Neurocomputing 655 (2025) 131378

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

G-NodeMixup: Enhancing graph neural networks reachability under extremely limited labels

Ziyu Guan [a], Beilei Ling [a], Weigang Lu [a,1,*], Meng Yan [a], Yaming Yang [a], Wei Zhao [a], Yibing Zhan [b], Dapeng Tao [c]

[a] School of Computer Science and Technology, Xidian University, Xi'an, China
[b] JD Explore Academy, Beijing, China
[c] School of Information Science and Engineering, Yunnan University, Kunming, China

ARTICLE INFO

Communicated by Z. Tao

Keywords:
Graph neural networks
Under-reaching
Extremely limited labels
Semi-supervised learning

ABSTRACT

Graph Neural Networks (GNNs) have shown remarkable performance in semi-supervised node classification, but their effectiveness diminishes in settings with extremely limited labeled data. The scarcity of labeled nodes leads to an under-reaching issue, where unlabeled nodes receive insufficient supervision, resulting in poor generalization. In this paper, we propose G-NodeMixup, a generalized extension of our previously proposed NodeMixup framework which was designed to address under-reaching by improving communication between labeled and unlabeled nodes. G-NodeMixup introduces three novel components: (1) Multi-set Pairing, which facilitates mixup between Labeled-Labeled, Labeled-Unlabeled, and Unlabeled-Unlabeled nodes to enhance node interactions and promote smoother decision boundaries; (2) Subgraph-based Mixup, which focuses mixup within k-hop subgraphs to preserve graph locality and avoid disruptive global edge modifications; and (3) Consistency Regularization-based Mixup Loss, which reduces reliance on noisy pseudo-labels by enforcing consistency between mixed node predictions. Our framework remains architecture-agnostic and can be applied to various GNN models without requiring significant architectural changes or excessive computational overhead. Experimental results across several benchmark datasets demonstrate that G-NodeMixup consistently improves GNN performance in extremely limited labeled settings, achieving state-of-the-art results and establishing its practical effectiveness.

1. Introduction

Graph Neural Networks (GNNs) [1–6], which are designed based on the message-passing protocol [7], have become the mainstream models for dealing with the semi-supervised node classification problem. A recent work [8] reveals that 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 Fig. 1. With the restriction of model depth, those lower-degree nodes can hardly transmit information to far-off labeled nodes. Thus, massive unlabeled nodes are hardly known by labeled nodes in popular 2-layer GNN architectures. This 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 classify. However, in practical scenarios, labeled nodes tend to be distributed sparsely in the graph. This phenomenon, namely *under-reaching* [9,10,13,14], will be discussed in detail.

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

* Corresponding author.
Email addresses: zyguan@xidian.edu.cn (Z. Guan), lingbeilei@stu.xidian.edu.cn (B. Ling), weiganglu314@outlook.cn (W. Lu), mengyan@stu.xidian.edu.cn (M. Yan), yym@xidian.edu.cn (Y. Yang), ywzhao@mail.xidian.edu.cn (W. Zhao), zhanyibing@jd.com (Y. Zhan), dapeng.tao@gmail.com (D. Tao).
[1] Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology, Hong Kong, Hong Kong SAR.

https://doi.org/10.1016/j.neucom.2025.131378
Received 1 April 2025; Received in revised form 26 June 2025; Accepted 23 August 2025
Available online 26 August 2025

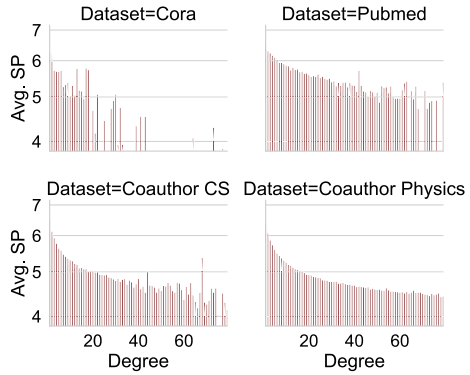


Fig. 1. Visual illustrations of under-reaching. Avg. SP is the mean value of the average shortest path length from each unlabeled node to all the labeled ones, where the mean is taken over unlabeled nodes with the same degree. Lower-degree nodes are farther away from labeled nodes while higher-degree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

to propagate more distant information seems to be a direct solution. Nevertheless, it still raises two additional problems, i.e., over-smoothing [18] which induces indistinguishable node representations and over-squashing [23] which causes distant information loss. To increase reachability through modifying the graph structure, [9,13] 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, developing an effective and flexible method to increase reachability is still a challenging problem.

The key insight to tackle under-reaching is to *improve communication between labeled and unlabeled nodes, facilitating distribution alignment in training*. Recently, mixup [24–32] 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 communication. However, the traditional mixup techniques are proposed to expand labeled data but are 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.

Previous Work: NodeMixup. To alleviate the under-reaching problem, we previously proposed NodeMixup, an architecture-agnostic data augmentation technique designed to increase communication between labeled and unlabeled nodes. NodeMixup addresses under-reaching by introducing Neighbor Label Distribution (NLD)-aware Pair Sampling and Labeled-Unlabeled Mixup strategies. Specifically, it selects node pairs based on similarities in neighborhood label distributions and interpolates between labeled and pseudo-labeled nodes to generate synthetic data that enhance model training. By incorporating intra-class and inter-class mixup strategies, NodeMixup effectively improves distribution alignment and reaches unlabeled nodes, thereby addressing under-reaching without modifying graph structure or increasing Gnn depth.

Limitations of NodeMixup. Despite its success, it has three primary limitations that need to be addressed for more effective performance, particularly in extremely limited labeled settings: (1) Restricted Pairing: NodeMixup only allows labeled-unlabeled pair mixup, limiting the diversity of synthetic samples and thus reducing the generalization potential of the model; (2) Structure Disruption from Edge Mixup: Mixing edges globally on the graph may introduce excessive and noisy connectivity, which could disrupt the graph structure; (3) Potential Noise from Pseudo-Labels: Pseudo-labels are inherently noisy, especially when only a few labeled nodes are available, which may adversely affect training if used directly in the mixup process.

