Regularizing Graph Neural Networks via Consistency-Diversity Graph Augmentations

Deyu Bo[1] *, **BinBin Hu**[2], **Xiao Wang**[1], **Zhiqiang Zhang**[2], **Chuan Shi**[1] †, **Jun Zhou**[2]

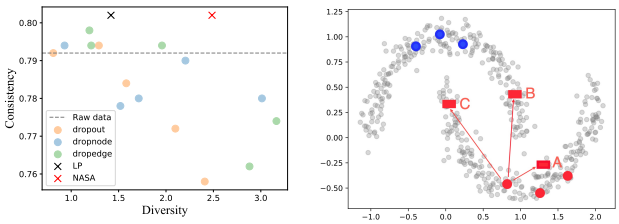
[1] Beijing University of Posts and Telecommunications
[2] Ant Financial Services Group, Hangzhou, China
{bodeyu, xiaowang, shichuan}@bupt.edu.cn, {bin.hbb,lingyao.zzq,jun.zhoujun}@antfin.com

Abstract

Despite the remarkable performance of graph neural networks (GNNs) in semi-supervised learning, it is criticized for not making full use of unlabeled data and suffering from overfitting. Recently, graph data augmentation, used to improve both accuracy and generalization of GNNs, has received considerable attentions. However, one fundamental question is how to evaluate the quality of graph augmentations in principle? In this paper, we propose two metrics, *Consistency* and *Diversity*, from the aspects of augmentation correctness and generalization. Moreover, we discover that existing augmentations fall into a dilemma between these two metrics. Can we find a graph augmentation satisfying both consistency and diversity? A well-informed answer can help us understand the mechanism behind graph augmentation and improve the performance of GNNs. To tackle this challenge, we analyze two representative semi-supervised learning algorithms: label propagation (LP) and consistency regularization (CR). We find that LP utilizes the prior knowledge of graphs to improve consistency and CR adopts variable augmentations to promote diversity. Based on this discovery, we treat neighbors as augmentations to capture the prior knowledge embodying homophily assumption, which promises a high consistency of augmentations. To further promote diversity, we randomly replace the immediate neighbors of each node with its remote neighbors. After that, a neighbor-constrained regularization is proposed to enforce the predictions of the augmented neighbors to be consistent with each other. Extensive experiments on five real-world graphs validate the superiority of our method in improving the accuracy and generalization of GNNs.

1 Introduction

Graph neural networks (GNNs), as a typical graph-based semi-supervised learning (SSL) method, has achieved state-of-the-art performance (Kipf and Welling 2017; Velickovic et al. 2018). Despite its success, GNNs has been criticized for not making full use of unlabeled data (Wang et al. 2020; Feng et al. 2020), which is an essential requirement of SSL (Yang et al. 2021). Previous methods tend to use pseudo labels to overcome this limitation (Sun, Lin, and Zhu 2020; Li, Han, and Wu 2018), but suffer from poor calibration (Guo et al. 2017). Recently, graph data augmentation is used to improve both accuracy and generalization of GNNs (Rong et al. 2020; Verma et al. 2021; Feng et al. 2019, 2020; Wang et al. 2020).



(a) Dilemma of augmentations (b) A toy semi-supervised dataset

Figure 1: (a) Consistency and diversity of different augmentations on Cora dataset. Dotted line represents the consistency of the raw data. Circles with different colors indicate different graph augmentations. Black and red crosses show the consistency and diversity of immediate neighbors and our proposed augmentation. (b) Toy example of data augmentations in SSL. Blue and red circles are labeled data, gray circles are unlabeled data and rectangles are augmentations.

Although there are some augmentation strategies on graphs, such as DropEdge (Rong et al. 2020) and DropNode (Feng et al. 2020), it is still unknown which augmentation is better for GNNs. Generally, an easy augmentation contributes less to the generalization of model and a hard augmentation may bring additional noise (Yin et al. 2019). Therefore, a natural question is *how to evaluate the quality of graph augmentations in principle?* To this goal, as the first contribution of this paper, we propose two metrics of graph augmentation in SSL: *Consistency* and *Diversity*. Consistency indicates whether the augmented data belong to the same class with the raw data and diversity reveals how different the distribution captured by augmented data is from raw data. Detailed descriptions can be found in Sec. 2. If the augmentation and original data are in different classes, it will hurt the accuracy of the model. While if the augmentation is similar to original data, it may contribute less to the generalization of the model. Therefore, a good augmentation should not only ensure the correctness but also provide sufficient generalization.

Based on the two evaluations, we test three commonly used graph augmentations, i.e., Dropout (Srivastava et al.

*Work done during Deyu's internship at Ant Group.
†Corresponding Author.

2014), DropEdge and DropNode, with different dropping rates. The results are shown in Fig. 1(a), where there is a dilemma between consistency and diversity: an augmentation with high consistency may have less diversity and vice versa. Since the dilemma of existing graph augmentations is identified, a natural question is *can we find a graph augmentation satisfying both consistency and diversity at the same time?* This is not a trivial task because we need to quantitatively define consistency and diversity for graph data and make a delicate balance between them.

To solve the dilemma, we need to know the factors that affect consistency and diversity. We analyze two representative SSL methods, label propagation (LP) and consistency regularization (CR), and find that LP uses neighbors as augmentations, which naturally captures the prior knowledge of graphs and improves consistency. While CR employs variable augmentations to promote diversity. Based on this discovery, in this paper, we propose NASA, short for Neighbors Are Special Augmentations, to augment and regularize GNNs. NASA consists of two parts: augmentation and regularization. In the augmentation, we treat neighbors as special augmentations and propose to disturb nodes by replacing their immediate neighbors with remote neighbors. Generally, neighbors can capture the prior knowledge of graphs, i.e., homophily assumption, and replacing neighbors can improve the variability, so we can preserve high consistency and diversity simultaneously. In the regularization, we propose a neighbor-constrained regularization, which enforces the predictions of neighbors to be consistent with each other, so that a large number of unlabeled nodes can be used in training. Moreover, we show that the proposed regularization can be used as a supplement of the traditional graph regularization.

