *et al.*, 2008] with GCN [Kipf and Welling, 2017] as the graph encoder. We rank nodes in the test set according to their TIG values from smallest to largest and divide these test nodes into four groups. We then test the model performance on these four groups with and without the CL module. As shown in Figure 2, we can obviously find that with the increase of TIG value, which means that node receive more effective information from graph, the GNN works better but the improvement brought by CL becomes smaller. We can conclude that **CL mainly improves the performance on nodes that are topologically far away from the labeled nodes.**

### 3 Distance-wise Graph Contrastive Learning

Our analysis in Section 2 demonstrates the importance of task information distribution in GCL. Motivated by this, we propose to enhance the contrastive learning in a distance-wise manner. Specifically, note that contrastive learning consists of two most important components: data augmentation (perturbation) and contrastive pair sampling. For the data augmentation, we propose a distance-wise graph perturbation mechanism based on the TIG values of nodes, from a global view of the supervision information distribution on the whole graph. As for the sampling strategy, from the local view of each node, we design a graph-specific hard negative sampling method based on Group PageRank and other features of each node. Our proposed sampling method can effectively select the semi-difficult nodes for each node, bringing clear improvements to the contrastive learning.

Apart from these two modules, we first propose to schedule CL loss weight for different sub-graphs. We conduct cosine

annealing schedule on CL loss weight based on the node TIG value. Therefore, nodes receiving less effective supervision information from graph can access more supplementary information from CL. The contrastive learning framework is thus more suitable for graph-based SSL tasks.

#### 3.1 Distance-wise Graph Perturbation

Data augmentation (perturbation) is an essential component in contrastive learning. Unlike perturbation in computer vision [He *et al.*, 2020; Sohn *et al.*, 2020] where samples can be perturbed independently, graph nodes interact with each other and the disturbance of one node will spread to its adjacent nodes. Therefore, if we perturb too many nodes in graph, the augmented graph will be too different from the original one and ruin the CL effect. In existing studies, there are two general ideas to remedy this issue: randomly or heuristically choosing part of graph nodes to disturb [Zhu *et al.*, 2020a; Zhu *et al.*, 2020b] or augmenting each node in its parallel universe [Wang *et al.*, 2020]. However, neither of these two approaches take account of the differences of nodes received task information from labeled node set.

We argue that the limited graph perturbation should be more likely to take place on local subgraphs with insufficient supervision information from graph learning. Thus, we propose to sample nodes to disturb according to their TIG value: a node will be selected with a higher probability for augmentation when it has a lower TIG value. Besides, considering the spreading of node augmentation, we dynamically adjust the selection probability by reducing the probability of subgraph around the augmented node. The detailed distance-wise graph perturbation strategy is shown in Appendix A.

#### 3.2 Distance-wise Contrastive Pair Sampling

Contrastive pair sampling, especially the negative pair sampling, has a crucial impact on CL [Robinson *et al.*, 2020; Kalantidis *et al.*, 2020]. Different from existing works which simply regard all other nodes [Wang *et al.*, 2020; Zhu *et al.*, 2020a] or nodes with other class [Wan *et al.*, 2020] as negative pairs, we propose to sample contrastive pairs based on the relative distance between nodes.

Our measurement for the node relative distance mainly depends on the difference of node Group PageRank, which

contains the global topology information and annotation information. Specifically, we normalize the Group PageRank vector  $\mathbf{p}^i$  and then calculate the Kullback-Leibler divergence ( $\text{KL}(\cdot, \cdot)$ ) between nodes as the global topology distance:

$$D_{i,j}^g = \text{KL}(\mathbf{p}_i, \mathbf{p}_j), \text{ with } \mathbf{p}_i = \text{NORM}(\mathbf{Z}_i^*) \quad (7)$$

$\text{NORM}(\cdot)$  is the normalization function to transfer the original Group PageRank vector ( $\mathbf{Z}_i^*$ ) into a probability distribution over all the categories. Besides, we supplement the relative distance measuring from the view of local topology distance and the node initial embedding distance. For nodes local topology distance  $D_{i,j}^l$ , we use the minimum jump hop number as indicator. For nodes embedding distance  $D_{i,j}^e$ , we calculate the cosine distance between the nodes embedding matrix. The final node relative distance is calculated by the weighted sum of the global/local topology distance and the nodes embedding distance:

$$D_{i,j} = S(D_{i,j}^g) + \lambda_1 S(D_{i,j}^l) + \lambda_2 S(D_{i,j}^e) \quad (8)$$

$\lambda_1, \lambda_2$  is the weight of the two supplement items;  $S(\cdot)$  represents the scale operation to transfer the original value to  $[0, 1]$ . Experimental results in Section 4.5 have shown that the  $D_{i,j}^g$  play the most important role in the node relative distance measuring.