Current Work: G-NodeMixup. To overcome these limitations, we propose G-NodeMixup, a generalized extension of the original framework that significantly enhances reachability and improves generalization in label scarcity scenarios. It introduces three novel enhancements: (1) Multi-set Pairing which allows mixup across labeled-labeled, labeled-unlabeled, and unlabeled-unlabeled pairs, expanding the diversity of interactions and promoting smoother decision boundaries; (2) Subgraph-based Mixup which limits mixup to k-hop subgraphs to maintain locality and mitigate the impact of noisy connections, thereby preserving the overall graph structure; (3) Consistency Regularization-based Mixup Loss which reduces reliance on noisy pseudo-labels by enforcing consistency between mixed node predictions, leading to more robust representations.

By extending mixup beyond the labeled-unlabeled setting and focusing on preserving graph locality through subgraph-based operations, G-NodeMixup is able to tackle under-reaching more effectively, even in extremely limited labeled data conditions. This framework remains architecture-agnostic, making it easy to apply to different Gnns without necessitating significant architectural changes or excessive computational costs.

Contributions. Our main contributions are as follows:

- We introduce G-NodeMixup, an advanced framework that mitigates under-reaching in Gnns by facilitating broader node interactions and addressing the challenges posed by extremely limited labeled data.
- We propose three new components: Multi-set Pairing, Subgraph-based Mixup, and Consistency Regularization-based Mixup Loss to address the limitations of NodeMixup, expanding its applicability to diverse node pairings and improving the quality of supervision.
- We conduct extensive experiments on multiple benchmark datasets to evaluate G-NodeMixup in extremely limited labeled settings, demonstrating its consistent improvement over state-of-the-art methods and establishing its efficacy and generalizability.

2. Preliminary and related works

2.1. Notations

For an arbitrary positive integer M, we denote $[M] := \{1, \dots, M\}$. We use $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y}\}$ to denote an undirected graph with self-loops, where $\mathcal{V} = \{v_1, \dots, v_N\}$ is the node set with N nodes, $\mathcal{E}$ is the edge set with $(v_i, v_j) \in \mathcal{E}$ indicating that there is a connection between v_i and v_j. We also have the feature set $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\}$, where each vector $\boldsymbol{x}_i \in \mathbb{R}^F$ is the input feature of v_i and F is the input dimensionality. We define the node label set as $\mathcal{Y} = \{y_1, \dots, y_{N_\text{L}}\}$, where the label of v_i is $y_i \in [C]$ and C is the number of classes. Here, N_L is the number of labeled nodes. We divide the data set into labeled data set $\mathcal{D}_L = \{(v_1, y_1) \cdots, (v_{N_\text{L}}, y_{N_\text{L}})\}$ and unlabeled data set $\mathcal{D}_U = \{v_{N_\text{L}+1}, \dots, v_{N_\text{L}+N_\text{U}}\}$, where $N_\text{L} + N_\text{U} = N$. Correspondingly, the labeled node set and unlabeled node set are defined as $\mathcal{V}_\text{L}, \mathcal{V}_\text{U} \subseteq \mathcal{V}$, respectively. Specifically, the training set can be divided into C subsets according to different classes, i.e., $\mathcal{V}_\text{L}^{(1)}, \dots, \mathcal{V}_\text{L}^{(C)}$. Besides, we assume that each subset contains T labeled nodes.

2.2. Related works

Graph Neural Networks. Gnns enable each node to accept the information from neighbors within the range of K hops. Gcn [1] introduces the foundational concept of convolution on graphs, using a first-order approximation of spectral graph convolutions. However, to reduce computational complexity, Sgc [3] removes non-linearities and collapses weight matrices between layers, streamlining the operations.

By introducing attention mechanisms, GAT [2] allows for the dynamic weighting of neighbor contributions. Meanwhile, GRAPHSAGE [6] tackles scalability by using a sampling and aggregation strategy, enabling GNN to handle large graphs more effectively. GCNII [4] then introduces residual connections and identity mapping, making it possible to build deeper and more powerful models. For more effective message passing, APPNP [5] applies personalized PageRank to propagate information across a graph, improving node classification performance, particularly in semi-supervised settings. GPRGNN [33] extends this by incorporating a learnable propagation function, offering greater adaptability to various graph structures. Other graph representation learning techniques also include methods based on graph autoencoders that explore multi-view fusion for enhanced representations [34]. GNNS also excel in graph similarity [58] and temporal knowledge graph embedding [59]. A general GNN framework consists of two key operations for each node v_i: (1) AGGREGATE which aggregates messages from v_i's neighborhood $\mathcal{N}_i = \{v_j | (v_i, v_j) \in \mathcal{E}\}$ and (2) UPDATE which updates v_i's representation based on the aggregated representations. For an L-layer GNN f_θ, the formulation of the l-th layer is written as:

$$\begin{aligned} \boldsymbol{m}_i^{(l)} &= \text{AGGREGATE}^{(l)}(\{\boldsymbol{h}_j^{(l-1)} : v_j \in \mathcal{N}_i\}) \\ \boldsymbol{h}_i^{(l)} &= \text{UPDATE}^{(l)}(\boldsymbol{h}_j^{(l-1)}, \boldsymbol{m}_i^{(l)}), \end{aligned} \tag{1}$$

where $1 \le l \le L$, $\boldsymbol{h}^{(0)} = \boldsymbol{x}_i$ is the input node feature, and $\boldsymbol{h}_i^{(l)}$ is the node representation of v_i in the l-th layer. From the individual node's view, we can obtain v_i's output $\boldsymbol{f}_i$ as follows:

$$\boldsymbol{f}_i = f_\theta(v_i, \mathcal{G}). \tag{2}$$

Therefore, the cross-entropy loss [35] ℓ can be adopt in the semi-supervised node classification as:

$$\mathcal{L}_{\text{GNN}}(\mathcal{D}_L, f_\theta, \mathcal{G}) = \mathop{\mathbb{E}}_{(v_i, y_i) \sim \mathcal{D}_L} \ell(f_\theta(v_i, \mathcal{G}), y_i). \tag{3}$$

Mixup. [24] 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 [36–41] 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, [25–27] 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:

$$\mathcal{M}_\lambda(\mathbf{a}, \mathbf{b}) = \lambda \mathbf{a} + (1 - \lambda)\mathbf{b}, \tag{4}$$

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

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 [9,13–17] 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 [18–22], where node representations become indistinguishable, severely impacting prediction performance. Even with the resolution of over-smoothing by enlarging receptive fields, the over-squashing issue [10–12,23,42] 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 [43], 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. Additionally, addressing imbalanced node classification is another critical challenge in this field [44].

