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.

Introduction

Graph Neural Networks (GNNs) (Kipf and Welling 2017; Velickovic et al. 2018; Wu et al. 2019; Chen et al. 2020; Klicpera, Bojchevski, and Günemann 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. 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 ≤ k ≤ 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-Gabrielson, 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-
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; Bruel-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 overcome 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 $G = \{V, E, A\}$ to denote an undirected graph with self-loops, where $V = \{x_1, \cdots, x_N\}$ is the node set with $N$ nodes, $E$ 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 $x_i$, where $F$ is the input dimensionality. Here, we abuse the notation $x_i$ 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 $y_i$ is the one-hot encoding of the label of node $i$. We divide the data set into labeled set $D_l = \{(x_1, y_1), \cdots, (x_{N_l}, y_{N_l})\}$ and unlabeled set $D_u = \{(x_{N_l+1}, \cdots, x_N)\}$, where $0 < N_l < N$. Specifically, the training set can be divided into $Q$ subsets according to different classes, i.e., $V_l^{(1)}, \cdots, V_l^{(Q)}$. 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.

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 a
and b are either feature vectors or one-hot label vectors, the mixup operation is defined as:
\[
M_\lambda(a, b) = \lambda a + (1 - \lambda)b,
\]
where \(\lambda\) is sampled from \(\text{Beta}(\alpha, \alpha)\) 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 research has 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 cannot 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 \(d_G(i, j)\) 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:
\[
\text{RC}_i = \frac{1}{|D_i|} \sum_{j \in D_i} \left(1 - \frac{\log |d_G(i, j)|}{\log |D_G|}\right),
\]
where \(D_G\) represents the diameter of graph \(G\), and \(d_G(i, j) = D_G\) 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 (Veličković et al. 2018), APPNP (Klicpera, Bojchevski, and Günnemann 2019), ChebNet (Defferrard, Bresson, and Vandergheynst 2016), and GraphSAGE (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

\(^{1}\text{Please refer to (Kornblith et al. 2019; Zheng et al. 2022) for more details about CKA.}\)
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:

\[
\text{CKA}(Z_l, Z_u) = \frac{||Z_l^T Z_u||_F}{||Z_l^T||_F ||Z_u^T||_F}.
\]  

(4)

A larger CKA (scaled to \([0, 1]\)) implies a higher similarity. Secondly, we divide unlabeled nodes from \( \text{CORA} \) into five subsets \( D_1, \ldots, D_5 \) according to different intervals of \( RC \), i.e., range I, \( \ldots \), range V. Finally, we calculate CKA between \( Z_l \) and representations of unlabeled nodes from \( D_1, \ldots, D_5 \) learned by different GNNs. To do so, we sample the same number of nodes in each pair of two sets \( (D_l, D_l), (D_l, D_u), (D_u, D_u) \), 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 among labeled nodes which 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 (Briel-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 improve reachability for GNNs. We present the detailed methodology of our NodeMixup in the following section.

**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 \( \gamma \) for unlabeled nodes and construct a pseudo-labeled data set \( D_{pl} \). 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 \( \hat{A} \) and inter-class-mixup nodes with an identity matrix \( E \) 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:

\[
\mathcal{L} = \mathcal{L}_{\text{GNN}} + \lambda_{\text{intra}} \mathcal{L}_{\text{intra-M}} + \lambda_{\text{inter}} \mathcal{L}_{\text{inter-M}},
\]  

(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 \( D_l \) and \( D_{pl} \) via Eq. (2). Suppose \( x_i \in D_l \) and \( x_j \in D_{pl} \) are classified as \( y_i \) and \( y_j \), where \( y_j \) is the pseudo label of node \( j \). We have the mixed intra-class data set \( D_{\text{intra}} \) as:

\[
D_{\text{intra}} \equiv \{ (\tilde{x}_i, \tilde{y}_i) | \tilde{x}_i = M_{\lambda}(x_i, x_j), \tilde{y}_i = M_{\lambda}(y_i, y_j), (x_i, y_i) \in D_l, (x_j, y_j) \in D_{pl}, y_i = y_j \}.
\]  

(6)
Furthermore, we also fuse their topological information together since these in-tra-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 \( \hat{A} \) as:

\[
\hat{A}_{ij} = M_\lambda(A_{ij}, A_{ji}),
\]

(7)

Row mixup: \( \hat{A}_{ii} = M_\lambda(A_{ii}, A_{ji}) \),

Column mixup: \( \hat{A}_{ij} = M_\lambda(A_{ii}, A_{ji}) \).