Then we construct positive and negative pairs for each node individually. For the  $i$ -th node (anchor node), the positive set  $\mathbf{P}_i$  is composed of the closest nodes with the smallest relative distance; for the negative set  $\mathbf{N}_i$ , we propose to sample the semi-difficult nodes [Schroff *et al.*, 2015; Kalantidis *et al.*, 2020] which are neither too far or too close to the anchor node. Given the personally ranked node list  $\mathbf{R}_i$  sorted by  $D_i$  from smallest to largest (the  $i$ -th node itself is excluded),  $\mathbf{P}_i$  and  $\mathbf{N}_i$  are truncated from  $\mathbf{R}_i$ , respectively:

$$\mathbf{P}_i = \mathbf{R}_i[0 : \text{post}_{\text{end}}], \mathbf{N}_i = \mathbf{R}_i[\text{negt}_{\text{beg}} : \text{negt}_{\text{end}}] \quad (9)$$

### 3.3 The Complete DwGCL Framework

For a GNN encoder  $\mathcal{F}$  (e.g., GCN), the supervised Cross Entropy loss  $\mathcal{L}_{CE}$  is calculated on the labeled set  $\mathbf{L}$ :

$$\mathbf{t} = \mathcal{F}(\mathbf{X}, \mathbf{A}, \boldsymbol{\theta}) \quad (10)$$

$$\mathcal{L}_{CE} = -\frac{1}{|\mathbf{L}|} \sum_{i \in \mathbf{L}} \sum_{c=0}^{k-1} \mathbf{y}_i \log p(\mathbf{t}_i^c, \tau) \quad (11)$$

$\mathbf{t}_i$  is the GNN output for node  $i$ ;  $\mathbf{y}_i$  is the gold label;  $\boldsymbol{\theta}$  is the parameters for model  $\mathcal{F}$ ;  $p(\mathbf{t}_i)$  is the softmax output of  $\mathbf{t}_i$  with temperature  $\tau$ .

Apart from the supervised loss, DwGCL consists of three kinds of unsupervised contrastive loss: the self-consistency loss, the contrastive loss with positive pairs and that with negative pairs. The self-consistency loss is calculated in the following equations:

$$\mathcal{L}_s^i = \text{KL}(\mathcal{F}(\mathbf{X}_p, \mathbf{A}_p, \boldsymbol{\theta})_i, \mathcal{F}(\mathbf{X}\mathbf{A}, \tilde{\boldsymbol{\theta}})_i) \quad (12)$$

The augmented node embedding matrix  $\mathbf{X}_p$  and adjacency matrix  $\mathbf{A}_p$  are generated by distance-wise graph perturbation. Following Xie *et al.* [2019] and Wang *et al.* [2020],  $\tilde{\boldsymbol{\theta}}$  is a fixed copy of current parameter  $\boldsymbol{\theta}$  and the gradient is not propagated

Dataset	Class	Feature	Node	Edge	Labeled Node
CORA	7	1,433	2,708	5,429	140
CiteSeer	6	3,703	3,327	4,732	120
PubMed	3	500	19,717	44,338	60
Photo	8	745	7,487	119,043	160
Computers	10	767	13,381	245,778	200

Table 1: Statistical information about datasets. Each class has 20 labeled nodes in the standard semi-supervised setting.

through  $\tilde{\boldsymbol{\theta}}$ . Similarly, the contrastive loss with positive and negative pairs are computed:

$$\mathcal{L}_p^i = \frac{1}{|\mathbf{P}_i|} \sum_{j \in \mathbf{P}_i} \text{KL}(\mathcal{F}(\mathbf{X}_p, \mathbf{A}_p, \boldsymbol{\theta})_i, \mathcal{F}(\mathbf{X}\mathbf{A}, \tilde{\boldsymbol{\theta}})_j) \quad (13)$$

$$\mathcal{L}_n^i = \frac{1}{|\mathbf{N}_i|} \sum_{j \in \mathbf{N}_i} \text{KL}(\mathcal{F}(\mathbf{X}_p, \mathbf{A}_p, \boldsymbol{\theta})_i, \mathcal{F}(\mathbf{X}\mathbf{A}, \tilde{\boldsymbol{\theta}})_j) \quad (14)$$

Existing CL related studies usually apply the inner production [He *et al.*, 2020; Wan *et al.*, 2020] or KL divergence [Xie *et al.*, 2019; Wang *et al.*, 2020] as the score function in contrastive pair similarity measuring, while our empirical results shows that KL divergence is more stable and effective than inner production for the low-dimension GNN output. We conjecture the reason lies in that the last GNN encoder layer usually also works as the classifier layer, so the the inner production between low-dimension logits vector is not robust and brings confusing signals to GNN training.

Then the complete unsupervised loss on node  $i$  is calculated in the following way:

$$\mathcal{L}_U^i = \mathcal{L}_s^i + \mu_1 \mathcal{L}_p^i - \mu_2 \mathcal{L}_n^i \quad (15)$$

$\mu_1$  and  $\mu_2$  is the weight of loss from positive and negative pairs, respectively. Experiments show that the model works best when  $\mu_1$  is 2-3 times that of  $\mu_2$ .

