NodeMixup: Tackling Under-Reaching for Graph Neural Networks

Weigang Lu, Ziyu Guan, Wei Zhao*, Yaming Yang, Long Jin

School of Computer Science and Technology, Xidian University, China
{wglu@stu., zyguan@, ywzhao@mail., yym@, jin@stu.}xidian.edu.cn

Abstract

Graph Neural Networks (GNNs) have become mainstream methods for solving the semi-supervised node classification problem. However, due to the uneven location distribution of labeled nodes in the graph, labeled nodes are only accessible to a small portion of unlabeled nodes, leading to the *under-reaching* issue. In this study, we firstly reveal under-reaching by conducting an empirical investigation on various well-known graphs. Then, we demonstrate that under-reaching results in unsatisfactory distribution alignment between labeled and unlabeled nodes through systematic experimental analysis, significantly degrading GNNs' performance. To tackle under-reaching for GNNs, we propose an architecture-agnostic method dubbed NodeMixup. The fundamental idea is to (1) increase the reachability of labeled nodes by labeled-unlabeled pairs mixup, (2) leverage graph structures via fusing the neighbor connections of intra-class node pairs to improve performance gains of mixup, and (3) use neighbor label distribution similarity incorporating node degrees to determine sampling weights for node mixup. Extensive experiments demonstrate the efficacy of NodeMixup in assisting GNNs in handling under-reaching. The source code is available at https://github.com/WeigangLu/NodeMixup.

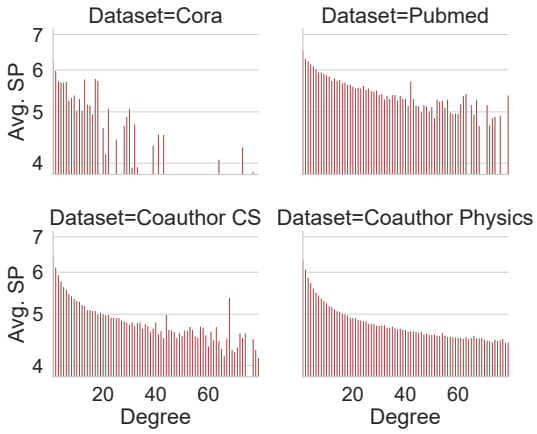

Figure 1: Visual illustrations of under-reaching. Avg. SP is the mean value of the average shortest path length from each unlabeled node to all the labeled ones, where the mean is taken over unlabeled nodes with the same degree. Lower-degree nodes are farther away from labeled nodes while higher-degree nodes tend to be closer to labeled nodes.

Introduction

Graph Neural Networks (GNNs) (Kipf and Welling 2017; Velickovic et al. 2018; Wu et al. 2019; Chen et al. 2020; Klicpera, Bojchevski, and Günnemann 2019; Hamilton, Ying, and Leskovec 2017), which are designed based on the message-passing protocol (Gilmer et al. 2017), have become the mainstream models for dealing with the semi-supervised node classification problem. A recent work (Zheng et al. 2022) reveals the success of GNNs is that the propagation on graphs narrows the distribution gap between labeled and unlabeled data (distribution alignment), thereby benefiting GNNs to make reasonable inferences over unlabeled data. At the training stage, the model is optimized by minimizing the supervised loss function which is defined on labeled nodes. Then, at the inference stage, the well-trained model makes predictions on unlabeled nodes. In other words, a K-layer GNN helps labeled nodes to receive information from k-hop neighbors ($1 \leq k \leq K$) and learns from these labeled nodes to capture a better picture of unlabeled data.

However, by revisiting several commonly-used graphs, we find that nodes with lower (higher) degrees usually stay farther away from (nearer to) labeled nodes, as illustrated in Figure 1. With the restriction of model depth, those lower-degree nodes could hardly transmit information to far-off labeled nodes. Thus, massive unlabeled nodes are hardly known by labeled nodes in popular 2-layer GNN architectures. It leads to incomplete knowledge about unlabeled nodes, hindering distribution alignment. As a result, the trained GNN can only recognize the nodes located near labeled nodes, whereas other unseen-during-training nodes are difficult to be classified. However, in practical scenarios, labeled nodes tend to distribute sparsely in the graph. This phenomenon, namely *under-reaching* (Sun et al. 2022; Di Giovanni et al. 2023; Brüel-Gabrielsson, Yurochkin, and Solomon 2022; Barceló et al. 2020), will be discussed in detail.

There have been several methods attempting to alleviate under-reaching. Intuitively, stacking more GNN layers to al-

*Corresponding author.

low all nodes to propagate more distant information seems to be a direct solution. Nevertheless, it still raises two additional problems, i.e., over-smoothing (Li, Han, and Wu 2018) that induces indistinguishable node representations and over-squashing (Alon and Yahav 2020) that causes distant information loss. To increase reachability through modifying the graph structure, (Sun et al. 2022; Brüel-Gabrielsson, Yurochkin, and Solomon 2022) leverage k-hop positional encodings to add edges between nodes. Unfortunately, they require substantial computational costs in calculating the shortest path between each node pair. Considering practical use, how to develop an effective and flexible method to increase reachability is still a challenging problem.

The key insight to tackle under-reaching is to *improve communications between labeled and unlabeled nodes, facilitating distribution alignment in training*. Recently, mixup (Zhang et al. 2017; Wu et al. 2021; Verma et al. 2021; Wang et al. 2021) techniques have been widely adopted to synthesize additional labeled data via random interpolation between pairs of data points and corresponding labels from original labeled data. The synthesized data can be used as the augmented input for the backbone model. Interestingly, interpolation is similar to the message-passing mechanism since they both essentially perform weighted sum. An intuitive idea is to mix up labeled and unlabeled nodes to enhance their communications. However, the traditional mixup techniques are proposed to expand labeled data but less adept at addressing the under-reaching issue due to the following reasons: (1) The mixed pairs are only sampled from the labeled set which leads to limited access to unlabeled nodes; (2) Traditional mixup methods often employ linear interpolation on data features and labels, which proves less adaptable to the intricate graph topology capturing relationships between nodes in graph-structured data.