3. Understanding under-reaching

3.1. How does under-reaching impact GNNS?

Nodes far from labeled nodes lack supervision information because the influence of labeled nodes decreases with topological distance [45]. 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 inadequately 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 Ref. [9] as:

$$\text{RC}_i = \frac{1}{|\mathcal{D}_L|} \sum_{j \in \mathcal{D}_L} \left(1 - \frac{\log |d_{\mathcal{G}}(i, j)|}{\log D_{\mathcal{G}}}\right), \tag{5}$$

where $D_{\mathcal{G}}$ represents the diameter of graph $\mathcal{G}$, and $d_{\mathcal{G}}(i, j) = D_{\mathcal{G}}$ when v_i and v_j do not belong to the same connected component. A larger RC value (scaled to $[0, 1)$) indicates greater reachability of v_i. In Fig. 2, we visualize the correlation between prediction scores on actual classes and RC values on CORA using different GNNS, i.e., GCN [1], GAT [2], APPNP [5], CHEBNET [46], and GRAPHSAGE [6]. Across all experiments, we vary the number of training nodes per class 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 label decreases, indicating that the reachability declines, the positive correlation becomes more significant. This 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.

3.2. Why does under-reaching fail GNNS?

A recent study [8] 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.

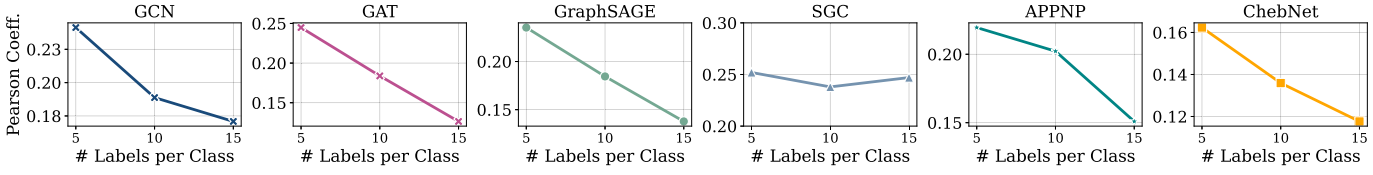


Fig. 2. The correlation between prediction scores on actual classes and RC values on CORA. The Pearson coefficient shows a positive correlation between prediction scores and RC, demonstrating that larger reachability yields better performance. As labeled nodes decrease which (indirectly) suggests poor reachability, a more significant positive correlation can be observed.

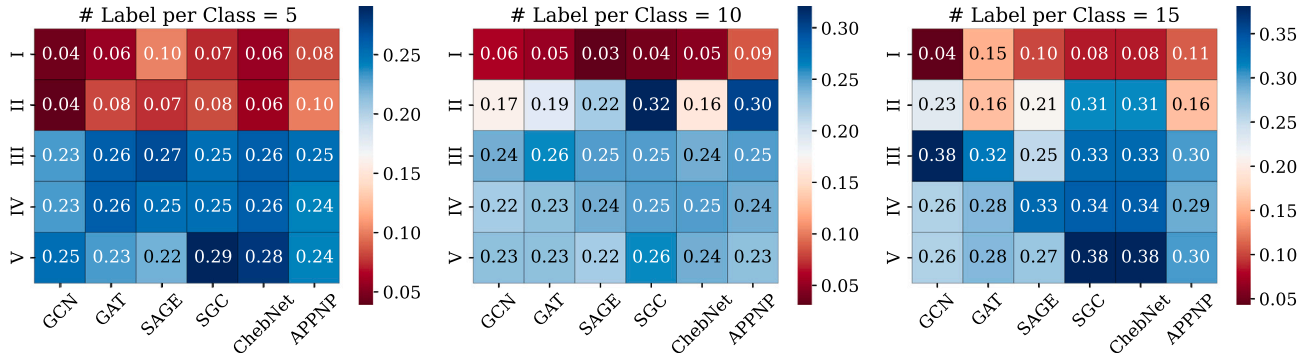


Fig. 3. The heat map of CKA between labeled and unlabeled distributions learned by different GNNs using CORA. A larger CKA value indicates more similar representation distributions between labeled and unlabeled nodes. I, …, V represent unlabeled nodes with various reachability arranged in ascending order.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

Intuitively, it facilitates inference as the two distributions get closer. Inspired by this, to further understand the negative impact of under-reaching, we investigate the distance between the labeled and unlabeled distributions at different levels of RC. We first briefly introduce the centered kernel alignment (CKA) [47] metric,[2] which is widely used to measure the representation similarity. Supposing $Z_{\mathrm{L}}, Z_{\mathrm{U}} \in \mathbb{R}^{m\times n}$ are 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:

$$\mathrm{CKA}(Z_{\mathrm{L}}, Z_{\mathrm{U}}) = \frac{\|Z_{\mathrm{U}}^T Z_{\mathrm{L}}\|_F}{\|Z_{\mathrm{L}} Z_{\mathrm{L}}^T\|_F \|Z_{\mathrm{U}} Z_{\mathrm{U}}^T\|_F}. \tag{6}$$

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

3.3. 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 computational costs for finding globally friendly neighbors [9,13] or lead to noisy graphs without sufficient supervision [9]. 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 [24–27] 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] Please refer to [8,47] for more details about CKA.