Instead of all the existing GCL works [Zhu *et al.*, 2020a; Wan *et al.*, 2020; Wang *et al.*, 2020] that add CL loss evenly with supervised loss on the whole graph, we propose to adjust the CL loss weight adaptively among different sub-graphs according to the amount of their received supervised information. Specifically, we set the CL weight  $w_i$  of each node differently using a cosine annealing schedule based on the node TIG value:

$$w_i = w_{\min} + \frac{1}{2}(w_{\max} - w_{\min})(1 + \cos(\frac{\text{Rank}(\mathbf{T}_i)}{n}\pi)). \quad (16)$$

$w_{\min}, w_{\max}$  is the minimum and the maximum value of CL loss, respectively;  $\text{Rank}(\mathbf{T}_i)$  is the ranking order of  $\mathbf{T}_i$  from smallest to largest. Then we combine the supervised and unsupervised loss in a distance-wise manner as the final loss:

$$\mathcal{L} = \mathcal{L}_{CE} + \frac{1}{n} \sum_{i=0}^{n-1} w_i \mathcal{L}_U^i \quad (17)$$

With this manner, sub-graphs receiving less supervision information from graph can benefit from the augmented supplementary information from CL.



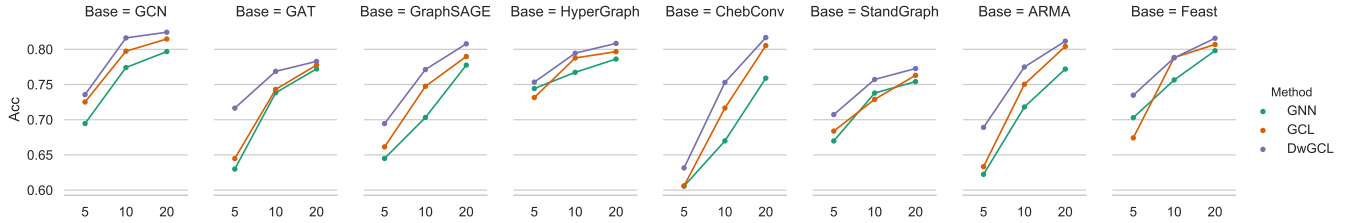


Figure 3: The results of the DwGCL method on multiple base GNN encoders under different labeled node sizes (5/10/20 annotated nodes in each category) in the CORA dataset. In all settings, DwGCL generally outperforms basic GNNs and GCL methods without the distance-wise mechanism.

## 4 Experiments

In this section, we first give a brief introduction of datasets used for experiments. Details of experimental settings are provided for reproducibility. We then show the effectiveness of our proposed DwGCL by comparing with both advanced GNN models and other GCL methods on multiple datasets. Then we test our method with various GNN encoders and annotation sizes in order to prove the generalizability of our methods. Further ablation studies are conducted to explore the contribution of different modules in our method.

### 4.1 Datasets

We conduct experiments on five widely-used graph benchmark datasets, namely paper citations networks [Sen *et al.*, 2008] (*CORA*, *CiteSeer* and *Pubmed*) and Amazon Co-purchase networks [Shchur *et al.*, 2018] (*Photo* and *Computers*). The statistics of datasets used are presented in Table 1.

### 4.2 Experimental Settings

In experiments, we use the standard semi-supervised node classification setting, i.e.g, each category has 20 nodes for training and 30 nodes for validation, and all the rest nodes are used for testing. Following Shchur *et al.* [2018] and Sun *et al.* [2019], we run 20 random dataset splittings for each experiment to relieve the random error caused by dataset selections.

We implement all methods based on PyTorch [Paszke *et al.*, 2017] and PyTorch Geometric [Fey and Lenssen, 2019]. For all comparable GNN encoders, we stack two GNN layers with ReLU activation function and with dropout probability set to 0.5. We use the Adam optimizer [Kingma and Ba, 2014] and early stopping strategy in training. The dataset-sensitive hyper-parameters, including learning rate, hidden size,  $\lambda$ ,  $\mu_1$ ,  $w_{\min}$  and  $\text{neg}t_{beg}$ , are tuned on the validation set for each dataset individually. In each group of experiments, all the hyper-parameters shared between the baselines and proposed methods are strictly aligned, to clearly show the pure effect brought by our proposed DwGCL method.

### 4.3 Results

We set two kinds of baselines to verify the effectiveness of our method: (1) Popular GNN models, including GCN [Kipf and Welling, 2017], GAT [Veličković *et al.*, 2018], GraphSAGE [Hamilton *et al.*, 2017] and GraphMix [Verma *et al.*, 2019]; (2) Recently proposed GCL methods without distance-wise mechanism, including GRACE [Zhu *et al.*,