The contribution of this paper is summarized as follows:

- We propose consistency and diversity to evaluate the quality of existing graph augmentations, and find that they cannot satisfy the two metrics at the same time. To the best of our knowledge, this is the first exploration of metrics of graph augmentations.
- We propose NASA, which generates graph augmentations with high consistency and diversity through replacing immediate neighbors with remote neighbors, and constrains the predictions of augmented neighbors to be consistent.
- We validate the effectiveness of NASA by comparing with state-of-the-art methods on five real-world datasets. We also conduct a generalization test to verify the superiority of NASA on improving the generalization of GNNs.

2 Evaluation of Augmentation

In this section, we will introduce the detailed description of the two metrics, i.e., Consistency and Diversity. Before that, we first explain the motivation for designing the two metrics.

Let's take the "two moons" data as an example (Verma et al. 2019), as shown in Fig. 1(b), where the blue and red circles are labeled data, and gray circles are unlabeled data. We can see that *the number of labeled data is relatively small and cannot reflect the distribution of the entire data*. In this situation, we consider three types of augmentations, i.e., *A, B, C*. It is obvious that although *A* lies in the correct class, it contributes little information because it is close to raw data (high consistency, low diversity); *B* is different from raw data, but it locates in a wrong class, which brings additional noise (low consistency, high diversity); *C* benefits the classification a lot because it not only has correct labels but also brings additional generalization (high consistency, high diversity). The aforementioned discussion shows that a good augmentation should generalize to the distribution beyond training data. Therefore, only using labeled data cannot comprehensively evaluate the quality of augmentations. To better measure the correctness and generalization of the augmentations, we need to introduce additional data, e.g. validation set, for evaluation. The main idea is as follows:

We first train two models $\mathcal{F}_\theta, \widetilde{\mathcal{F}}_\theta : \mathbb{R}^d \to \mathbb{R}^C$, through the training data D_{train} and its augmentations $\widetilde{D}_{train}$, respectively, where d is the dimension of input features, C is the number of classes and θ denotes the parameters. After that, we use the two models to predict on the validation set D_{val}. If the augmentations have better correctness and generalization, the model $\widetilde{\mathcal{F}}_\theta$ should have higher accuracy on validation set and establish a more different decision boundary from $\mathcal{F}_\theta$. This leads to the metrics of consistency and diversity:

Metric of Consistency. We use the accuracy of augmented model on validation set to represent the level of consistency:

$$\mathcal{C} = Acc(\widetilde{\mathcal{F}}_\theta(D_{val}), Y_{val}), \tag{1}$$

where Y_{val} denotes the labels of validation data. A lower value of $\mathcal{C}$ means that the augmentations are inconsistent with the raw data, which may hurt the accuracy of the model. However, a higher value of $\mathcal{C}$ does not mean that the quality of augmentation is necessarily good, because it may contribute less to the generalization of the model, which leads to the metric of diversity.

Metric of Diversity. We use the difference between the predictions of the original model $\mathcal{F}_\theta$ and augmented model $\widetilde{\mathcal{F}}_\theta$ to represent the level of diversity:

$$\mathcal{D} = ||\widetilde{\mathcal{F}}_\theta(D_{val}) - \mathcal{F}_\theta(D_{val})||_F^2, \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm. A lower value of $\mathcal{D}$ indicates that the augmentations have a similar distribution with original data, which cannot benefit the generalization of models (Yin et al. 2019). But a higher value of $\mathcal{D}$ cannot ensure the correctness of augmentations. Therefore, the combination of the two metrics is necessary for the evaluation.

Note that the metrics of consistency and diversity are not limited to graph data. Instead, they can be used to evaluate the quality of data augmentations in other semi-supervised field, such as computer vision (Berthelot et al. 2019; Xie et al. 2020). In the next section, we will introduce our method and explain how these two metrics guide the model design.

3 Methodology

Let $G = (V, E)$ denote a graph, where V is the set of nodes with $|V| = N$ and E is the set of edges. Each graph G has an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$, where $\mathbf{A}_{ij} = 1$ means there is an edge between v_i and v_j, otherwise 0.

$\mathbf{X} \in \mathbb{R}^{N\times d}$ are the node features and $\mathbf{H} \in \mathbb{R}^{N\times C}$ are the node presentations learned by GNNs. Generally, most existing GNN can be summarized as a message passing architecture (Gilmer et al. 2017), which can be formulated as $\mathbf{H} = \mathbf{Trans}(\mathbf{Agg}\{\mathbf{A}, \mathbf{X}; \mathbf{\Phi}\}; \mathbf{\Theta})$. $\mathbf{Agg}$ means aggregating information from neighbors in the graphs and $\mathbf{Trans}$ is to transform the aggregated information into new node representations. The parameters $\mathbf{\Phi}$, $\mathbf{\Theta}$ are used for aggregation and transformation, respectively. In a graph augmentation, the perturbation may occur in both node features and structures. Therefore, the augmented node representations can be calculated as $\widetilde{\mathbf{H}} = \mathbf{Trans}(\mathbf{Agg}\{\widetilde{\mathbf{A}}, \widetilde{\mathbf{X}}; \mathbf{\Phi}\}; \mathbf{\Theta})$, where $\widetilde{\mathbf{A}}$ and $\widetilde{\mathbf{X}}$ are the augmented features and structures, respectively.

3.1 Connection Between Consistency Regularization and Label Propagation

A basic requirement of SSL is to make good use of the unlabeled data (van Engelen and Hoos 2020; Chong et al. 2020). Here we review two representative SSL algorithms and discuss how they use augmentations to assist unlabeled nodes.