Inspired by these insights, we develop NodeMixup, an architecture-agnostic method to tackle under-reaching for GNNs. To improve communications between labeled and unlabeled nodes, we propose cross-set pairing that chooses mixed pairs from labeled and pseudo-labeled unlabeled sets (Lee et al. 2013). Besides, we enhance intra-class interactions by merging neighbor connections among intra-class nodes, and use the standard mixup operation for inter-class node pairs which contributes to characterizing more generalizable classifying boundaries (Zhang et al. 2017). Notably, we propose a novel Neighbor Label Distribution (NLD)-Aware Sampling, which leverages the similarity of neighborhood label distributions along with node degrees to compute sampling weights. It ensures unlabeled nodes with dissimilar/similar neighbor patterns to labeled nodes, as well as nodes with lower degrees, are more likely to be selected for inter/intra-class mixup. NodeMixup enables direct interactions between node pairs and escapes from the restriction of the graph structure or model depth. This simple but effective approach can be applied to any GNN without complex architecture adjustments or significant computational costs.

Contributions. Our main contributions are as follows: (1) We revisit and analyze the under-reaching issue through empirical observations, highlighting its negative impacts on communications between labeled and unlabeled nodes, which hampers distribution alignment. (2) We propose NodeMixup, an architecture-agnostic framework that facilitates direct communications between labeled and unlabeled nodes, overcoming the limitations imposed by the graph structure and effectively alleviating under-reaching for GNNs. (3) We apply NodeMixup on popular GNNs and evaluate it on six real-world graphs. It consistently achieves significant performance gains for GNNs, showing its practicality and generalizability.

Preliminary and Related Works

Notations

We use $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$ to denote an undirected graph with self-loops, where $\mathcal{V} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ is the node set with N nodes, $\mathcal{E}$ is the edge set and $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix. We also have the input feature matrix $X \in \mathbb{R}^{N\times F}$ whose i-th row vector is denoted as $\mathbf{x}_i$, where F is the input dimensionality. Here, we abuse the notation $\mathbf{x}_i$ a bit to denote it as the node index of node i. We define the node label matrix as $Y \in \mathbb{R}^{N\times C}$, where C is the number of classes and $\mathbf{y}_i$ is the one-hot encoding of the label of node i. We divide the data set into labeled set $\mathcal{D}_l = \{(\mathbf{x}_1, \mathbf{y}_1) \cdots, (\mathbf{x}_{N_l}, \mathbf{y}_{N_l})\}$ and unlabeled set $\mathcal{D}_u = \{\mathbf{x}_{N_l+1}, \cdots, \mathbf{x}_N\}$, where $0 < N_l < N$. Specifically, the training set can be divided into C subsets according to different classes, i.e., $\mathcal{V}_l^{(1)}, \cdots, \mathcal{V}_l^{(C)}$. Besides, we assume that each subset contains T labeled nodes.

Related Works

Graph Neural Networks. (Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2017; Velickovic et al. 2018; Xu et al. 2019; Wu et al. 2019; Chen et al. 2020; Klicpera, Bojchevski, and Günnemann 2019; Chien et al. 2021) enable each node to accept the information from neighbors in the range of K hops. The discernible difference is how they aggregate messages from connected nodes at each layer. Assuming $\mathbf{h}^{(l)} \in \mathbb{R}^F$ is a F-dimension representation, the l-th GNN layer $f^{(l)}$ aggregates and transforms neighbor information to produce $\mathbf{h}^{(l+1)} \in \mathbb{R}^F$, which can be generalized as: $\mathbf{h}^{(l+1)} = f^{(l)}(\mathbf{h}^{(l)}, \theta^{(l)}, A)$, where $\theta^{(l)} \in \mathbb{R}^{F\times F}$ is the learnable parameter and σ is an activation function. Without loss of generalizability, we can define a L layer GNN as $G_\theta(\mathbf{x}, A) = (f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)})_\theta(\mathbf{x}, A)$, which is parameterized by θ. Therefore, the cross-entropy loss (Bishop and Nasrabadi 2006) ℓ can be adopt in semi-supervised node classification as:

$$\mathcal{L}_{\text{GNN}}(\mathcal{D}_l, G_\theta, A) = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_l} \ell(G_\theta(\mathbf{x}, A), \mathbf{y}). \tag{1}$$

Mixup. (Zhang et al. 2017) proposes the mixup technique that mixes features and corresponding labels of pairs of labeled samples to generate virtual training data. Because of the simplicity and effectiveness of mixup, some works (Crisostomi et al. 2022; Park, Shim, and Yang 2022; Guo and Mao 2021; Navarro and Segarra 2022) adapt it to the graph domain. However, they only focus on the graph classification problem and can not be directly applied to the node-level task. To overcome the node classification problem, (Wu et al. 2021; Wang et al. 2021; Verma et al. 2021) develop improved mixup mechanisms to enhance GNNs. Formally, assuming both $\mathbf{a}$

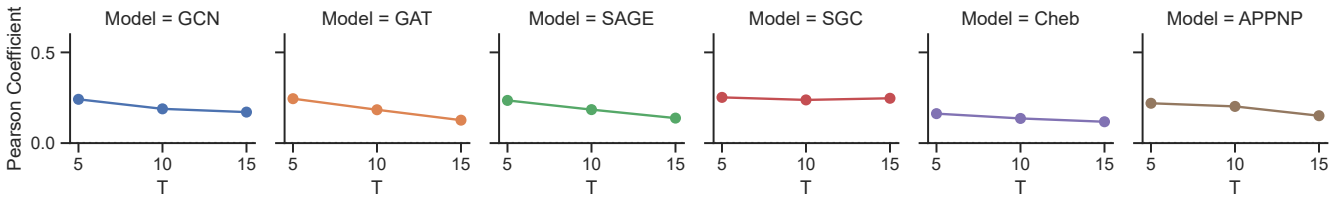

Figure 2: The correlation between prediction scores on actual classes and RC values on CORA dataset. $T = m$ represents m labeled nodes per class. The Pearson coefficient shows a positive correlation between prediction scores and RC, demonstrating that larger reachability yields better performance. As labeled nodes decrease which (indirectly) suggests poor reachability, a more significant positive correlation can be observed.

and $\mathbf{b}$ are either feature vectors or one-hot label vectors, the mixup operation is defined as:

$$\mathcal{M}_\lambda(\mathbf{a}, \mathbf{b}) = \lambda \mathbf{a} + (1 - \lambda)\mathbf{b}, \tag{2}$$

where λ is sampled from $\mathrm{Beta}(\alpha, \alpha)$ distribution and $\alpha \geq 0$.