Method	CORA	CiteSeer	PubMed	Photo	Computers
GCN	81.0 $\pm$ 0.7	69.1 $\pm$ 0.8	76.8 $\pm$ 0.8	88.0 $\pm$ 1.7	80.7 $\pm$ 0.7
GAT	79.9 $\pm$ 2.1	69.6 $\pm$ 1.4	77.2 $\pm$ 1.0	88.7 $\pm$ 1.9	77.9 $\pm$ 2.1
GraphSAGE	80.7 $\pm$ 1.2	67.8 $\pm$ 1.4	76.3 $\pm$ 0.6	87.7 $\pm$ 1.2	79.6 $\pm$ 1.2
GraphMix	81.5 $\pm$ 0.9	69.3 $\pm$ 0.4	78.0 $\pm$ 1.3	89.5 $\pm$ 1.7	81.2 $\pm$ 2.7
GRACE	82.1 $\pm$ 1.6	70.3 $\pm$ 1.5	77.8 $\pm$ 1.8	89.3 $\pm$ 1.3	82.1 $\pm$ 1.5
NodeAug	82.7 $\pm$ 1.3	71.1 $\pm$ 1.2	78.4 $\pm$ 0.7	89.7 $\pm$ 1.8	83.0 $\pm$ 1.9
MultiView	81.8 $\pm$ 0.8	71.5 $\pm$ 1.7	77.9 $\pm$ 0.1	88.9 $\pm$ 1.5	82.8 $\pm$ 2.0
GraphConL	82.4 $\pm$ 1.1	70.8 $\pm$ 0.6	78.2 $\pm$ 0.5	90.3 $\pm$ 2.3	83.2 $\pm$ 0.7
DwGCL(Ours)	<b>83.6<math>\pm</math>0.8</b>	<b>72.5<math>\pm</math>0.6</b>	<b>79.2<math>\pm</math>0.6</b>	<b>91.0<math>\pm</math>0.6</b>	<b>83.6<math>\pm</math>1.4</b>

Table 2: Empirical results of the proposed DwGCL and baseline methods. The mean value and standard deviation of the classification accuracy are reported based on 20 rounds experiments with different dataset splitting. The top four methods are typical GNN models without CL mechanism, and the middle four methods are equipped with CL modules. Best results are shown in bold. Methods with CL module obviously outperform those without CL module, and our DwGCL can further improve the CL effect on all datasets.

2020a], NodeAug [Wang *et al.*, 2020], MultiView [Hassani and Khasahmadi, 2020] and GraphConL [You *et al.*, 2020].

The overall results are shown in Table 2. We can find that: (1) Methods with CL modules (middle 4 models) generally outperform that without CL modules (top 4 models), since it can leverage the information from the large amount of unsupervised nodes by comparing the perturbed representations. (2) Our DwGCL can further improve the CL effect in graph-based SSL by paying more attention to the nodes that are topologically far away from the labeled nodes and thus receive insufficient or confusing task information from graph.

### 4.4 Generalizability of DwGCL

Our DwGCL method is model-agnostic and thus can be applied to any GNN encoders. To prove the generalizability of DwGCL, we test the effect of our DwGCL with multiple GNN architectures and different labeled nodes sizes. As shown in Figure 3, we conduct experiments on 8 different GNN models: GCN [Kipf and Welling, 2017], GAT [Veličković *et al.*, 2018], GraphSAGE [Hamilton *et al.*, 2017], HyperGraph [Bai *et al.*, 2019], ChebConv [Defferrard *et al.*, 2016], StandGraph [Morris *et al.*, 2019], ARMA [Bianchi *et al.*, 2019] and Feast [Verma *et al.*, 2018] on CORA dataset. We can find that, although these models vary greatly in terms of mathematics and architecture, our DwGCL method can effectively promote the GNN performance comparing to CL method without distance-wise module under different settings

Version	CORA	CiteSeer	PubMed
Full Method	83.6 $\pm$ 0.8	72.5 $\pm$ 0.6	79.2 $\pm$ 0.6
w/o Group PageRank	80.8 $\pm$ 1.7 ( $\downarrow$ 2.8)	69.5 $\pm$ 1.2 ( $\downarrow$ 3.0)	77.4 $\pm$ 2.3 ( $\downarrow$ 1.8)
w/o Dw Perturbation	81.8 $\pm$ 1.0 ( $\downarrow$ 1.8)	70.9 $\pm$ 0.7 ( $\downarrow$ 1.6)	78.1 $\pm$ 0.7 ( $\downarrow$ 1.1)
w/o Dw CL Schedule	82.7 $\pm$ 1.3 ( $\downarrow$ 0.9)	72.1 $\pm$ 0.6 ( $\downarrow$ 0.4)	78.7 $\pm$ 0.7 ( $\downarrow$ 0.5)
w/o $D^g$ (GpPR KL)	81.9 $\pm$ 1.0 ( $\downarrow$ 1.7)	70.8 $\pm$ 0.6 ( $\downarrow$ 1.7)	78.3 $\pm$ 0.9 ( $\downarrow$ 0.6)
w/o $D^l$ (Jump Hop)	83.1 $\pm$ 1.0 ( $\downarrow$ 0.5)	72.1 $\pm$ 0.7 ( $\downarrow$ 0.4)	79.0 $\pm$ 0.6 ( $\downarrow$ 0.2)
w/o $D^e$ (Node Simi.)	82.8 $\pm$ 1.3 ( $\downarrow$ 0.8)	71.9 $\pm$ 0.5 ( $\downarrow$ 0.6)	78.8 $\pm$ 0.8 ( $\downarrow$ 0.4)

Table 3: Ablation study results of DwGCL from both global (top three rows) and personal view (bottom three rows). **w/o Group PageRank** means using the minimum jump hop instead of Group PageRank to measure the graph nodes distance. **w/o Dw Perturbation** represents conducting the general graph perturbation without distance-wise mechanism. **w/o Dw CL Schedule** represents removing  $w_i$  in Eq. 17. The bottom three ablation versions are designed to verify the effect of the relative distance metrics in Eq. 8.

of labeling size (5/10/20 annotated nodes in each category), which proves the effectiveness of our DwGCL framework in general graph-based SSL scenarios.