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

4. NODEMIXUP

NODEMIXUP was originally proposed to address the under-reaching problem in GNNs, where the sparse distribution of labeled nodes limits the ability of the model to propagate information effectively to unlabeled nodes. As a result, the model struggles to align the representations of labeled and unlabeled nodes, which hampers its performance in semi-supervised node classification tasks. To mitigate this issue, NODEMIXUP introduces a novel data augmentation technique based on two key strategies: **(1) Neighbor Label Distribution (NLD)-aware Pair Sampling (NLDS)** which ensures that nodes with similar neighbor patterns and lower degrees are more likely to be selected; **(2) Labeled-Unlabeled Mixup (LUMixup)** which interpolates between labeled and unlabeled nodes.

We present the pipeline of NODEMIXUP in Fig. 4. The figure illustrates the steps involved in the NODEMIXUP process: (1) Starting from the original graph, pseudo-labels are generated for unlabeled nodes to create the pseudo-labeled graph. (2) NLD-aware weights are used to select intra-class and inter-class pairs. (3) Mixup is performed in the graph, with different strategies for intra-class (including edge mixup) and inter-class (feature-only mixup) node pairs.

4.1. NLD-aware pair sampling

In graphs, nodes within the same class often form cohesive communities or clusters, leading to similar neighbor label distributions. Conversely, nodes from different classes tend to have dissimilar neighbor

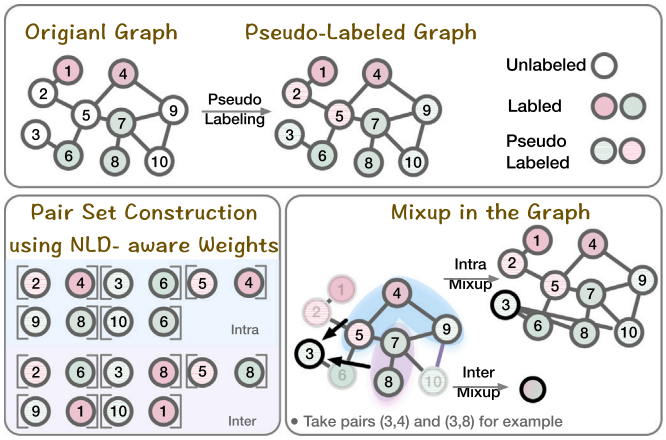


Fig. 4. An overview of the NODEMIXUP methodology. Nodes with the same color belong to the same class. Examples of intra-class and inter-class pairings are highlighted to show how NODEMIXUP works.

distributions. The characteristics of a node are influenced not only by its own features but also by the features and labels of its neighbors. To capture such neighborhood information and effectively sample node pairs for mixup, we adapt the Neighborhood Label Distribution (NLD) [48] and introduce an NLD-aware pair sampling strategy (NLDS). For a given node v_i, NLDS samples a target node v_j to form a node pair (v_i, v_j) from the target node set $\mathcal{V}_{\texttt{target}}$ as follows:

$$(v_i, v_j) \sim \texttt{NLDS}(v_i, \mathcal{V}_{\texttt{target}}). \tag{7}$$

In our NodeMixup, we use the pseudo-labeled node set $\mathcal{V}_{\text{PL}}$ as the target node set.

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

Sharpening of NLD. To enhance the contrast between class probabilities in the NLD and focus on the most probable classes, we apply the sharpening technique [49]. The sharpened NLD $\boldsymbol{q}'_{ij}$ for v_i is computed as:

$$\mathbf{q}'_{ij} = \frac{\mathbf{q}_i^{1/\tau}}{\sum_{k=1}^{C}\mathbf{q}_{ik}^{1/\tau}}, \tag{8}$$

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.

Sampling Weight Calculation. First, we use the cosine similarity s_{ij} between $\boldsymbol{q}'_i$ and $\boldsymbol{q}'_j$ to determine the likelihood of v_i and v_j sharing a similar neighbor pattern, where

$$s_{ij} = \frac{\mathbf{q}'^{T}_i\mathbf{q}'_j}{\|\mathbf{q}'_i\|_2\|\mathbf{q}'_j\|_2} \tag{9}$$

To account for the under-reaching problem affecting low-degree nodes, we incorporate the node degrees into the sampling weights. For a node pair (v_i, v_j), we define the sampling weight w_{ij} as follows:

$$w_{ij} = \begin{cases} \frac{1}{1+\beta_d d_j}e^{\beta_s s_{ij}}, & \text{if} \quad y_i = \hat{y}_j, \\ \frac{1}{1+\beta_d d_j}e^{-\beta_s s_{ij}}, & \text{if} \quad y_i \neq \hat{y}_j, \end{cases} \tag{10}$$

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.

Mixup Pair Set Construction. Using the calculated sampling weights w_{ij}, we construct the mixup pair set $\mathcal{P}$ by selecting pairs of nodes with probabilities proportional to w_{ij}. Firstly, we determine the number of pairs as follows:

$$K = \min(N_{\text{L}}, N_{\text{PL}}). \tag{11}$$

Here, N_{L} represents the number of labeled nodes, and N_{PL} denotes the number of nodes in the pseudo-labeled node set. Then, we form the mixup pair set $\mathcal{P}$ as follows:

$$\mathcal{P} = \{(v_i, v_j) | (v_i, v_j) \sim \texttt{NLDS}(v_i, \mathcal{V}_{\text{PL}}), v_i \in \sigma(\mathcal{V}_{\text{L}}, K)\}, \tag{12}$$

where $\sigma(\cdot, K)$ is the function that randomly samples K nodes from the given node set. Each pair (v_i, v_j) is selected based on the sampling weights, ensuring that nodes with higher weights are more likely to be included.