Under-reaching, Over-smoothing, and Over-squashing. They are the graph-specific information shortage issues in the context of semi-supervised node classification. From a topological perspective, prior researchs have described the under-reaching issue (Sun et al. 2022; Brüel-Gabrielsson, Yurochkin, and Solomon 2022; Barceló et al. 2020) as a node's inability to be aware of nodes that are farther away than the number of layers. However, directly increasing the number of layers gives rise to the over-smoothing issue (Li, Han, and Wu 2018; Oono and Suzuki 2020; Lu et al. 2021), where node representations become indistinguishable, severely impacting prediction performance. Even with the resolution of over-smoothing by enlarging receptive fields, the over-squashing issue (Alon and Yahav 2020; Di Giovanni et al. 2023; Topping et al. 2021) still persists. This issue pertains to the loss of information from distant nodes due to message propagation over the graph-structured data, where features from exponentially-growing neighborhoods are compressed into fixed-length node vectors. Drawing inspiration from the distribution shift concept (Shimodaira 2000), where the difference between labeled and unlabeled distributions affects the model's generalization, we identify the under-reaching issue as a lack of communication between labeled and unlabeled nodes, leading to difficulties in making accurate inferences over unlabeled nodes. Thus, this issue represents a more generalized graph-specific challenge about how to improve communication between labeled and unlabeled nodes rather than propagate distant information in the semi-supervised node classification regime.

Method

In this section, we first explain our motivation by introducing under-reaching. Then, we present our proposed NodeMixup framework in order to increase the reachability of GNNs.

Motivation: Understanding Under-reaching

How Does Under-reaching Impact on GNNs? Nodes far from labeled nodes lack supervision information because the influence of labeled nodes decreases with topology distance (Buchnik and Cohen 2018). With the restriction of model depth, nodes at r-hop away ($r > K$) from labeled nodes can not be reached when a K-layer model (e.g., GCN) is used. Since the supervised loss function is purely defined on labeled nodes, the optimization might be misled by the inadequate received information. We define $d_\mathcal{G}(i, j)$ as the shortest path length between node i and node j. To measure reachability for each node, we first introduce the reaching coefficient (**RC**) from (Sun et al. 2022) as:

$$\mathrm{RC}_i = \frac{1}{|\mathcal{D}_l|} \sum_{j \in \mathcal{D}_l} \left(1 - \frac{\log |d_\mathcal{G}(i, j)|}{\log D_\mathcal{G}}\right), \tag{3}$$

where $D_\mathcal{G}$ represents the diameter of graph $\mathcal{G}$, and $d_\mathcal{G}(i, j) = D_\mathcal{G}$ when node i and node j do not belong to the same connected component. A larger RC value (scaled to $[0, 1)$) indicates greater reachability of node i. In Figure 2, we visualize the correlation between prediction scores on actual classes and RC values on CORA dataset using different GNNs, i.e., **GCN** (Kipf and Welling 2017), **GAT** (Velickovic et al. 2018), **APPNP** (Klicpera, Bojchevski, and Günnemann 2019), **Cheb**Net (Defferrard, Bresson, and Vandergheynst 2016), and Graph**SAGE** (Hamilton, Ying, and Leskovec 2017). Across all experiments, we vary the number of training nodes per class ($T \in 5, 10, 15$). We can observe positive correlations (Pearson Coefficient larger than 0) in all the cases, which further demonstrates the benefit of better reachability. Additionally, as T decreases, indicating that the reachability declines, the positive correlation becomes more significant. It is because only a few unlabeled nodes can be seen during training. It is easier for GNNs to classify correctly those unlabeled nodes located nearby labeled nodes.

Why Does Under-reaching Fail GNNs? A recent study (Zheng et al. 2022) underscores that GNN success hinges on aligning the distributions of labeled and unlabeled nodes. The propagation enables labeled nodes to receive information from unlabeled nodes to narrow the distribution gap between labeled and unlabeled nodes. Intuitively, it would facilitate inference as the two distributions get close. Inspired by this, to further understand the negative impact of under-reaching, we here investigate the distance between the labeled and unlabeled distribution at different levels of RC. We first briefly introduce the centered kernel alignment (CKA) (Kornblith et al. 2019) metric[1], which is widely used to measure the representation similarity. Supposing $Z_l, Z_u \in \mathbb{R}^{m \times n}$ are

[1]Please refer to (Kornblith et al. 2019; Zheng et al. 2022) for more details about CKA.