Label propagation is a traditional graph-based semi-supervised algorithm, which propagates labels to unlabeled nodes along graph topology (Zhou et al. 2003). The objective function can be defined as:

$$\mathcal{L}_{LP} = \sum_{i\in V_L} ||\mathbf{h}_i - \mathbf{y}_i||_2^2 + \alpha \sum_{i\in V} \sum_{j\in \mathcal{N}_i} ||\mathbf{h}_i - \mathbf{h}_j||_2^2, \quad (3)$$

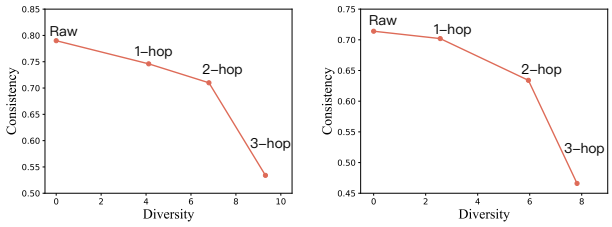
where $\mathbf{h}_i$ is the i-th row of $\mathbf{H}$, V_L represents the labeled nodes, α is a hyper-parameter, $\mathbf{y}_i$ is a one-hot vector denoted as the label of v_i and $\mathcal{N}_i$ denotes the neighbors of v_i. The first term is a classification loss, here we take the mean square loss as example. The second term is a graph Laplacian regularization, which enforces the representations of neighbors to be consistent. Note that the closed-form solution of Eq. 3 is $\mathbf{H} = (\mathbf{I} + \alpha\mathbf{L})^{-1}\mathbf{Y}$, where $\mathbf{L}$ is the Laplacian matrix of $\mathbf{A}$.

Consistency regularization is an emerging semi-supervised model, which enforces model to have similar predictions between raw data and random augmentations, so that the model will be robust to the small data perturbations (Xie et al. 2020). The objective function can be formulated as:

$$\mathcal{L}_{CR} = \sum_{i\in V_L} ||\mathbf{h}_i - \mathbf{y}_i||_2^2 + \alpha \sum_{i\in V} \sum_{k=1}^{K} ||\mathbf{h}_i - \widetilde{\mathbf{h}}_i^{(k)}||_2^2, \quad (4)$$

where K is the number of random augmentations and $\widetilde{\mathbf{h}}_i^{(k)}$ is the representation of k-th augmentation. The first term of CR is the same as LP and the second term is a regularization, which uses the prediction of v_i as a pseudo label to supervise the output of its augmentations.

Remark 1. *(Two perspectives of LP and CR) Comparing Eq. 3 and Eq. 4, we can find that the difference between LP and CR is the regularization. From the perspective of LP, using neighbors as augmentations explicitly utilizes the prior knowledge of graphs, i.e., the homophily assumption. Therefore, the consistency of neighbors is higher than random augmentations. From the perspective of CR, the features and structures of neighbors $\boldsymbol{h}_j$ are fixed during training, while random augmentations $\widetilde{\mathbf{h}}_i^{(k)}$ will change dynamically, e.g. dropedge will drop different edges in each epoch, which improves the generalization of GNNs implicitly.*



(a) Neighbors on Cora (b) Neighbors on Citeseer

Figure 2: Empirical study of different neighbors' consistency and diversity. "Raw" represents the original training nodes, and "k-hop" indicates the neighbors that are k hops away from the training nodes, where $k \in \{1, 2, 3\}$.

The aforementioned discussion reveals that a good augmentation should not only utilize of the prior knowledge of data (for consistency), but also provide variable augmentations (for diversity). This motivates the design of our model.

3.2 Our Proposed Model: NASA

We introduce the details of our proposed model, which consists of two components: augmentation and regularization. In the augmentation, we propose to use remote neighbors to replace immediate neighbors to promote diversity. In the regularization, we propose two techniques to constrain the predictions of augmentations.

Augmentation on Neighbors. Inspired by the design of LP, we aim to use neighbors as augmentations to improve the consistency. However, this way lacks variability and may be affected by the noise. Therefore, an effective augmentation strategy is to change the neighbors during training.

To determine which neighbors we should use as substitutes, we make an empirical study to identify their quality. Specifically, we divide the neighbors into different groups according to their distances to the training nodes. We then calculate the consistency and diversity through Eq. 1 and 2, where we use graph convolutional networks (GCNs) as the test model $\mathcal{F}_\theta$, the training nodes are D_{train} and their neighbors are $\widetilde{D}_{train}$. The results are shown in Fig. 2. It can be seen that the farther the neighbors are from training data, the lower the consistency and the higher the diversity. In particular, comparing the 2-hop neighbors with 3-hop neighbors, we can find that the consistency of 2-hop neighbors decreases slightly, but 3-hop neighbors hurt the consistency heavily and do not add much diversity.

Based on the results, we propose Neighbor Replace (NR) to randomly replace the 1-hop neighbors by the 2-hop neighbors. Specifically, for node v_i, we use a Bernoulli distribution to sample its neighbors randomly, i.e., $\forall v_j \in \mathcal{N}_i$, $\epsilon_j \sim Bern(p)$. For each sampled neighbor v_j with $\epsilon_j = 1$, we drop the edges between v_j and v_i, and randomly choose a neighbor of v_j as the new neighbor of v_i, i.e., $\mathcal{N}_i^{new} = \{v_k \sim \mathcal{N}_j, \epsilon_j = 1\}$. For the neighbors with $\epsilon_j = 0$, we do not change them and denote them as $\mathcal{N}_i^{old} = \{v_j \in \mathcal{N}_i, \epsilon_j = 0\}$.

Therefore, the augmented neighbors of v_i is defined as $\widetilde{\mathcal{N}}_i = \mathcal{N}_i^{new} \cup \mathcal{N}_i^{old}$. The benefits of NR are two-fold: first, the exchange between 1-hop neighbors and 2-hop neighbors perturbs graph structures, but does not seriously hurt the correctness. Second, the supervision signals can be propagated to more unlabeled nodes so that the generalization can be promoted.

Although graph structures contain the consistency information, the inter-edges (Zhao et al. 2021) and NR augmentations may introduce some noise. Here we propose two techniques, i.e., neighbor-constrained regularization and dynamic training, to prevent pseudo labels from being heavily disturbed.