4.2. Labeled-unlabeled mixup

After selecting node pairs using the NLD-aware pair sampling strategy, we perform LUMixup to generate synthetic training data that enhances the model's ability to generalize from labeled to unlabeled nodes. This mixup process integrates both intra-class and inter-class Mixup into a unified framework, allowing for an efficient data augmentation strategy.

Mixed Feature and Label Generation. For each pair $(v_i, v_j) \in \mathcal{P}$ with (pseudo) labels as y_i and $\hat{y}_j$, we generate the mixed feature and label vectors ($\tilde{\boldsymbol{x}}_{ij}$ and $\tilde{y}_{ij}$) as follows:

$$\begin{aligned} &\text{Feature Mixup:} \quad \tilde{\boldsymbol{x}}_{ij} = \mathcal{M}_\lambda(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &\text{Label Mixup:} \quad \tilde{y}_{ij} = \mathcal{M}_\lambda(y_i, \hat{y}_j) \end{aligned}, \tag{13}$$

where $\hat{y}_j$ is the pseudo label of v_j.

Mixed Structure Generation. To incorporate structural information into the mixup process, we define the mixed edge set $\tilde{\mathcal{E}}_{ij}$ for each pair (v_i, v_j) as follows.

For the intra-class pair ($y_i = \hat{y}_j$), where the labeled node and the unlabeled node share the same (pseudo) label, we fuse their structural information to promote intra-class similarity. The mixed edge set $\tilde{\mathcal{E}}^{+}_{ij}$ is defined as:

$$\tilde{\mathcal{E}}^{+}_{ij} = \{(v_i, v_m) | v_m \in \mathcal{N}_i\} \cup \{(v_i, v_p) | v_p \in \mathcal{N}_j\}. \tag{14}$$

This effectively merges the neighborhoods of v_i and v_j, connecting the mixed node v_i to the neighbors of both original nodes.

For the inter-class pair ($y_i \neq \hat{y}_j$), where the labeled node and the unlabeled node belong to different classes, we avoid fusing their structural information to prevent introducing noisy connections. Instead, we keep the mixed node as an isolated node to focus on learning a smooth transition between different classes. The mixed edge set $\tilde{\mathcal{E}}^{-}_{ij}$ is defined as:

$$\tilde{\mathcal{E}}^{-}_{ij} = \{(v_i, v_i)\}, \tag{15}$$

where (v_i, v_i) represents a self-loop for the mixed node v_i. The reason behind inter-class mixup is that interpolating between inputs with different labels results in decision boundaries that transition linearly from one class to another, reducing prediction errors [24]. 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. Combining both cases, we generalize the mixed edge set $\tilde{\mathcal{E}}_{ij}$ as:

$$\tilde{\mathcal{E}}_{ij} = \begin{cases} \tilde{\mathcal{E}}^{+}_{ij}, & \text{if} \quad y_i = \hat{y}_j \\ \tilde{\mathcal{E}}^{-}_{ij}, & \text{if} \quad y_i \neq \hat{y}_j \end{cases}. \tag{16}$$

We then add the new edges to a copy of the original graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}, \mathcal{Y}\}$ and replace the original features to form a mixed graph $\tilde{\mathcal{G}} = \{\tilde{\mathcal{V}}, \tilde{\mathcal{E}}, \tilde{\mathcal{X}}, \tilde{\mathcal{Y}}\}$:

$$\begin{aligned} \tilde{\mathcal{V}} &= \mathcal{V} \\ \tilde{\mathcal{E}} &= \mathcal{E} \underset{(v_i,v_j)\in\mathcal{P}}{\cup} \tilde{\mathcal{E}}_{ij} \\ \tilde{\mathcal{X}} &= \left(\mathcal{X}\backslash\{\boldsymbol{x}_i | (v_i, v_j) \in \mathcal{P}\}\right) \cup \{\tilde{\boldsymbol{x}}_{ij} | (v_i, v_j) \in \mathcal{P}\} \\ \tilde{\mathcal{Y}} &= \left(\mathcal{Y}\backslash\{y_i | (v_i, v_j) \in \mathcal{P}\}\right) \cup \{\tilde{y}_{ij} | (v_i, v_j) \in \mathcal{P}\} \end{aligned} \tag{17}$$

4.3. Loss function

Supervised Loss on the Original Graph. For the labeled node set D_L, we use the supervised loss with the cross-entropy loss function (defined

in Eq. 3) as follows:

$$\mathcal{L}_{\texttt{sup}} = \mathbb{E}_{(v_i,y_i)\sim\mathcal{D}_L}\ell(f_\theta(v_i,\mathcal{G}),y_i). \tag{18}$$

This standard supervised loss ensures that the model learns to predict the correct labels for the labeled nodes based on their features.

Mixup Loss on the Mixed Graph. The mixup loss is designed to train the model on the synthetic data generated through the LUMixup process. We use the standard cross-entropy loss between the model's prediction on the mixed feature and the mixed label:

$$\mathcal{L}_{\texttt{mix}} = \sum_{(v_i,v_j)\in\mathcal{P}} \left(\alpha^+\mathbb{I}_{[y_i=\hat{y}_j]} + \alpha^-\mathbb{I}_{[y_i\neq\hat{y}_j]}\right)\ell\left(f_\theta(v_i,\tilde{\mathcal{G}}),\tilde{y}_{ij}\right), \tag{19}$$

where $\mathbb{I}_{[\cdot]}$ is the indicator function, which equals 1 if the condition inside is true and 0 otherwise. $\alpha^+,\alpha^-\in(0,1]$ are hyperparameters controlling the contributions of intra-class and inter-class mixup losses, respectively.