#### 4.5 Ablation Study

To verify the contribution of different modules in our proposed method, we conduct ablation study by removing or replacing different components of DwGCL based on GCN. The detailed ablated versions and their performance are displayed in Table 3. The experiment in the first row shows that, when replacing Group PageRank with a minimum jump hop number as the topology distance measurement, our model performance along all the three citation graphs suffers a sharp decline, which proves the effectiveness of the proposed Group PageRank on measuring the task information obtained by the unlabeled node from graph topology. Besides, when the distance-wise graph perturbation and CL loss schedule module are removed, the results also decline. Moreover, removing distance-wise graph perturbation causes a more significant drop, demonstrating that it has a bigger impact in our model. The results of three ablation versions that removing different relative distance metrics in Eq. 8 shows that the global topology distance ( $D^g$ ) plays a more important role than local topology distance ( $D^l$ ) and nodes embedding distance ( $D^e$ ).

## 5 Analysis

Here we provide in-depth analyses of our proposal, regarding why DwGCL can enhance graph contrastive learning, the over-smoothing problem in graph learning, and the trade-off in contrastive pair sampling strategy.

**DwGCL enhance CL effect on under-represented nodes in graph.** To explain why our methods can help graph-based SSL benefit more from CL, we display the comparison of our DwGCL with GCL (our ablation version) method at each TIG rank group in Figure 4. We can find that our DwGCL method obviously outperforms GCL models at groups with smaller TIG values and achieves comparable performance at groups with larger TIG values. This indicates that our DwGCL can effectively strengthen the CL effect for the node with insufficient task information from graph learning, thus achieving higher

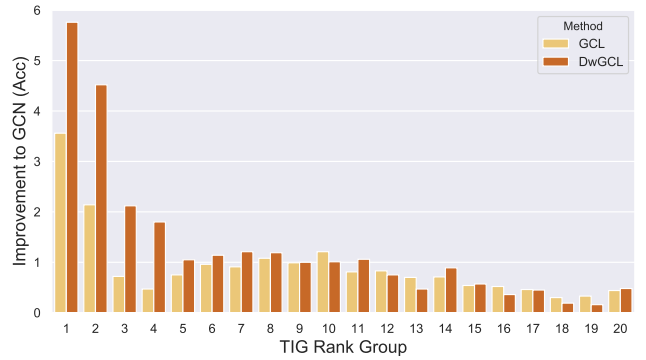


Figure 4: The improvement gap comparing to GCN across TIG rank groups from smallest to largest. We can find that our DwGCL method can further strengthen the CL effect on the remote nodes than the GCL method.

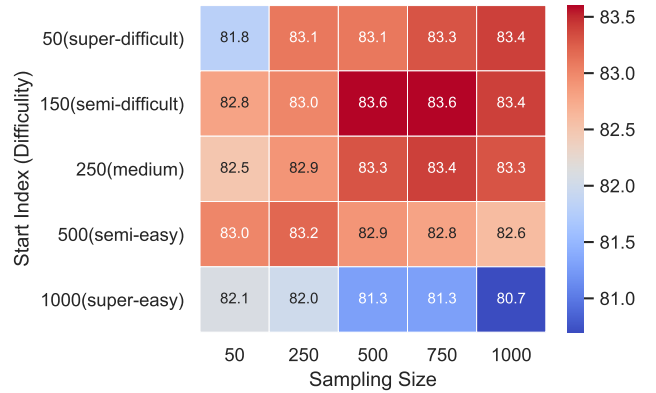


Figure 5: Heatmap of DwGCL's performance under different negative pair sampling settings (Start Index means  $\text{negt}_{beg}$  in Eq. 9; Sampling Size means  $\text{negt}_{end} - \text{negt}_{beg}$ ). A smaller Start Index represents that the sampled nodes are closer to the anchor node and thus it is harder to tell them apart, and vice versa.

performance than existing CL methods without distance-wise mechanism.