Neighbor-constrained Regularization. After perturbing the neighbors of each node, we feed the augmented graph topology $\widetilde{\mathbf{A}}$ and original node features $\mathbf{X}$ into an arbitrary GNNs to learn the node representations: $\widetilde{\mathbf{H}} = \mathbf{Trans}(\mathbf{Agg}\{\widetilde{\mathbf{A}}, \mathbf{X}; \mathbf{\Phi}\}; \mathbf{\Theta})$. For the labeled nodes, a cross-entropy loss is used to supervise the predictions of GNNs:

$$\mathcal{L}_{CE} = -\frac{1}{N_L} \sum_{i \in V_L} \mathbf{y}_i \log \widetilde{\mathbf{h}}_i. \tag{5}$$

Note that here we use labels to supervise the augmented representations $\widetilde{\mathbf{h}}_i$ because we find that this approach can reduce the risk of over-fitting. For the unlabeled nodes, we design a novel neighbor-constrained regularization to enforce the predictions of neighbors to be consistent with each other. Specifically, we first fuse the predictions of neighbors as the pseudo label of the center node: $\widetilde{\mathbf{y}}_i = \frac{1}{|\widetilde{\mathcal{N}}_i|} \sum_{j \in \widetilde{\mathcal{N}}_i} \widetilde{\mathbf{h}}_j$. The average of neighbors' predictions is similar to the *voting results*, which can effectively prevent the pseudo labels from being affected by the noisy neighbors.

Before using the averaged pseudo labels to supervise the prediction of neighbors, we utilize the *sharpening* trick to enforce the classifier output a low-entropy prediction:

$$\widetilde{\mathbf{p}}_{ij} = \widetilde{\mathbf{y}}_{ij}^{\frac{1}{T}} \Big/ \sum_{c=0}^{C-1} \widetilde{\mathbf{y}}_{ic}^{\frac{1}{T}}, \tag{6}$$

where $T \in (0, 1]$ is a scaling factor, controlling the sharpness of the prediction, i is the index of nodes, j and c indicate the specific dimensions of the representation ($0 < j < C - 1$). Then we use the sharpened pseudo labels to supervise the predictions of augmented neighbors:

$$\mathcal{L}_{CR} = \frac{1}{N} \sum_{i \in V} \sum_{j \in \widetilde{\mathcal{N}}_i} KL\left(\widetilde{\mathbf{p}}_i || \widetilde{\mathbf{h}}_j\right), \tag{7}$$

where KL is the Kullback-Leibler divergence (Joyce 2011), measuring the distance between two distributions. Besides, we will not use the gradient of the pseudo label $\widetilde{\mathbf{p}}_i$ to update parameters $\mathbf{\Phi}$ and $\mathbf{\Theta}$, as suggested by (Miyato et al. 2019). Through this regularization, unlabeled nodes can be used in training to prevent the model from over-fitting. The final loss function is the combination of classification and neighbor-constrained regularization:

$$\mathcal{L} = \mathcal{L}_{CE} + \alpha \mathcal{L}_{CR}, \tag{8}$$

Table 1: Statistics of datasets.

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Computer	13,381	245,778	767	10
Photo	7,487	119,043	745	8

where α is a hyper-parameter for balancing.

Finally, we give a further explanation to show why this regularization is called "neighbor-constrained". In addition, we analyze its connection to the traditional graph regularization (Belkin and Niyogi 2003). We can rewrite Eq. 7 as:

$$\mathcal{L}_{CR} = \frac{1}{N} \sum_{i \in V} \sum_{j \in \widetilde{\mathcal{N}}_i} \left(\widetilde{\mathbf{p}}_i \log \widetilde{\mathbf{p}}_i - \widetilde{\mathbf{p}}_i \log \widetilde{\mathbf{h}}_j\right), \tag{9}$$

where the first term can be removed because of the gradient truncation. Therefore, if we ignore the sharpening trick, the second term can be rewritten as:

$$\mathcal{L}_{CR} = -\frac{1}{N^2} \sum_{p,q \in \widetilde{\mathcal{N}}_i} \widetilde{\mathbf{h}}_p \log \widetilde{\mathbf{h}}_q, \tag{10}$$

which can be seen as the cross-entropy loss between the augmented neighbors. Eq. 10 requires the predictions of neighbors to be consistent with each other. That is why we call this regularization "neighbor-constrained".

Connection with manifold learning. Similart to Eq. 3 and Eq. 4, the objective function of NASA can be rewritten as:

$$\mathcal{L} = \sum_{i \in V_L} ||\widetilde{\mathbf{h}}_i - \mathbf{y}_i||_2^2 + \alpha \sum_{i \in V} \sum_{j \in \mathcal{N}_i} ||\widetilde{\mathbf{h}}_j - \sum_j \widetilde{\mathbf{h}}_j||_2^2. \tag{11}$$

The second term of Eq. 11 is similar to the local linear embedding (LLE) (Roweis and Saul 2000) algorithm, which uses the weighted sum of neighbors to reconstruct the target nodes. In this way, the manifold of high-dimensional data can be preserved in the low dimensional space.

Dynamic Training. During training, we perform NR on each node in each epoch, that is to say, the augmented graph topology $\widetilde{\mathbf{A}}$ is different in each epoch. We call this dynamic training, otherwise static training. The dynamic training of NASA makes the model more robust. On the one hand, in each epoch, different neighbors are used for training, which makes the model to be invariant to the change of neighbors. On the other hand, there may exist some neighbors that do not belong to the same class. Using dynamic training can prevent the model from over-fitting the unsatisfactory augmentations. Ablation studies can be found in Sec. 4.3.

Complexity. The time complexity consists of two parts: one is the complexity of GNNs. Here we take GCNs (Kipf and Welling 2017) as an example, whose complexity is $\mathcal{O}(L|E|d^2)$ and L is the number of layers. Another is the complexity of the regularization, whose complexity is $\mathcal{O}(|E|d)$. Therefore, the overall complexity of is $\mathcal{O}(|E|(Ld^2 + d))$, which is linear to the number of edges.