Total Loss. Finally, the model is optimized by minimizing the loss $\mathcal{L}_{\text{NODEMIXUP}}$ from the combination of $\mathcal{L}_{\texttt{sup}}$ and $\mathcal{L}_{\texttt{mix}}$ as follows:

$$\mathcal{L}_{\text{NODEMIXUP}} = \mathcal{L}_{\texttt{sup}} + \mathcal{L}_{\texttt{mix}}. \tag{20}$$

The original NodeMixup framework offers a flexible, architecture-agnostic strategy to improve the reachability of GNNs by augmenting labeled data through the interpolation of labeled and pseudo-labeled nodes. While NodeMixup effectively enhances the communication between labeled and unlabeled nodes, *three key limitations have been identified*: **(1) Restricted Pairing:** NodeMixup is limited to labeled-unlabeled node pairs. This restriction means that the model does not fully leverage the potential diversity of mixup. By not including labeled-labeled and unlabeled-unlabeled pairs, NodeMixup does not capitalize on all available pairings, which could otherwise enrich the model's capacity to generalize across different node categories. **(2) Structure Disruption from Edge Mixup:** Performing edge mixup directly on the original graph leads to global edge modifications, which can cause an undesirable increase in connectivity. This excessive edge addition may compromise the graph's original structure, leading to potentially noisy connections. **(3) Potential Noise from Pseudo-Labels:** NodeMixup relies on pseudo-labels for unlabeled nodes during mixup. However, pseudo-labels can be noisy, especially in scenarios where label confidence is low. This introduces potential errors that could negatively impact the generalization of the model.

To address these limitations, we propose G-NodeMixup, a more generalized framework that augments NodeMixup with *three novel enhancements*: **(1) Multi-set Pairing** which allows mixup across labeled-labeled, labeled-unlabeled, and unlabeled-unlabeled pairs. **(2) Subgraph-based Mixup** approach to preserve the global graph structure. **(3) Consistency Regularization-based Mixup Loss** which avoids reliance on potentially noisy pseudo-labels.

5. G-NodeMixup

We provide an overview of G-NodeMixup in Fig. 5. The figure illustrates the construction of multi-set pairs and the subgraph-based mixup operations. Subgraphs are extracted for the selected pairs to maintain locality, and intra-class and inter-class mixup are performed. Examples of mixup between pairs (1, 4) and (1, 7) are shown, highlighting the multi-set and subgraph-based strategies of G-NodeMixup, which ensure improved reachability and more effective feature learning while preserving graph structure. We detail these modifications and their implementation in G-NodeMixup below.

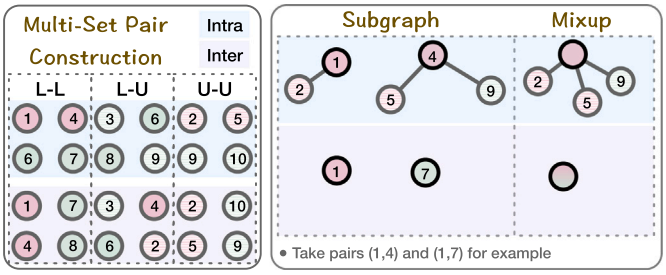


Fig. 5. An overview of G-NodeMixup.

5.1. Multi-set pairing

The Multi-set Pairing module (MSP) in G-NodeMixup expands the pairing capabilities of the original framework by supporting labeled-labeled (L-L), labeled-unlabeled (L-U), and unlabeled-unlabeled (U-U) pairing modes, to leverage the entire set of node interactions.

Performing mixup directly between labeled nodes that have correct and reliable labels allows us to enhance the representations within the labeled space itself without introducing the risk of noise, which is present in pseudo-labels. Since labeled nodes carry true labels, mixing them reinforces class-specific patterns and aids the model in learning smoother decision boundaries based on trusted data. This reduces the model's reliance on pseudo-labels, which may be uncertain, thus enhancing the quality of feature representations derived from the labeled data. A large portion of nodes may remain unlabeled in graphs. Performing mixup between unlabeled nodes allows the model to better explore the unlabeled space, enforcing smooth transitions and consistency in feature representation across similar unlabeled regions. While these nodes lack the certainty of labeled data, their structural or feature-based similarities can still provide useful cues for the model.

Similar to the mixup pair set construction introduced in LUMixup, we construct these three mixup pair sets, i.e., $\mathcal{P}_{\texttt{LL}}$, $\mathcal{P}_{\texttt{LU}}$, and $\mathcal{P}_{\texttt{UU}}$, as follows:

$$\begin{aligned}
\mathcal{P}_{\texttt{LL}} &= \{(v_i,v_j)|(v_i,v_j)\sim \texttt{NLDS}(v_i,\mathcal{V}_{\texttt{L}}), v_i\in\sigma(\mathcal{V}_{\texttt{L}},K_{\texttt{LL}})\}\\
\mathcal{P}_{\texttt{LU}} &= \{(v_i,v_j)|(v_i,v_j)\sim \texttt{NLDS}(v_i,\mathcal{V}_{\texttt{PL}}), v_i\in\sigma(\mathcal{V}_{\texttt{L}},K_{\texttt{LU}})\}\\
\mathcal{P}_{\texttt{UU}} &= \{(v_i,v_j)|(v_i,v_j)\sim \texttt{NLDS}(v_i,\mathcal{V}_{\texttt{PL}}), v_i\in\sigma(\mathcal{V}_{\texttt{PL}},K_{\texttt{UU}})\},
\end{aligned} \tag{21}$$

where $K_{\texttt{LL}}=N_{\texttt{L}}$, $K_{\texttt{LU}}=\min(N_{\texttt{L}},N_{\texttt{PL}})$, and $K_{\texttt{UU}}=N_{\texttt{PL}}$.

5.2. Subgraph-based mixup

To mitigate the adverse effects of within-graph edge mixup (used in intra-class mixup), we apply edge mixup on k-hop subgraphs rather than on the entire graph. Our Subgraph-based Mixup (SGMixup) allows us to control the extent of connectivity changes by performing mixup within localized regions surrounding each node.