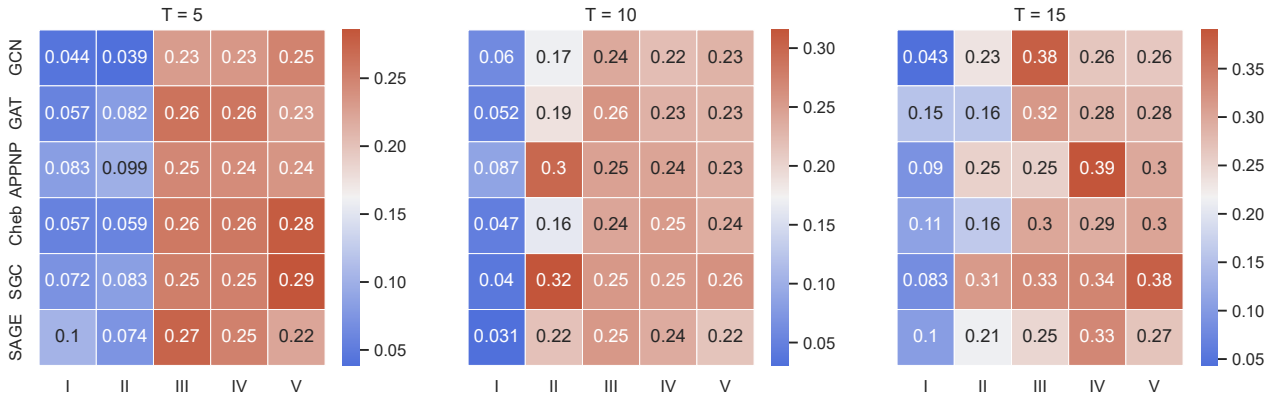

Figure 3: The heat map of CKA between labeled and unlabeled distributions learned by different GNNs using CORA. A larger CKA value indicates more similar representation distributions between labeled and unlabeled nodes. I, ..., V represent unlabeled nodes with various reachability arranged in ascending order. Larger reachability is in favor of narrowing the distribution gap.

the representations (learned by an arbitrary GNN) sampled from labeled and unlabeled nodes, the distribution similarity between Z_l and Z_u is measured by CKA as:

$$\text{CKA}(Z_l, Z_u) = \frac{\|Z_u^T Z_l\|_F}{\|Z_l Z_l^T\|_F \|Z_u Z_u^T\|_F}. \tag{4}$$

A larger CKA (scaled to $[0, 1]$) implies a higher similarity. Secondly, we divide unlabeled nodes from CORA into five subsets $\mathcal{D}_{\text{I}}$, ..., $\mathcal{D}_{\text{V}}$ according to different interval ranges of RC, i.e., range I, ..., range V[2]. Finally, we calculate CKA between Z_l and representations of unlabeled nodes from $\mathcal{D}_{\text{I}}$, ..., $\mathcal{D}_{\text{V}}$ learned by different GNNs. To do so, we sample the same number of nodes in each pair of two sets ($\mathcal{D}_l$, $\mathcal{D}_{\text{I}}$), ..., ($\mathcal{D}_l$, $\mathcal{D}_{\text{V}}$), and the number is determined by the minimum element number of each set pair. We visualize the results in Figure 3, in which each block represents the CKA value between labeled and unlabeled distributions learned by various GNNs. We can see that lower reachability (e.g., range I and II) tends to result in a larger distribution gap while higher reachability can bridge the gap.

How Do We Alleviate Under-reaching? From the above analysis, we can see that under-reaching hinders the distribution alignment since the labeled nodes can only reach a small part of unlabeled nodes. A straightforward idea for assisting labeled nodes in reaching more unlabeled nodes is to add edges between them or stack more GNN layers. However, the edge-adding strategy could induce prohibitive computation costs for finding globally friendly neighbors (Brüel-Gabrielsson, Yurochkin, and Solomon 2022; Sun et al. 2022) or lead to noisy graphs without sufficient supervision (Sun et al. 2022). Besides, deepening GNNs leads to over-smoothing or over-squashing. Based on these limitations, we intend to develop an efficient framework for various GNNs to tackle under-reaching. Recently, interpolation-based methods (Zhang et al. 2017; Wu et al. 2021; Wang et al. 2021; Verma et al. 2021) show great effectiveness and flexibility in augmenting labeled data. Inspired by this, we propose NodeMixup, which mixes labeled and unlabeled data to increase reachability for GNNs. We present the detailed methodology of our NodeMixup in the following section.

[2]Let RC_{max} be the maximum value of RC in all the unlabeled nodes. Then, we define range I as $[0, {}^{\text{RC}}max/5]$, range II as $({}^{\text{RC}}max/5, {}^{2\text{RC}}max/5]$, ..., and range V as $({}^{4\text{RC}}max/5, \text{RC}_{max}]$.

Methodology

We start by giving a high-level overview of NodeMixup and present its pipeline in Figure 4. Our NodeMixup is designed as a data augmentation technique that can be easily applied to various GNNs. At each iteration, NodeMixup samples node pairs from labeled and unlabeled nodes. This cross-set pairing ensures labeled nodes reach more unlabeled nodes to bridge the distribution gap. Then, we use the pseudo labels (Lee et al. 2013) with a confidence threshold γ for unlabeled nodes and construct a pseudo-labeled data set $\mathcal{D}_{pl}$. Guided by pseudo labels, we mix nodes via different mixup operations, i.e., intra-class and inter-class mixup. Then, the intra-class-mixup nodes with the mixed adjacency matrix $\tilde{A}$ and inter-class-mixup nodes with an identity matrix $E \in \mathbb{R}^{N_l \times N_l}$ are used as inputs for the GNN model, where N_l is the number of labeled nodes. Note that a GNN fed with an identity matrix equals a MLP. Besides, to enhance the mixup effect, we propose a neighbor label distribution (NLD)-aware sampling strategy. It ensures that nodes with similar neighbor patterns and lower degrees are more likely to be selected. Finally, the model is optimized via minimizing the loss $\mathcal{L}$ from the combination of $\mathcal{L}_{\text{GNN}}$, $\mathcal{L}_{\text{intra-M}}$, and $\mathcal{L}_{\text{inter-M}}$ (defined by Eq. (1), Eq. (8), and Eq. (10), respectively) from each branch as follows:

$$\mathcal{L} = \mathcal{L}_{\text{GNN}} + \lambda_{\text{intra}} \mathcal{L}_{\text{intra-M}} + \lambda_{\text{inter}} \mathcal{L}_{\text{inter-M}}, \tag{5}$$

where two hyper-parameters $\lambda_{\text{intra}}, \lambda_{\text{inter}} \in (0, 1]$ control the regularization effect of NodeMixup.