Table 2: Node classification results under different label split (%). A higher value indicates a better performance. **Bold** for the best. (-) means the standard deviation is too large to have a stable result.

	Standard Split			Less Label Split			Random Split	
	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed	Computer	Photo
LP	70.4±0.0	50.6±0.0	71.8±0.0	64.9±3.3	41.8±4.2	71.4±3.8	79.8±3.4	79.0±4.8
GLP	80.3±0.2	71.7±0.6	78.8±0.4	70.1±2.8	60.7±5.5	73.2±4.0	81.9±1.1	89.6±0.7
GCN-LPA	82.8±0.1	72.3±0.2	78.6±0.2	68.8±3.3	53.2±4.7	71.5±3.6	80.4±2.4	89.4±1.5
PTA	83.0±0.5	71.6±0.4	80.1±0.1	67.7±2.8	58.5±4.9	71.5±3.2	82.3±0.9	90.7±2.1
GCN	81.5±0.3	70.3±0.9	79.0±0.2	70.1±2.7	58.4±5.2	71.8±4.4	82.3±1.5	90.4±0.7
GAT	83.0±0.7	72.5±0.7	79.0±0.3	71.4±3.7	62.2±6.5	72.5±4.0	-	-
MixHop	81.9±0.4	71.4±0.8	80.8±0.6	67.9±3.0	59.0±5.5	71.3±3.1	-	-
GMNN	83.7±0.3	72.9±0.5	80.3±0.4	71.4±2.1	60.5±3.2	72.8±3.1	82.7±1.3	91.0±2.9
APPNP	83.8±0.3	71.6±0.5	79.7±0.3	69.9±2.1	59.3±2.8	71.4±3.5	82.1±1.9	90.6±2.0
GAUG	83.6±0.5	73.3±1.1	80.2±0.3	72.5±2.8	62.2±5.8	73.2±2.7	-	-
DropEdge	82.8±0.9	72.3±1.3	79.6±0.8	71.4±3.0	62.0±6.6	72.2±3.9	81.5±1.4	89.4±1.7
GraphVAT	82.9±0.5	73.8±0.9	79.5±0.3	70.6±4.2	61.2±5.1	73.4±3.3	82.3±3.1	90.5±2.6
GraphMix	83.9±0.6	74.7±0.6	81.0±0.5	72.3±6.1	61.0±4.5	**74.6±3.2**	84.2±2.5	91.3±1.9
GRAND	84.5±0.3	74.2±0.3	80.0±4.3	73.4±2.4	62.6±4.2	74.0±2.7	84.8±1.5	91.7±2.2
NodeAug	84.3±0.5	74.9±0.5	**81.5±0.5**	74.2±3.2	62.4±4.1	74.4±3.5	84.5±2.2	92.3±2.5
NASA	**85.1±0.3**	**75.5±0.4**	80.2±0.3	**75.2±4.0**	**63.4±4.8**	74.0±2.3	**85.5±3.3**	**92.7±2.9**

4 Experiments

4.1 Experimental Setup

We test the performance of different methods in the semi-supervised node classification task. Specifically, we use five different datasets — three citation datasets, e.g., Cora, Citeseer and Pubmed from (Kipf and Welling 2017) and two co-purchase datasets, e.g., Amazon Computers and Amazon Photo from (Shchur et al. 2018). The statistics of these datasets are shown in Table 1. Besides, we consider three different data splits to evaluate these methods more comprehensively. The first is the standard split of citation networks, provided by (Kipf and Welling 2017), which is widely used in the node classification task (Velickovic et al. 2018). In the standard split, each class has 20 labeled nodes, and 500 nodes for validation, 1000 nodes for testing. The second is a less label split of citation networks, where each class has 5 labeled nodes, and the set of validation and testing nodes is same to the standard split. The less label split poses a greater challenge to the model's generalization. The third split is the random split of co-purchase datasets, where 20 nodes per class are randomly sampled for training, 30 nodes for validation and others for testing, as suggested by (Shchur et al. 2018). All the data splits are widely used in previous works (Feng et al. 2020; Wang et al. 2020).

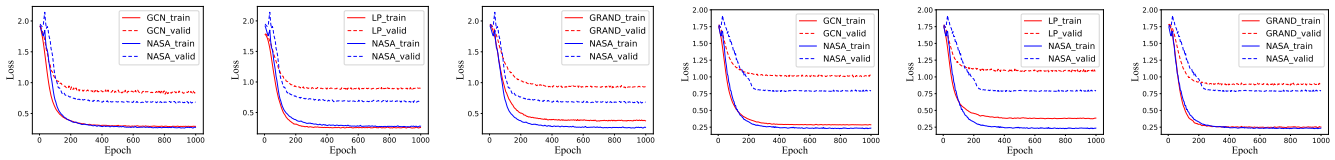
Benchmarks. We choose three kinds of methods as benchmarks: LP-based methods, GNNs-based methods and regularization-based methods. A detailed description and discussion of these methods can be found in Sec. 5

- LP-based methods: Original LP (Zhou et al. 2003), GLP (Li et al. 2019), GCN-LPA (Wang and Leskovec 2020) and PTA (Dong et al. 2021).
- GNNs-based methods: GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), MixHop (Abu-El-Haija et al. 2019), GMNN (Qu, Bengio, and Tang 2019) and APPNP (Klicpera, Bojchevski, and Günnemann 2019).
- Regularization-based methods: GAUG(Zhao et al. 2021), DropEdge (Rong et al. 2020), GraphVAT (Feng et al. 2019), GraphMix (Verma et al. 2021), GRAND (Feng et al. 2020) and NodeAug (Wang et al. 2020).