**DwGCL effectively alleviates over-smoothing problem.** Over-smoothing is a common issue in GNNs training [Zhao and Akoglu, 2019; Li *et al.*, 2018], especially when stacking multiple layers in GNN models. In Table 4, we display the MADGap [Chen *et al.*, 2020a; Feng *et al.*, 2020] value, which is an indicator for model over-smoothness, for GCNs with various layers on CORA dataset. We can find that DwGCL can effectively relieve the over-smoothing problem, thus promoting the GNN performance even for deep GNNs. We analyze the reason lies in that, the interactions of graph nodes make nodes similar to each other [Chen *et al.*, 2020a], while the vanilla GCN model lacks the mechanism for pulling representations of nodes apart. The contrastive learning module can relieve this problem to some extent by self-consistency or pair-wise regularization. Our DwGCL adaptively samples the positive and negative pairs for each node according to the relative distance measurement, thus alleviates the over-smoothing phenomenon more effectively than the existing GCL method.

Layer	Acc ( $\uparrow$ )			MADGap ( $\uparrow$ )		
	GCN	GCL	DwGCL	GCN	GCL	DwGCL
2	79.3 $\pm$ 1.7	81.6 $\pm$ 1.8	83.6 $\pm$ 0.8	0.82 $\pm$ 0.02	0.85 $\pm$ 0.02	0.87 $\pm$ 0.01
4	76.2 $\pm$ 1.2	81.3 $\pm$ 1.6	82.7 $\pm$ 1.4	0.77 $\pm$ 0.04	0.78 $\pm$ 0.02	0.78 $\pm$ 0.01
6	62.0 $\pm$ 9.9	77.0 $\pm$ 2.5	79.5 $\pm$ 1.5	0.66 $\pm$ 0.10	0.76 $\pm$ 0.02	0.77 $\pm$ 0.02
8	43.5 $\pm$ 10.3	68.5 $\pm$ 8.2	75.8 $\pm$ 4.7	0.62 $\pm$ 0.10	0.65 $\pm$ 0.08	0.72 $\pm$ 0.05

Table 4: Analysis of the over-smoothing phenomenon in deep GNNs. MADGap measures the GNN over-smoothing degree.  $\uparrow$  means the bigger the better. We can find that DwGCL can effectively relieve the over-smoothing phenomenon and improve model performance for deep GNNs.

### Trade-off between sampling size and difficulty of samples.

Contrastive pair sampling has been proven to be essential for the performance of contrastive learning [Chen *et al.*, 2020d; Chen *et al.*, 2020c]. In Eq. 9, we propose to rank all the node by the relative distance from the anchor node and then sample the contrastive pairs. Empirical results have shown that the closest nodes works best as the positive pairs, and for the negative pair sampling, we study the influence of sampling size and difficulty of sampled nodes in Figure 5. We observe that model can hardly trained well by too easy or too difficult negative pairs, and the semi-difficult nodes serve best as negative pairs. Existing works [Kalantidis *et al.*, 2020; Robinson *et al.*, 2020] shown that the hard negative sampling is helpful for CL, and Schroff *et al.* [2015] have shown the effectiveness of semi-difficult negative pairs in other CL scenes other than graph. Besides, we notice that increasing the negative sampling size can effectively improve CL performance. However, a very large sampling size will naturally reduce the difficulty of the selected pairs, demonstrating a trade-off between the sampling size and the difficulty of sampled nodes.

## 6 Related Work

The general idea behind contrastive learning (CL) is to utilize self-supervised information between contrastive pairs, which are constructed by adding random perturbation on the original samples. By regularizing the perturbed representations, i.e., pulling the distance between positive pairs closer while pushing negative pairs apart, CL learns high quality representations for both images [Chen *et al.*, 2020c; Sohn *et al.*, 2020; Berthelot *et al.*, 2019] and texts [Chen *et al.*, 2020b; Qu *et al.*, 2020; Luo *et al.*, 2020].

Graph neural networks (GNNs) [Yang *et al.*, 2016; Kipf and Welling, 2017; Veličković *et al.*, 2018; Hamilton *et al.*, 2017] have shown prominent modeling ability in graph-based SSL, since it can take advantage of the graph structure to propagate task information. However, traditional GNNs usually calculate the loss on the limited labeled node set only, which limits its modeling ability.

Recently, efforts have been made for applying CL to graph-based semi-supervised learning, since it can effectively utilize the large amount unlabeled nodes and boost the performance of GNNs. Existing GCL studies mainly focus on developing graph data augmentation methods: Zhu *et al.* [2020a] propose to generate different graph views by randomly removes edges and masking node features. Wang *et al.* [2020] design a Noda-Parallel Augmentation scheme to perturb each nodes

independently. Wan *et al.* [2020] propose to access different graph views with different GNN encoders and introduce the graph generative learning target. Hassani and Khasahmadi [2020] propose to generate various views of graph by contrasting encodings from first-order neighbors and a graph diffusion. You *et al.* [2020] analyze the influence of different combination of graph augmentations. Zhu *et al.* [2020b] propose some heuristic graph augmentation methods based on graph node centrality. Feng *et al.* [2020] design a random propagation strategy to perform graph data augmentation.

However, all these existing GCL methods ignore the unbalanced task information distribution and apply CL uniformly to the whole graph. Our DwGCL can solve these two issues and combines contrastive learning with graph learning more adaptively and effectively with the distance-wise manner.