Intra-class Mixup. We sample and mix nodes with the equal class from $\mathcal{D}_l$ and $\mathcal{D}_{pl}$ via Eq. (2). Suppose $\mathbf{x}_i \in \mathcal{D}_l$ and $\mathbf{x}_j \in \mathcal{D}_{pl}$ are classified as $\mathbf{y}_i$ and $\hat{\mathbf{y}}_j$, where $\hat{\mathbf{y}}_j$ is the pseudo label of node j. We have the mixed intra-class data set $\mathcal{D}_{\text{intra}}$ as:

$$\begin{aligned}\mathcal{D}_{\text{intra}} = \{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) | \tilde{\mathbf{x}}_i = \mathcal{M}_\lambda(\mathbf{x}_i, \mathbf{x}_j), \tilde{\mathbf{y}}_i = \mathcal{M}_\lambda(\mathbf{y}_i, \hat{\mathbf{y}}_j), \\ (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_l, (\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}_{pl}, \mathbf{y}_i = \hat{\mathbf{y}}_j\}.\end{aligned} \tag{6}$$

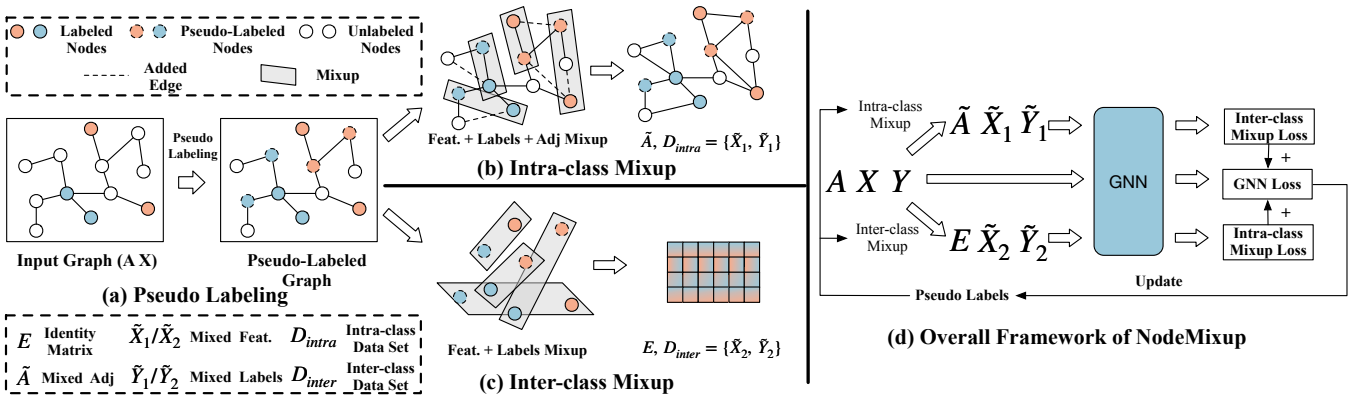

Figure 4: The pipeline of NodeMixup. The NLD-aware sampling is omitted in this figure.

Furthermore, we also fuse their topological information together since these intra-class pairs tend to share similar neighbor distribution, which prompts intra-class similarity. Thus, we mix the topological information of node i and node j to generate a new adjacency matrix $\tilde{A}$ as:

$$\begin{aligned} &\text{Row mixup:} \quad \tilde{A}_{i,:} = \mathcal{M}_\lambda(A_{i,:}, A_{j,:}), \\ &\text{Column mixup:} \quad \tilde{A}_{:,i} = \mathcal{M}_\lambda(A_{:,i}, A_{:,j}). \end{aligned} \tag{7}$$

Here, $A_{i,:}$ and $A_{:,i}$ represents the i-th row and column vector of A. Finally, we calculate the intra-class mixup loss $\mathcal{L}_{\text{intra-M}}$ with a GNN model G_θ as follows:

$$\mathcal{L}_{\text{intra-M}} = \mathcal{L}_{\text{GNN}}(\mathcal{D}_{\text{intra}}, G_\theta, \tilde{A}) = \mathop{\mathbb{E}}_{(\tilde{\mathbf{x}},\tilde{\mathbf{y}})\sim\mathcal{D}_{\text{intra}}} \ell(G_\theta(\tilde{\mathbf{x}}, \tilde{A}), \tilde{\mathbf{y}}). \tag{8}$$

Inter-class Mixup. The mixup operation facilitates a linear behavior of the model between selected nodes, thus improving its generalization (Zhang et al. 2017). Besides, empirical observations indicate that interpolating between inputs with different labels results in decision boundaries that transition linearly from one class to another, reducing prediction errors. Thus, the inter-class mixup operation enhances the model's ability to distinguish boundaries between different classes. By combining nodes from different classes, the model can effectively learn shared features and differences between classes, thereby improving its generalization capability. Based on this, we generate the mixed inter-class data set $\mathcal{D}_{\text{inter}}$ as:

$$\begin{aligned} \mathcal{D}_{\text{inter}} = &\{(\tilde{\mathbf{x}}_i, \tilde{\mathbf{y}}_i) | \tilde{\mathbf{x}}_i = \mathcal{M}_\lambda(\mathbf{x}_i, \mathbf{x}_j), \tilde{\mathbf{y}}_i = \mathcal{M}_\lambda(\mathbf{y}_i, \hat{\mathbf{y}}_j), \\ &(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}_l, (\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}_{pl}, \mathbf{y}_i \neq \hat{\mathbf{y}}_j\}. \end{aligned} \tag{9}$$

Furthermore, we argue that inter-class interaction could probably lead to noisy learning. Therefore, we feed the backbone GNN with an identity matrix E to block the message passing between inter-class nodes. Similar to Eq. (8), we define the inter-class mixup loss $\mathcal{L}_{\text{inter-M}}$ as follows:

$$\mathcal{L}_{\text{inter-M}} = \mathcal{L}_{\text{GNN}}(\mathcal{D}_{\text{inter}}, G_\theta, E) = \mathop{\mathbb{E}}_{(\tilde{\mathbf{x}},\tilde{\mathbf{y}})\sim\mathcal{D}_{\text{inter}}} \ell(G_\theta(\tilde{\mathbf{x}}, E), \tilde{\mathbf{y}}). \tag{10}$$

NLD-aware Sampling. In a graph, intra/inter-class nodes often form cohesive/incohesive communities or clusters, leading to similar/dissimilar neighbor distributions. The characteristics of a node are not only determined by itself but also influenced by its neighbors. To capture such neighborhood information, we adapt Neighborhood Label Distribution (NLD) (Zhu et al.) and define it in the next.