Implementation. The hyper-parameters are set as follows: learning rate=0.01, weight decay=1e-3, hidden unit=32 and Adam optimizer (Kingma and Ba 2015) for all methods. For the benchmarks, if the original papers provide the hyper-parameters, we set them as the authors suggested. For NASA, dropout rate is searched in $\{0.1, ..., 0.9\}$, temperature of sharpening is searched in $\{0.1, ..., 1.0\}$ and $\alpha = \{0.1, ..., 1.0\}$ for all datasets. We run NASA for 1000 epochs and select the model with the lowest validation loss for test. For the less label split and random split, we make 5 random splits with seed $\{0, 1, 2, 3, 4\}$, and for each method, we run 10 times and report the mean accuracy and standard deviation. Note that for fair comparison, we use the standard two-layer GCNs as the backbone for the regularization-based methods and NASA, because we want to ensure that the improvement comes from the regularization term itself instead of the advanced GNNs.

4.2 Performance on Node Classification

The performance of different methods are summarized in Table 2. From top to bottom, we show the results of the three types of baselines, from which we can draw the following conclusions: First, the accuracy of LP-based methods is usually lower than the other two types of methods, indicating



(a) NASA *vs.* GCN (b) NASA *vs.* LP (c) NASA *vs.* GRAND (d) NASA *vs.* GCN (e) NASA *vs.* LP (f) NASA *vs.* GRAND

Figure 3: Curves of training and validation loss on Cora (a-c) and Citeseer (d-f). A smaller gap between the training and validation loss indicates a better generalization.

Table 3: Ablation study on augmentation (%)

NASA	Cora	Citeseer
w/o augmentation	84.5±0.1	75.0±0.1
w/ NR	85.1±0.3	75.5±0.4
w/ dropedge	84.7±0.5	75.1±0.2
w/ dropnode	84.6±0.3	74.9±0.3
w/ dropout	84.5±0.2	74.7±0.2

Table 4: Ablation study on regularization (%)

NASA	Cora	Citeseer
w/ dynamic training	85.1±0.3	75.5±0.4
w/ static training	84.7±0.9	70.7±12.6
w/o augmentation	84.5±0.1	75.0±0.1
w/o neighbor	83.4±0.4	73.1±0.7
w/o sharpening	83.7±0.5	72.7±0.5

that only using the dependency of labels cannot achieve satisfactory results. Besides, the performance of regularization-based methods are significantly higher than the GNNs-based methods, which shows the effectiveness of regularization term. Specially, NASA relatively improves the performance of GCNs by 4.4%, 7.4% and 1.5% on standard Cora, Citeseer and Pubmed, respectively. As for the less label split, NASA makes more improvements, i.e., 7.3%, 8.6% and 3.1%, which proves the superiority of our proposed regularization in utilizing large amounts of unlabeled data. In the random split, NASA also achieves state-of-the-art performance. Finally, we notice that the performance of NASA on Pubmed is weaker than GraphMix and NodeAug. We guess this is because in Pubmed, the neighbors do not contribute much to classification.

4.3 Ablation Study

In order to prove the effectiveness of different components in NASA, we conduct two ablation study on two datasets: Cora and Citeseer. Specifically, we validate the effectiveness of the graph augmentation strategy and regularization term of NASA, respectively. The results are shown in Table 3 and 4.

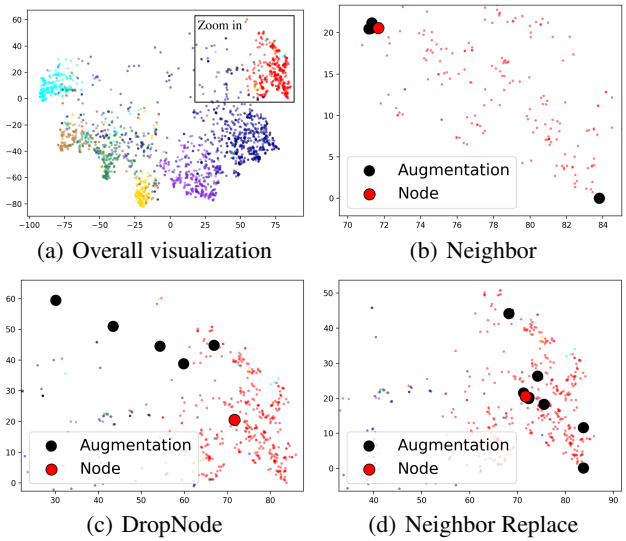
In Table 3, we test how different augmentation strategies influence the performance of NASA. First, we can find that without augmentation, the result is more stable but the accuracy drops, which indicates that augmentations can help to improve the performance of model. Besides, the augmentations on graph structures, i.e., NR and dropedge, are more useful than the augmentations on node features, i.e., dropnode and dropout. This phenomenon is also observed by (You, Ying, and Leskovec 2020). Therefore, the future work of graph augmentations can pay more attentions on perturbing the topology of the graphs.

In Table 4, we list the results of different variants of the regularization term in NASA. The first two rows reveal the advantage of dynamic training in regularization. We can find that the accuracy of static training is lower than dynamic training, and the standard deviation is much higher, especially in Citeseer. This shows that static training is easily affected by the extreme augmentations, while dynamic training is more stable. The middle two rows validate the effectiveness of augmentation and neighbor. Without any of them, the performance of the NASA will decrease, which reflects the observations in Fig. 1(a). The last row demonstrates the usefulness of sharpening.

4.4 Generalization Analysis

We design this experiment to validate the superiority of NASA on improving the generalization of GNNs. Specifically, we use the generalization gap (GP) to measure the generalization of different models. GP is a commonly used metric of model generalization (Jiang et al. 2019), which is defined as the difference between the training loss and validation loss. Note that a smaller value of GP indicates a better generalization. In the experiment, we first jointly optimize the classification and regularization loss in the training process. While in inference, the regularization term is removed, and the training and validation loss is calculated by the backbone GNNs only. In this situation, augmentations can only affect the models in the training stage, which requires the regularization term to make full use of the unlabeled data.