## 7 Conclusion

In this work, we investigate contrastive learning in the graph-based semi-supervised learning. We point out that current graph contrastive learning methods lack consideration of the uneven distribution of task information across the graph. We show that the benefit brought by contrastive learning in graph learning mainly comes from the nodes that are topologically far away from the information source, since these nodes lack sufficient task information from graph while can obtain supplementary supervision from contrastive learning.

Motivated by our findings, we propose the Distance-wise Graph Contrastive Learning framework to improve the CL effect on graph learning. We design a distance-wise graph perturbation strategy and a CL loss weight schedule mechanism to strength the learning of nodes that are far away from the information source. We also propose a novel contrastive pair sampling by taking both the topological characteristic and feature embedding of each node into account, to further enhance the efficiency of contrastive pair construction. Comprehensive experiments on five benchmarks demonstrate the superiority of DwGCL over previous methods. Further analysis on various graph semi-supervised learning scenarios demonstrates the generalizability of our proposal.

## Acknowledgement

This work was supported by a Tencent Research Grant. Xu Sun is the corresponding author of this paper.

## References

- [Bai *et al.*, 2019] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph Convolution and Hypergraph Attention. *arXiv preprint arXiv:1901.08150*, 2019.
- [Berthelot *et al.*, 2019] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.
- [Bianchi *et al.*, 2019] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph Neural Networks with Convolutional Arma Filters. *arXiv preprint arXiv:1901.01343*, 2019.

- [Chen *et al.*, 2020a] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and Relieving the Over-smoothing Problem for Graph Neural Networks from the Topological View. In *AAAI*, pages 3438–3445, 2020.
- [Chen *et al.*, 2020b] Jiaao Chen, Zichao Yang, and Diyi Yang. MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification. In *ACL*, pages 2147–2157, 2020.
- [Chen *et al.*, 2020c] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.
- [Chen *et al.*, 2020d] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.
- [Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *NIPS*, pages 3837–3845, 2016.
- [Feng *et al.*, 2020] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. In *NeurIPS*, 2020.
- [Fey and Lenssen, 2019] Matthias Fey and Jan E. Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.
- [Hamilton *et al.*, 2017] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *NIPS*, pages 1024–1034, 2017.
- [Hassani and Khasahmadi, 2020] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive Multi-View Representation Learning on Graphs. In *ICML*, pages 3451–3461, 2020.
- [Haveliwala, 2003] Taher H Haveliwala. Topic-sensitive Pagerank: A Context-sensitive Ranking Algorithm for Web Search. *TKDE*, 15(4):784–796, 2003.
- [He *et al.*, 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.
- [Kalantidis *et al.*, 2020] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard Negative Mixing for Contrastive Learning. *arXiv preprint arXiv:2010.01028*, 2020.
- [Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.
- [Klicpera *et al.*, 2019] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*, 2019.
- [Li *et al.*, 2018] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper Insights into Graph Convolutional Networks for Semi-supervised Learning. In *AAAI*, pages 3538–3545, 2018.
- [Luo *et al.*, 2020] Fuli Luo, Pengcheng Yang, Shicheng Li, Xuancheng Ren, and Xu Sun. CAPT: Contrastive Pre-Training for Learning Denoised Sequence Representations. *arXiv preprint arXiv:2010.06351*, 2020.
- [Morris *et al.*, 2019] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman go Neural: Higher-order Graph Neural Networks. In *AAAI*, volume 33, pages 4602–4609, 2019.
- [Page *et al.*, 1999] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
- [Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *Proceedings of Workshop of Neural Information Processing Systems*, 2017.
- [Qu *et al.*, 2020] Yanru Qu, Dinghan Shen, Yelong Shen, Sandra Sajeev, Jiawei Han, and Weizhu Chen. CoDA: Contrast-enhanced and Diversity-promoting Data Augmentation for Natural Language Understanding. *arXiv preprint arXiv:2010.08670*, 2020.
- [Robinson *et al.*, 2020] Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive Learning with Hard Negative Samples. *arXiv preprint arXiv:2010.04592*, 2020.
- [Schroff *et al.*, 2015] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.
- [Sen *et al.*, 2008] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.
- [Shchur *et al.*, 2018] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation. *arXiv preprint arXiv:1811.05868*, 2018.
- [Sohn *et al.*, 2020] Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020.
- [Sun *et al.*, 2019] Ke Sun, Piotr Koniusz, and Jeff Wang. Fisher-Bures Adversary Graph Convolutional Networks. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 465–475, 2019.