Definition 1 *(Neighborhood Label Distribution (NLD)) Given $\bar{Y}$ is the label matrix for all the nodes (the label of unlabeled nodes are their predictions), the neighborhood label distribution of node i is defined as $\mathbf{q}_i = \frac{1}{|\mathcal{N}_i|}\sum_{v\in\mathcal{N}_i} \bar{Y}_{v,:}$, where $\mathcal{N}_i$ is the neighbor set of node i.*

Besides, we employ the sharpening (Berthelot et al. 2019) trick to enhance the contrast between class probabilities in NLD. Specifically, we calculate the new distribution $\mathbf{q}'_{ij}$ as following:

$$\mathbf{q}'_{ij} = \frac{\mathbf{q}_i^{1/\tau}}{\sum_{k=1}^{C} \mathbf{q}_{ik}^{1/\tau}}, \tag{11}$$

where $0 < \tau \leq 1$ is the temperature parameter that controls the sharpness of the distribution. As $\tau \to 0$, it leads to a sharper probability distribution. Then, we use the cosine similarity s_{ij} between $\mathbf{q}'_i$ and $\mathbf{q}'_j$ to determine the likelihood of node i and node j sharing a similar neighbor pattern, where

$$s_{ij} = \frac{\mathbf{q'}_i^T \mathbf{q'}_j}{\|\mathbf{q'}_i\|_2 \|\mathbf{q'}_j\|_2} \tag{12}$$

Additionally, we take node degrees into account since low-degree nodes suffer more from under-reaching. Therefore, for a labeled node i, the sampling weight w_{ij} of an unlabeled node j to be mixed is defined as follows:

$$w_{ij} = \begin{cases} \frac{1}{1+\beta_d d_j} e^{\beta_s s_{ij}}, & \text{if} \quad \mathbf{y}_i = \hat{\mathbf{y}}_j, \\ \frac{1}{1+\beta_d d_j} e^{-\beta_s s_{ij}}, & \text{if} \quad \mathbf{y}_i \neq \hat{\mathbf{y}}_j, \end{cases} \tag{13}$$

where d_j is the degree of node j, and $\beta_s > 0$ and $\beta_d > 0$ control the strength of NLD similarity and node degree, respectively. The term $1/1+\beta_d d_j$ ensures that the influence of node degree on the sampling weight is monotonic, leading to a higher sampling weight for low-degree nodes and vice versa. This sampling weight calculation balances the effect of node similarity and node degree, resulting in a reasonable and interpretable mechanism for the Mixup operation between nodes in the graph.

Models	Strategy	CORA	CITESEER	PUBMED	CS	PHYSICS
GCN	Original	81.57±0.4	70.50±0.6	77.91±0.3	91.24±0.4	92.56±1.3
	UPS	82.35±0.4	72.82±0.6	78.45±0.4	91.62±0.3	93.01±0.3
	PASTEL	81.97±0.6	71.32±0.4	78.92±0.2	91.76±0.6	$>$ 3 days
	ReNode	81.98±0.6	69.48±0.4	78.13±0.7	91.32±0.1	OOM
	GraphMix	82.29±3.7	**74.55±0.5**	**82.82±0.5**	91.90±0.2	90.43±1.7
	NodeMixup	**83.47±0.3**	74.12±0.3	81.16±0.2	**92.69±0.4**	**93.97±0.4**
GAT	Original	82.04±0.6	71.82±0.8	78.00±0.7	90.52±0.4	91.97±0.6
	UPS	82.17±0.5	72.97±0.7	78.56±0.9	91.26±0.4	92.45±1.1
	PASTEL	82.21±0.3	72.35±0.8	78.74±0.9	90.31±0.2	$>$ 3 days
	ReNode	81.88±0.7	71.73±1.2	79.68±0.5	88.36±0.5	OOM
	GraphMix	82.76±0.6	73.04±0.5	78.82±0.4	90.57±1.0	92.90±0.4
	NodeMixup	**83.52±0.3**	**74.30±0.1**	**81.26±0.3**	**92.69±0.2**	**93.87±0.3**
APPNP	Original	80.03±0.5	70.30±0.6	78.67±0.2	91.79±0.5	92.36±0.8
	UPS	81.24±0.6	71.02±0.7	78.69±0.7	91.77±0.3	92.31±0.5
	PASTEL	81.56±0.3	70.68±0.8	78.39±0.2	91.98±0.4	$>$ 3 days
	ReNode	81.12±0.2	70.04±0.8	78.58±0.3	91.99±0.2	OOM
	GraphMix	82.98±0.4	70.26±0.4	78.73±0.4	91.53±0.6	94.12±0.1
	NodeMixup	**83.54±0.4**	**75.12±0.3**	**79.93±0.1**	**92.82±0.2**	**94.34±0.2**
GraphSAGE	Original	78.12±0.3	68.09±0.8	77.30±0.7	91.01±0.9	93.09±0.4
	UPS	81.83±0.3	70.29±0.6	77.82±0.6	91.35±0.4	93.20±0.4
	PASTEL	78.58±0.6	70.31±0.3	78.26±0.7	91.77±0.6	$>$ 3 days
	ReNode	76.48±1.0	70.79±0.9	78.67±1.2	89.61±0.7	OOM
	GraphMix	80.09±0.8	70.97±1.2	79.85±0.4	91.55±0.3	93.25±0.3
	NodeMixup	**81.93±0.2**	**74.12±0.4**	**79.97±0.5**	**91.97±0.2**	**94.76±0.2**

Table 1: Node Classification Results on Medium-scale Graphs.