Here, \( A_{ij} \) and \( A_{ji} \) 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 \( G_\theta \) as follows:

\[
\mathcal{L}_{\text{intra-M}} = \mathcal{L}_{\text{GNN}}(\mathcal{D}_{\text{intra}}, G_\theta, \hat{A}) = \mathop{\mbox{E}}_{(x, y) \sim \mathcal{D}_{\text{intra}}} \ell(G_\theta(x, \hat{A}), \hat{y}).
\]

(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 intra-class data set \( \mathcal{D}_{\text{inter}} \) as:

\[
\mathcal{D}_{\text{inter}} = \{(\hat{x}_i, \hat{y}_i) | x_i = M_\lambda(x_i, x_j), y_i = M_\lambda(y_i, y_j), (x_i, y_i) \in \mathcal{D}_i, (x_j, y_j) \in \mathcal{D}_j, y_i \neq \hat{y}_j \}.
\]

(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:

\[
\mathcal{L}_{\text{inter-M}} = \mathcal{L}_{\text{GNN}}(\mathcal{D}_{\text{inter}}, G_\theta, E) = \mathop{\mbox{E}}_{(x, \hat{y}) \sim \mathcal{D}_{\text{inter}}} \ell(G_\theta(x, E), \hat{y}).
\]

(10)

**NLD-aware Sampling.** In a graph, intra/inter-class nodes often form cohesive/cohesive 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 \( Y \) 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 \( q_i = \frac{1}{|N_i|} \sum_{v \in N_i} Y_{vi} \), where \( N_i \) 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 \( q_i' \) as following:

\[
q_i' = \frac{q_i^{\tau}}{\sum_{k=1}^{C} q_k^{\tau}},
\]

(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 \( q_i' \) and \( q_j' \) to determine the likelihood of node i and node j sharing a similar neighbor pattern, where

\[
s_{ij} = \frac{q_i^{T}q_j'}{\|q_i'\|_2\|q_j'\|_2}
\]

(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:

\[
w_{ij} = \begin{cases} 
\frac{1}{1+\beta_{sd}d_j} e^{-\beta_s s_{ij}}, & \text{if } y_i = \hat{y}_j, \\
\frac{1}{1+\beta_{sd}d_j} e^{-\beta_d s_{ij}}, & \text{if } y_i \neq \hat{y}_j,
\end{cases}
\]

(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 \( \frac{1}{1+\beta_{sd}d_j} \) 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.
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.

### Tables and Figures

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

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

#### 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 comparison 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 $\lambda_{int}$ and $\lambda_{lra}$ in \{1, 1.1, 1.2, \ldots, 1.5\}, $\beta_d$ and $\beta_s$ in \{0.5, 1, 1.5, 2\}, and $\gamma$ 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). For OGBN-ARXIV, we perform five runs and report the mean results.
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/).

<table>
<thead>
<tr>
<th>Models</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>71.74±0.2</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>71.49±0.2</td>
</tr>
<tr>
<td>JKNet</td>
<td>72.19±0.2</td>
</tr>
<tr>
<td>GCNII</td>
<td>72.74±0.1</td>
</tr>
<tr>
<td>GCN+NodeMixup</td>
<td>73.46±0.2</td>
</tr>
<tr>
<td>GraphSAGE+NodeMixup</td>
<td>73.24±0.2</td>
</tr>
</tbody>
</table>

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.

Table 3: Ablation Analysis.

<table>
<thead>
<tr>
<th></th>
<th>GCN</th>
<th>GCN + vanilla mixup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-set Pairing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LL</td>
<td>81.85±0.2</td>
<td>81.67±0.3</td>
</tr>
<tr>
<td>LA</td>
<td>81.97±0.6</td>
<td></td>
</tr>
<tr>
<td>LU</td>
<td>83.64±0.5</td>
<td></td>
</tr>
<tr>
<td>Class-specific Mixup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IE</td>
<td>81.97±0.3</td>
<td></td>
</tr>
<tr>
<td>IR</td>
<td>82.06±0.3</td>
<td></td>
</tr>
<tr>
<td>IE+IR</td>
<td>83.64±0.5</td>
<td></td>
</tr>
<tr>
<td>NLD-aware Sampling</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random</td>
<td>82.41±0.2</td>
<td></td>
</tr>
<tr>
<td>NLD-aware</td>
<td>83.64±0.5</td>
<td></td>
</tr>
</tbody>
</table>

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
Chien, E.; Peng, J.; Li, P.; and Milenkovic, O. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. ICLR.
Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. ICLR.
Oono, K.; and Suzuki, T. 2020. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. ICLR.
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


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