- [Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.
- [Verma *et al.*, 2018] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered Graph Convolutions for 3d Shape Analysis. In *CVPR*, pages 2598–2606, 2018.
- [Verma *et al.*, 2019] Vikas Verma, Meng Qu, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. GraphMix: Regularized Training of Graph Neural Networks for Semi-Supervised Learning. *arXiv preprint arXiv:1909.11715*, 2019.
- [Wan *et al.*, 2020] Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. Contrastive and Generative Graph Convolutional Networks for Graph-based Semi-Supervised Learning. *arXiv preprint arXiv:2009.07111*, 2020.
- [Wang *et al.*, 2020] Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. NodeAug: Semi-Supervised Node Classification with Data Augmentation. In *KDD*, pages 207–217, 2020.
- [Xie *et al.*, 2019] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.
- [Yang *et al.*, 2016] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting Semi-supervised Learning with Graph Embeddings. In *ICML 2016*, pages 40–48, 2016.
- [You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph Contrastive Learning with Augmentations. In *NeurIPS*, 2020.
- [Zhao and Akoglu, 2019] Lingxiao Zhao and Leman Akoglu. PairNorm: Tackling Oversmoothing in GNNs. *arXiv preprint arXiv:1909.12223*, 2019.
- [Zhu *et al.*, 2020a] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep Graph Contrastive Representation Learning. *arXiv preprint arXiv:2006.04131*, 2020.
- [Zhu *et al.*, 2020b] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph Contrastive Learning with Adaptive Augmentation. *arXiv preprint arXiv:2010.14945*, 2020.

## A Distance-wise Graph Perturbation Algorithm

The details of our distance-wise graph perturbation algorithm in displayed in Algorithm 1. Nodes to augment are selected according to their received task information from graph. Three kinds of augmentation operations are taken:

- Randomly adding edges
- Randomly removing edges
- Randomly masking node features

We conduct sequential selection for nodes, and dynamically adjust the selecting probability by reducing the probability of

sub-graphs to remedy the excessive perturbation issue caused by the spreading of graph perturbation. The algorithm stops when the graph disturbance reaches a preset threshold.

---

### Algorithm 1 Distance-wise Graph Perturbation with Negative Feedback

---

**Require:** Graph node size  $n$ , embedding dim  $h$ , feature matrix  $\mathbf{X}$ , Adjacency Matrix  $\mathbf{A}$ , Jump Hop Matrix  $\mathcal{H}$ . CL schedule weight  $w$ , sharpen coefficient  $t$ , graph perturbation threshold  $\sigma$ , maximum edge adding/removing number  $n_{add}, n_{rmv}$ , feature mask rate  $m$ , correction hop range list  $\mathbf{L}^H$ , correction decay ratio list  $\mathbf{L}^D$ .

```

1: function CORRECT_PROB( $i, \text{old}_p, \text{hop}_i$ )
2:    $\text{new}_p \leftarrow \text{old}_p$ 
3:    $\text{new}_p[i] = 0$ 
4:   for ( $\text{iter}_h, \text{iter}_w$ )  $\in$  ZIP( $\mathbf{L}^H, \mathbf{L}^D$ ) do
5:      $\text{neb}_{idx} = \text{WHERE}(\text{hop}_i == \text{iter}_h)$ 
6:      $\text{new}_p[\text{neb}_{idx}] * = \text{iter}_w$ 
7:   end for
8:   return NORM( $\text{new}_p$ )
9: end function

10:  $\mathbf{X}_p \leftarrow \mathbf{X}, \mathbf{A}_p \leftarrow \mathbf{A}$ 
11:  $\text{prob} \leftarrow \text{NORM}(\text{POWER}(w, t))$ 
12:  $\text{gap\_now} \leftarrow 0$ 
     $\triangleright$  Sharpen the selection probability distribution
13: while  $\text{gap\_now} < \sigma$  do
14:    $i = \text{PROB\_CHOICE}(\text{RANGE}(n), 1, \text{prob})$ 
15:    $\text{hop}_i \leftarrow \mathcal{H}[i]$ 
16:    $\text{cand}_{add} = \text{hop}_i[\text{WHERE}(\text{hop}_i == 0)]$ 
17:    $\text{cand}_{rmv} = \text{hop}_i[\text{WHERE}(\text{hop}_i == 1)]$ 
18:    $\text{add}_{idx} = \text{RANDOM\_CHOICE}(\text{cand}_{add}, n_{add})$ 
19:    $\text{rmv}_{idx} = \text{RANDOM\_CHOICE}(\text{cand}_{rmv}, n_{rmv})$ 
20:    $\mathbf{A}_p[\text{add}_{idx}][i] = \mathbf{A}_p[i][\text{add}_{idx}] = 1$ 
     $\triangleright$  Randomly add edges
21:    $\mathbf{A}_p[\text{rmv}_{idx}][i] = \mathbf{A}_p[i][\text{rmv}_{idx}] = 0$ 
     $\triangleright$  Randomly remove edges
22:    $\text{mask}_{idx} = \text{RANDOM\_CHOICE}(\text{RANGE}(h), h * t)$ 
23:    $\mathbf{X}_p[i][\text{mask}_{idx}] = 0$   $\triangleright$  Randomly mask node feature
24:    $\text{prob} = \text{CORRECT\_PROB}(i, \text{prob}, \text{hop}_i)$ 
25:    $\text{gap\_now} = \sqrt{\sum_{i=1}^n \sum_{j=1}^n (\mathbf{A}_p[i][j] - \mathbf{A}[i][j])^2}$ 
     $\triangleright$  Calculating the Frobenius norm

26: end while

Ensure:  $\mathbf{X}_p, \mathbf{A}_p$ 

```

---