Complexity. The mixup operation (Eq. (2)), encompassing both nodes and labels, centers on matrix addition, which incurs negligible computational overhead due to its parallelizability across vectors. The predominant computational expenditure lies within the NLD-aware sampling phase, which unfolds in three sequential steps: NLD calculation, NLD cosine similarity evaluation, and sampling weight computation. The computational cost of NLD computation is determined by the number of edges in the graph, rendering it $\mathcal{O}(|\mathcal{E}|)$. When determining the NLD-similarity between labeled and selected unlabeled nodes, the complexity equates to $\mathcal{O}(|\mathcal{D}_{pl}|^2)$, where $|\mathcal{D}_{pl}|$ is the subset of confident nodes used for mixup. Subsequently, the computation of sampling weights for these unlabeled nodes involves $\mathcal{O}(|\mathcal{D}_{pl}|)$ operations. Overall, the integration of NodeMixup within GNNs introduces a marginal computational burden $\mathcal{O}(|\mathcal{E}| + |\mathcal{D}_{pl}|^2 + |\mathcal{D}_{pl}|)$ because $|\mathcal{D}_{pl}|$ is much smaller than $|\mathcal{E}|$, preserving the overall efficiency of the model.

Experiments

We evaluate NodeMixup on the semi-supervised node classification task. We use five medium-scale datasets, i.e., CORA, CITESEER, and PUBMED (Yang, Cohen, and Salakhudinov 2016), COAUTHOR CS and COAUTHOR PHYSICS (Shchur et al. 2018), and a large-scale graph, i.e., OGBN-ARXIV (Hu et al. 2020). Besides, we do detailed ablation experiments to probe into the design of NodeMixup.

Evaluation on Medium-scale Graphs

Settings. Similar to the experimental setup of (Sun et al. 2022), we choose GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), APPNP (Klicpera, Bojchevski, and Günnemann 2019), and GraphSAGE (Hamilton, Ying, and Leskovec 2017) as backbone models. For the comparing strategies, we choose PASTEL (Sun et al. 2022), ReNode (Chen et al. 2021), and GraphMix (Verma et al. 2021). PASTEL and ReNode address the topology imbalance issue. GraphMix is a data augmentation method that mixes hidden node representations. UPS (Rizve et al. 2021) is a state-of-the-art (SOTA) pseudo-labeling method that generates pseudo-labels with uncertainty-aware selection. We simply apply grid search for all the models since we do not intend to achieve new SOTA performance. We keep the parameters of comparing strategies as they are in the original papers or reference implementations. For our proposed NodeMixup, which is implemented by PyTorch Geometric Library (Fey and Lenssen 2019) with the Adam optimizer (Kingma and Ba 2015), we search both λ_{inter} and λ_{intra} in $\{1, 1.1, \cdots, 1.5\}$, β_d and β_s in $\{0.5, 1, 1.5, 2\}$, and γ in $\{0.5, 0.7, 0.9\}$. All the experiments are conducted on an NVIDIA GTX 1080Ti GPU.

Results. For CORA, CITESEER, and PUBMED datasets, we stick to the public splits (20 nodes per class for training, 1000 nodes for validation, and 500 nodes for testing) used in (Yang, Cohen, and Salakhudinov 2016). For COAUTHOR CS and COAUTHOR PHYSICS, we follow the splits in (Shchur

et al. 2018), i.e., 20 labeled nodes per class as the training set, 30 nodes per class as the validation set, and the rest as the test set. The overall results are reported in Table 1 and all the results are obtained over 10 different runs. We can observe significant improvement made by NodeMixup in all the graphs and models. In addition, it is worth noting that PASTEL and ReNode tend to have higher computational costs or increased complexity compared to other methods. It can limit their practical application, particularly when dealing with large graphs. The best result w.r.t each backbone model is shown in boldface. OOM stands for "out-of-memory."

Evaluation on the Large-scale Graph

Settings. We test NodeMixup's effectiveness on the challenging large-scale graph OGBN-ARXIV to enhance GNN performance. Baseline models include GCN, GraphSAGE, JKNet (Xu et al. 2018), and GCNII (Chen et al. 2020). Data splits come from (Hu et al. 2020). Due to GPU limitations, we use GCN and GraphSAGE as backbone models, employing identical configurations as outlined in the source code from (Hu et al. 2020).

Results. The results are provided in Table 2. Notably, NodeMixup demonstrates its efficacy in bolstering the performance of fundamental GNN models, such as GCN and GraphSAGE. This enhancement remains pronounced even in the context of large-scale graphs, effectively surpassing the performance of more intricate deep GNN models, i,e, GCNII. This observation underscores the ability of NodeMixup to yield substantial gains in predictive accuracy while maintaining efficiency. The baseline performance are obtained from the OGB Leaderboards (https://ogb.stanford.edu/docs/leader_nodeprop/).

Models	Accuracy (%) Test	Valid
GCN	71.74±0.2	73.00±0.1
GraphSAGE	71.49±0.2	72.77±0.1
JKNet	72.19±0.2	73.35±0.1
GCNII	72.74±0.1	-
GCN+NodeMixup	**73.46±0.2**	74.13±0.2
GraphSAGE+NodeMixup	73.24±0.2	**74.14±0.1**

Table 2: Node Classification on OGBN-ARXIV.

Ablation Analysis

NodeMixup comprises three key modules: cross-set mixup, class-specific mixup, and NLD-aware sampling. In Table 3, we conduct an investigation into each design utilizing the CORA dataset and GCN as the underlying model. Note that, except for the design under examination, NodeMixup remains unchanged. Our observations reveal several fascinating properties, which are outlined below.

Cross-set Pairing. The traditional mixup (Zhang et al. 2017) approach samples mixed pairs exclusively from the labeled set, making it function primarily as a data augmentation technique without addressing the under-reaching problem. Consequently, it does not generate significant improvements since a considerable number of unlabeled nodes remain inaccessible. In contrast, our labeled-unlabeled (LU) pairing, as opposed to labeled-labeled (LL) and labeled-all (LA) pairings, allows GNNs to leverage information from a larger pool of unlabeled nodes. This facilitates distribution alignment and ultimately leads to enhanced performance.