Mixed Subgraph. For each pair of nodes mixup (v_i,v_j) selected for edge mixup, we extract their respective k-hop subgraphs $\mathcal{G}_i^k$ and $\mathcal{G}_j^k$:

$$\mathcal{G}_i^k = \{\mathcal{V}_i^k,\mathcal{E}_i^k,\mathcal{X}_i^k,\mathcal{Y}_i\},\quad \mathcal{G}_j^k = \{\mathcal{V}_j^k,\mathcal{E}_j^k,\mathcal{X}_j^k,\mathcal{Y}_j\}, \tag{22}$$

where $\mathcal{V}_i^k$ contains all the nodes within k-hop neighbors of v_i and $\mathcal{E}_i^k$ and $\mathcal{X}_i^k$ are the corresponding edge set and feature matrix, respectively. If v_i is a labeled node, $\mathcal{Y}_i=\{y_i\}$; otherwise, if v_i is an unlabeled node, $\mathcal{Y}_i=\{\hat{y}_i\}$. We then merge these subgraphs to create a combined subgraph $\tilde{\mathcal{G}}_{ij}^k$:

$$\tilde{\mathcal{G}}_{ij}^k = \{\tilde{\mathcal{V}}_2^k,\tilde{\mathcal{E}}_{ij}^k,\tilde{\mathcal{X}}_{ij}^k,\tilde{\mathcal{Y}}_{ij}\}. \tag{23}$$

Mixed Node Set. In $\tilde{\mathcal{V}}_2^k$, we combine all the nodes from $\mathcal{V}_i^k$ and $\mathcal{V}_j^k$ but remove the central node v_j from $\mathcal{V}_j^k$. Therefore, $\tilde{\mathcal{V}}_2^k$ should be formally written as:

$$\tilde{\mathcal{V}}_2^k = \mathcal{V}_i^k \cup \left(\mathcal{V}_j^k\backslash\{v_j\}\right). \tag{24}$$

Mixed Edge Set. In $\tilde{\mathcal{E}}_{ij}^k$, v_i has all the 1-hop connections of v_j by "taking the place of" v_j:

$$\tilde{\mathcal{E}}_{ij}^k = \mathcal{E}_i^k \cup \mathcal{E}_j^k \cup \{(v_i,v_k)|(v_j,v_k)\in\mathcal{E}_j^k\}\backslash\{(v_j,v_k)|(v_j,v_k)\in\mathcal{E}_j^k\}. \tag{25}$$

Mixed Feature Set. For $\tilde{\mathcal{X}}_{ij}^k$, it contains all the features of nodes from $\tilde{\mathcal{V}}_2^k$ but with the central node's feature $\boldsymbol{x}_i$ replaced with a mixed feature $\tilde{\boldsymbol{x}}_{ij}$ between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$:

$$\tilde{\mathcal{X}}_{ij}^{k} = \{\boldsymbol{x}_k | v_k \in \left(\tilde{\mathcal{V}}_2^k \backslash \{v_i\}\right)\} \cup \{\tilde{\boldsymbol{x}}_{ij} | \tilde{\boldsymbol{x}}_{ij} = \mathcal{M}_\lambda(\boldsymbol{x}_i, \boldsymbol{x}_j)\}. \tag{26}$$

For the inter-class mixup, we still maintain the same operation as discussed in Section 4.2. To generalize intra- and inter-class mixup, we reformulate Eq. (23) as follows:

$$\tilde{\mathcal{G}}_{ij}^{k} = \begin{cases} \{\tilde{\mathcal{V}}_2^k, \tilde{\mathcal{E}}_{ij}^k, \tilde{\mathcal{X}}_{ij}^k\}, & \hat{y}_i = \hat{y}_j \quad \text{(Intra-class)} \\ \{\{v_i\}, \{(v_i, v_i)\}, \{\tilde{\boldsymbol{x}}_{ij}\}\}, & \hat{y}_i \neq \hat{y}_j \quad \text{(Inter-class)} \end{cases}, \tag{27}$$

This equation defines two distinct strategies guided by the predicted labels. **Intra-class** mixup is applied when nodes v_i and v_j share the same predicted label ($\hat{y}_i = \hat{y}_j$), with the goal of enhancing in-class compactness. Conversely, **Inter-class** mixup operates on pairs with different predicted labels ($\hat{y}_i \neq \hat{y}_j$), aiming to improve class discriminability.

Specifically, for the Inter-class case, the notation $\{\{v_i\}, \{(v_i, v_i)\}, \{\tilde{\boldsymbol{x}}_{ij}\}\}$ indicates that the resulting subgraph for the mixed node consists of only a single node with a self-loop.

5.3. Consistency regularization-based mixup loss

The primary motivation behind Consistency Regularization-based Mixup Loss (CRML) stems from the fact that pseudo-labels (used in NodeMixup) are inherently noisy, especially in extremely limited label settings where the model must predict labels for a significant number of unlabeled nodes. Relying directly on these pseudo-labels during mixup can introduce noise, leading to incorrect supervision signals and negatively impacting model performance.

Mixup Loss for Unlabeled Nodes. To overcome this issue, instead of interpolating the labels of mixed nodes (which can be unreliable for pseudo-labeled nodes), we interpolate the model's predictions for the original node pairs and then enforce consistency between this interpolated prediction and the prediction for the mixed node. This strategy ensures that the model learns to make smooth transitions between the predictions of the original nodes, regardless of their labels, thus promoting more robust feature representations.

For a pair of nodes (v_i, v_j) where one or both nodes are unlabeled, we first compute the outputs (before Softmax) using the original graph $\mathcal{G}$ as:

$$\boldsymbol{f}_i = f_\theta(v_i, \mathcal{G}), \quad \boldsymbol{f}_j = f_\theta(v_j, \mathcal{G}). \tag{28}$$

Then, we compute an interpolated output $\tilde{\boldsymbol{f}}_{ij}$ as a weighted combination of these original outputs:

$$\tilde{\boldsymbol{f}}_{ij} = \mathcal{M}_\lambda(\boldsymbol{g}_i, \boldsymbol{g}_j). \tag{29}$$

CRML is defined by enforcing consistency between the model's prediction for the mixed feature vector $f_\theta(\tilde{\boldsymbol{x}}_{ij}, \tilde{\mathcal{E}}_{ij}^k)$ and the interpolated prediction $\tilde{\boldsymbol{f}}_{ij}$:

$$\mathcal{L}_{\text{CRML}} = \mathbb{E}_{(v_i, v_j) \in \mathcal{P}_{\text{LU}} \cup \mathcal{P}_{\text{UU}}} \| f_\theta(v_i, \tilde{\mathcal{G}}_{ij}^k) - \tilde{\boldsymbol{f}}_{ij} \|^2. \tag{30}$$

This consistency loss ensures that the model learns smooth transitions between node representations without directly relying on the pseudo-labels, which might be noisy. It promotes smooth and consistent decision boundaries in the feature space.