From Fig. 3, we can find that the gap of CR-based methods, i.e., NASA and GRAND, is always smaller than GCN and LP, indicating that CR has an advantage on improving the generalization of GNNs. Besides, compared with GRAND, the gap of NASA shrinks 12.5% and 25% on Citeseer and Cora, respectively. This observation shows that the regularization of NASA is more effective than the state-of-the-art regularization method on GNNs. It is worth noting that the shrinking of NASA's gap benefits from the decrease of validation loss



(a) Overall visualization (b) Neighbor (c) DropNode (d) Neighbor Replace

Figure 4: (a) Visualization of the node representations in Cora. Colors denote different classes. We zoom in the red class to show the augmentations of (b) Neighbors for LP, (c) DropNode for CR and (d) Neighbor Replace for NASA.

rather than the increase of training loss, which proves that NASA can make good use of the unlabeled data. Finally, we find an interesting phenomenon that the loss curve of NASA will increase in the begin of training. We think this is because the model tends to optimize the regularization term at first.

4.5 Case Visualization

In Fig. 1(a), we introduce the dilemma between the consistency and diversity of graph augmentations. Here, we give a closer visualization of different augmentations. We consider three graph augmentation strategies: immediate neighbors, DropNode and NR, which are corresponding to LP, CR and NASA, respectively. For DropNode, the drop probability is set to 0.5, as suggested by (Feng et al. 2020). We take one node in the training set as an example and zoom in its representation and augmentations together. The visualizations are shown in Fig. 4(a).

In Fig. 4(b), we can find that, except for one neighbor, the others (black circles) are close to the original node (red circle), which indicates that the consistency of neighbors is good, but the diversity is poor. In Fig. 4(c), the augmentations are far from the original node and some of them are out of the cluster. This shows that although DropNode can provide a better diversity, the consistency of it cannot be guaranteed. Fig. 4(d) shows the augmentations of NR. We can see that the augmentations are in the different locations of the cluster, which exhibits a better consistency and diversity than LP and CR. The reasons why NR performs well is that it uses the neighbors within two-hops as augmentations, which have more diversity than the immediate neighbors and better consistency than random augmentations.

5 Related Work

Label Propagation. LP (Zhou et al. 2003) is a simple yes effective algorithm in graph-based SSL, which propagates labels to the unlabeled nodes along network structures. The major shortcoming of LP is that it cannot utilize node features, so its performance heavily depends on the network structures and initialization. Some methods are proposed to deal with this problem. Generalized Label Propagation (GLP) (Li et al. 2019) generalizes LP by extending the graph filter of LP to node features. GCN-LPA (Wang and Leskovec 2020) combines GNNs with LP, where the objective function of LP is used to learn the weights of edges for graph convolution. Besides, (Dong et al. 2021) proves that the decoupled GCNs, e.g., APPNP(Klicpera, Bojchevski, and Günnemann 2019), is equal to a two-step label propagation.

Graph Neural Networks. GNNs makes a breakthrough in the field of semi-supervised node classification. Currently, GNNs can be divided into two categories: spectral methods and spatial methods. Spectral methods aim to utilize the theory of graph signal processing to design graph filters, such as GCN (Kipf and Welling 2017) and GraphHeat (Xu et al. 2019). Spatial methods focus on designing the message passing of GNNs. For example, GAT (Velickovic et al. 2018) uses attention mechanism to learn the importance of neighbors and MixHop (Abu-El-Haija et al. 2019) concatenates the representations of neighbors with different orders. However, none of them explicitly utilize the unlabeled nodes for training, which are easily to over-fit the scarce training data.

Regularization on GNNs. The use of CR in SSL is first adopted in the field of computer vision (Berthelot et al. 2019; Sohn et al. 2020; Xie et al. 2020) and then draws attentions in graph data. CR provides an explicit way to use unlabeled data, which significantly improve the generalization of models. Data augmentation is an important component of CR. In order to apply CR to GNNs, a lot of graph augmentations are proposed. For example, GRAND (Feng et al. 2020) proposes DropNode, GraphVAT (Feng et al. 2019) designs graph virtual adversarial training, GAUG (Zhao et al. 2021) proposes a learnable augmentation strategy and GraphMix (Verma et al. 2021) uses linear interpolation. They prefer to perform random perturbations on either graph structures or node features or both. Different from them, we tend to use the prior knowledge to augment graphs, thus guaranteeing the consistency of the augmentations.

6 Conclusions

In this paper, we study how to use graph augmentation to regularize GNNs and improve its performance and generalization ability. We find that existing graph augmentations fall into a dilemma between consistency and diversity. To solve this problem, we propose a new regularization, NASA, to utilize the augmented neighbors with high consistency and diversity to regularize GNNs. Experimental results validate the superiority of NASA on improving the performance and generalization of GNNs. An important future work is to prevent NASA from being affected by the noisy neighbors and generalize the method to heterophilic graphs.

7 Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. U20B2045, 61772082, 61702296, 62002029, 62172052), the Fundamental Research Funds for the Central Universities 2021RC28, and BUPT Excellent Ph.D. Students Foundation (No. CX2020115).

References

Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Alipourfard, N.; Lerman, K.; Harutyunyan, H.; Steeg, G. V.; and Galstyan, A. 2019. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. In *ICML*.

Belkin, M.; and Niyogi, P. 2003. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Comput.*, 15(6): 1373–1396.

Berthelot, D.; Carlini, N.; Goodfellow, I. J.; Papernot, N.; Oliver, A.; and Raffel, C. 2019. MixMatch: A Holistic Approach to Semi-Supervised Learning. In *NeurIPS*.

Chong, Y.; Ding, Y.; Yan, Q.; and Pan, S. 2020. Graph-based semi-supervised learning: A review. *Neurocomputing*, 408: 216–230.

Dong, H.; Chen, J.; Feng, F.; He, X.; Bi, S.; Ding, Z.; and Cui, P. 2021. On the Equivalence of Decoupled Graph Convolution Network and Label Propagation. In *WWW*.