Class-specific Mixup. Table 3 reveals that both inter-class (IE) and intra-class (IR) interpolations contribute to performance enhancements when applied between nodes. Nonetheless, solely interpolating nodes with the same label does not significantly improve the backbone model, aligning with findings in (Zhang et al. 2017). It is because such an approach fails to take advantage of the mutually beneficial neighbor information among intra-class nodes.

NLD-aware Sampling. To investigate the low-degree biased sampling strategy employed in NodeMixup, we conducted a comparison with random sampling. Our findings consistently demonstrated that our sampling strategy consistently outperforms random sampling. This can be attributed to two factors: (1) NLD similarity guarantees a higher probability of mixing nodes with similar neighbor patterns, thereby enhancing information alignment and generalization; (2) assigning larger sampling weights to low-degree nodes, which are typically difficult to access in a general scenario.

GCN		81.57±0.4
GCN + vanilla mixup		81.67±0.3
Cross-set Pairing	LL	81.85±0.2
	LA	81.97±0.6
	LU	**83.64±0.5**
Class-specific Mixup	IE	81.97±0.3
	IR	82.06±0.3
	IE+IR	**83.64±0.5**
NLD-aware Sampling	Random	82.41±0.2
	NLD-aware	**83.64±0.5**

Table 3: Ablation Analysis.

Conclusion

In this work, we take an in-depth look at the under-reaching issue through comprehensive empirical investigations and extensive experimental analysis on widely-used graphs. The issue widely exists in various GNN models and degrades their performance. Our findings shed light on the fact that the reachability of GNNs should be further strengthened. To this purpose, we propose an architecture-agnostic framework, NodeMixup, to improve reachability for GNNs. It effectively addresses the under-reaching issue, helping GNNs to achieve better performance using different graphs.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62133012, 61936006, 62073255, and 62303366, in part by the Innovation Capability Support Program of Shaanxi under Grant 2021TD-05, and in part by the Key Research and Development Program of Shaanxi under Grant 2020ZDLGY04-07.

References

Alon, U.; and Yahav, E. 2020. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*.

Barceló, P.; Kostylev, E. V.; Monet, M.; Pérez, J.; Reutter, J.; and Silva, J.-P. 2020. The logical expressiveness of graph neural networks. In *8th International Conference on Learning Representations (ICLR 2020)*.

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

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Brüel-Gabrielsson, R.; Yurochkin, M.; and Solomon, J. 2022. Rewiring with positional encodings for graph neural networks. *arXiv preprint arXiv:2201.12674*.

Buchnik, E.; and Cohen, E. 2018. Bootstrapped graph diffusions: Exposing the power of nonlinearity. In *Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems*, 8–10.

Chen, D.; Lin, Y.; Zhao, G.; Ren, X.; Li, P.; Zhou, J.; and Sun, X. 2021. Topology-imbalance learning for semi-supervised node classification. *Advances in Neural Information Processing Systems*, 34: 29885–29897.

Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, 1725–1735. PMLR.

Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. *ICLR*.

Crisostomi, D.; Antonelli, S.; Maiorca, V.; Moschella, L.; Marin, R.; and Rodolà, E. 2022. Metric Based Few-Shot Graph Classification. *arXiv preprint arXiv:2206.03695*.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Di Giovanni, F.; Giusti, L.; Barbero, F.; Luise, G.; Lio, P.; and Bronstein, M. 2023. On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology. *arXiv preprint arXiv:2302.02941*.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*, 1263–1272. PMLR.

Guo, H.; and Mao, Y. 2021. ifmixup: Towards intrusion-free graph mixup for graph classification. *arXiv e-prints*, arXiv–2110.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

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

Kingma, D.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Kipf, N. T.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. *international conference on learning representations*.

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

Kornblith, S.; Norouzi, M.; Lee, H.; and Hinton, G. 2019. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, 3519–3529. PMLR.

Lee, D.-H.; et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 896.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

Lu, W.; Zhan, Y.; Guan, Z.; Liu, L.; Yu, B.; Zhao, W.; Yang, Y.; and Tao, D. 2021. SkipNode: On Alleviating Over-smoothing for Deep Graph Convolutional Networks. *arXiv preprint arXiv:2112.11628*.

Navarro, M.; and Segarra, S. 2022. GraphMAD: Graph Mixup for Data Augmentation using Data-Driven Convex Clustering. *arXiv preprint arXiv:2210.15721*.

Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. *ICLR*.

Park, J.; Shim, H.; and Yang, E. 2022. Graph transplant: Node saliency-guided graph mixup with local structure preservation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 7966–7974.

Rizve, M. N.; Duarte, K.; Rawat, Y. S.; and Shah, M. 2021. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*.

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

Shimodaira, H. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2): 227–244.

Sun, Q.; Li, J.; Yuan, H.; Fu, X.; Peng, H.; Ji, C.; Li, Q.; and Yu, P. S. 2022. Position-aware structure learning for graph

topology-imbalance by relieving under-reaching and over-squashing. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 1848–1857.

Topping, J.; Giovanni, D. F.; Chamberlain, P. B.; Dong, X.; and Bronstein, M. M. 2021. Understanding over-squashing and bottlenecks on graphs via curvature. *ICLR 2022*.

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

Verma, V.; Qu, M.; Kawaguchi, K.; Lamb, A.; Bengio, Y.; Kannala, J.; and Tang, J. 2021. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, 10024–10032.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, 3663–3674.

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

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

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? *international conference on learning representations*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, 5453–5462. PMLR.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, 40–48. PMLR.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

Zheng, Q.; Xia, X.; Zhang, K.; Kharlamov, E.; and Dong, Y. 2022. On the distribution alignment of propagation in graph neural networks. *AI Open*, 3: 218–228.

Zhu, J.; Yan, Y.; Heimann, M.; Zhao, L.; Akoglu, L.; and Koutra, D. ???? Heterophily and Graph Neural Networks: Past, Present and Future. *Data Engineering*, 10.