Feng, F.; He, X.; Tang, J.; and Chua, T.-S. 2019. Graph Adversarial Training: Dynamically Regularizing Based on Graph Structure. *IEEE Trans. Knowl. Data Eng.*

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. In *NeurIPS*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In *ICML*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On Calibration of Modern Neural Networks. In *ICML*.

Jiang, Y.; Krishnan, D.; Mobahi, H.; and Bengio, S. 2019. Predicting the Generalization Gap in Deep Networks with Margin Distributions. In *ICLR*.

Joyce, J. M. 2011. Kullback-Leibler Divergence. In *International Encyclopedia of Statistical Science*, 720–722. Springer.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *ICLR*.

Li, Q.; Han, Z.; and Wu, X. 2018. Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning. In *AAAI*.

Li, Q.; Wu, X.; Liu, H.; Zhang, X.; and Guan, Z. 2019. Label Efficient Semi-Supervised Learning via Graph Filtering. In *CVPR*.

Miyato, T.; Maeda, S.; Koyama, M.; and Ishii, S. 2019. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8): 1979–1993.

Qu, M.; Bengio, Y.; and Tang, J. 2019. GMNN: Graph Markov Neural Networks. In *ICML*.

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

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

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of Graph Neural Network Evaluation. *CoRR*, abs/1811.05868.

Sohn, K.; Berthelot, D.; Carlini, N.; Zhang, Z.; Zhang, H.; Raffel, C.; Cubuk, E. D.; Kurakin, A.; and Li, C. 2020. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In *NeurIPS*.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958.

Sun, K.; Lin, Z.; and Zhu, Z. 2020. Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labeled Nodes. In *AAAI*.

van Engelen, J. E.; and Hoos, H. H. 2020. A survey on semi-supervised learning. *Mach. Learn.*, 109(2): 373–440.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Verma, V.; Lamb, A.; Kannala, J.; Bengio, Y.; and Lopez-Paz, D. 2019. Interpolation Consistency Training for Semi-supervised Learning. In *IJCAI*.

Verma, V.; Qu, M.; Lamb, A.; Bengio, Y.; Kannala, J.; and Tang, J. 2021. GraphMix: Regularized Training of Graph Neural Networks for Semi-Supervised Learning. *AAAI*.

Wang, H.; and Leskovec, J. 2020. Unifying Graph Convolutional Neural Networks and Label Propagation. *CoRR*, abs/2002.06755.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; Liu, J.; and Hooi, B. 2020. NodeAug: Semi-Supervised Node Classification with Data Augmentation. In *KDD*.

Xie, Q.; Dai, Z.; Hovy, E. H.; Luong, T.; and Le, Q. 2020. Unsupervised Data Augmentation for Consistency Training. In *NeurIPS*.

Xu, B.; Shen, H.; Cao, Q.; Cen, K.; and Cheng, X. 2019. Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning. In *IJCAI*.

Yang, X.; Song, Z.; King, I.; and Xu, Z. 2021. A Survey on Deep Semi-supervised Learning. *CoRR*, abs/2103.00550.

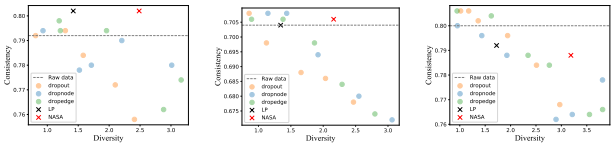
Yin, D.; Lopes, R. G.; Shlens, J.; Cubuk, E. D.; and Gilmer, J. 2019. A Fourier Perspective on Model Robustness in Computer Vision. In *NeurIPS*, 13255–13265.

You, J.; Ying, Z.; and Leskovec, J. 2020. Design Space for Graph Neural Networks. In *NeurIPS*.

Zhao, T.; Liu, Y.; Neves, L.; Woodford, O. J.; Jiang, M.; and Shah, N. 2021. Data Augmentation for Graph Neural Networks. In *AAAI*.

Zhou, D.; Bousquet, O.; Lal, T. N.; Weston, J.; and Schölkopf, B. 2003. Learning with Local and Global Consistency. In *NeurIPS*.

A Experimental Investigate



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

B Detailed Information of Datasets and Environment

The environment where the code runs is shown as follows:

- Operating system: Linux 4.9.151-015.x86_64.
- CPU information: Intel(R) Xeon(R) CPU E5-2682 v4 @2.50GHz.
- GPU information: NVIDIA® Tesla™ M40 GPU Computing Accelerator - 12G.

C Detailed Parameters of NASA

Table 1: Hyper-parameters of NASA.

Split	Dataset	Dropout	Balance (α)	Scaling (T)
Standard Split	Cora	0.7	1.0	0.5
	Citeseer	0.1	1.0	0.5
	Pubmed	0.5	0.5	0.2
Less Label Split	Cora	0.8	1.0	0.7
	Citeseer	0.8	1.0	1.0
	Pubmed	0.5	0.5	0.5
Random Split	Computer	0.3	0.7	0.5
	Photo	0.5	1.0	0.3

t

D Source Code of Benchmarks

We make sure that the code and data we use are public and do not contain any information about the authors of this paper. The acquisition of code and data complies with the provider's license and all of them do not contain any offensive content. The address of benchmarks' data and code are listed as follows:

Cora, Citeseer, Pubmed, Amazon-Computer & Amazon-Photo (Apache-2.0 License): https://docs.dgl.ai/en/latest/api/python/dgl.data.html#node-prediction-datasets

LP & GLP (MIT License): https://github.com/liqimai/Efficient-SSL

GCN-LPA (MIT License): https://github.com/hwwang55/GCN-LPA

PTA (MIT License): https://github.com/DongHande/PT_propagation_then_training

GCN, GAT, MixHop, GMNN, APPNP, GRAND & DropEdge (Apache-2.0 License): https://github.com/dmlc/dgl/tree/master/examples/pytorch

GraphVAT (without License): https://github.com/fulifeng/GraphAT

GraphMix (without License): https://github.com/vikasverma1077/GraphMix