Mixup Loss for Labeled Nodes. For L-L pairs, we can safely apply the standard mixup loss because the labels are correct and reliable. In this case, mixup is applied not only to the node features but also to the labels of the nodes. For a pair of labeled nodes (v_i, y_i) and (v_j, y_j), the mixup loss is defined as:

$$\mathcal{L}_{\text{LLmix}} = \sum_{(v_i, v_j) \in \mathcal{P}_{\text{LL}}} \left(\alpha^+ \mathbb{I}_{[y_i = y_j]} + \alpha^- \mathbb{I}_{[y_i \neq y_j]} \right) \ell \left(f_\theta(v_i, \tilde{\mathcal{G}}_{ij}^k), \tilde{y}_{ij} \right). \tag{31}$$

Finally, the overall loss for G-NodeMixup should be written as follows:

$$\mathcal{L}_{\text{G-NODEMIXUP}} = \mathcal{L}_{\text{sup}} + \alpha_{\text{CRML}} \mathcal{L}_{\text{CRML}} + \alpha_{\text{LLmix}} \mathcal{L}_{\text{LLmix}}, \tag{32}$$

where $\alpha_{\text{CRML}}, \alpha_{\text{LLmix}} \in \mathbb{R}$ are the balance factors.

5.4. Computational complexity analysis

This section analyzes the computational complexity of the G-NodeMixup framework, comparing it with baseline graph augmentation techniques. Our analysis focuses on the additional overhead introduced by the augmentation methods, excluding the fundamental computational cost of the underlying Gnn backbone.

The G-NodeMixup framework comprises two components: **NLD-aware Pair Sampling** is crucial for selecting node pairs in G-NodeMixup by leveraging Neighbor Label Distribution similarity, involving an initial NLD calculation with complexity $O(|\mathcal{E}|)$, where $\mathcal{E}$ is the edge set, followed by NLD cosine similarity computation with complexity $O(|\mathcal{V}_{PL}|^2)$, where $\mathcal{V}_{PL}$ is the pseudo-labeled node set, and sampling weight computation with complexity $O(|\mathcal{P}|)$, where $\mathcal{P} = \mathcal{P}_{LL} \cup \mathcal{P}_{LU} \cup \mathcal{P}_{UU}$ is the total pair set, $|\mathcal{P}| = |\mathcal{V}_L| + \min(|\mathcal{V}_L|, |\mathcal{V}_{PL}|) + |\mathcal{V}_{PL}|$, and $\mathcal{V}_L$ is the labeled node set. The total complexity is $O(|\mathcal{E}| + |\mathcal{V}_{PL}|^2 + |\mathcal{P}|)$. **Subgraph extraction and mixup** for a node pair $(v_i, v_j) \in \mathcal{P}$ involves traversing the k-hop neighbors of each node using breadth-first search (BFS) to construct the corresponding subgraph, where k represents the number of hops (typically small, e.g., $k = 2$) defining the subgraph's locality. In sparse graphs, this process has a complexity of $O(d_{avg}^{k+1})$ per node pair, where d_{avg} is the average degree of the graph, leading to a total complexity of $O(|\mathcal{P}| \cdot d_{avg}^{k+1})$ for all $|\mathcal{P}|$ node pairs. The feature mixup step, which performs linear interpolation of subgraph feature matrices, incurs a complexity of $O(|\mathcal{P}| \cdot d_{avg}^{k})$, where d_{avg}^{k} approximates the average number of nodes in a k-hop subgraph, while the label mixup step has a complexity of $O(|\mathcal{P}|)$. Given the optimization in sparse graphs, the overall complexity is dominated by the subgraph extraction, resulting in $O(|\mathcal{P}| \cdot d_{avg}^{k+1})$.

The computational complexities of the baseline methods are summarized as follows: NodeMixup [55] has a complexity of $\mathcal{O}(|\mathcal{E}| + |\mathcal{D}_{\text{pl}}|^2 + |\mathcal{D}_{\text{pl}}|)$, while IGraphMix [56] has a complexity of $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}|)$. Here, $|\mathcal{V}|$ denotes the number of nodes, $|\mathcal{E}|$ denotes the number of edges, and $|\mathcal{D}_{\text{pl}}|$ denotes the size of the confident pseudo-labeled node set.

In summary, our G-NodeMixup exhibits a complexity of $O(|\mathcal{E}| + |\mathcal{V}_{PL}|^2 + |\mathcal{P}|)$. NodeMixup has a complexity of $\mathcal{O}(|\mathcal{E}| + |\mathcal{D}_{\text{pl}}|^2 + |\mathcal{D}_{\text{pl}}|)$, and IGraphMix has a complexity of $\mathcal{O}(|\mathcal{E}| + |\mathcal{V}|)$. Notably, G-NodeMixup achieves a significant improvement in data augmentation performance with a comparable complexity to NodeMixup.

6. Experiments

In this section, we evaluate the performance of G-NodeMixup on several benchmark datasets for the node classification tasks with extremely limited labels available. We compare our proposed method with state-of-the-art baselines and conduct ablation studies to demonstrate the effectiveness of each component in G-NodeMixup. The experiments are designed to answer the following research questions:

- **RQ1:** Can G-NodeMixup be applied to various Gnn models?
- **RQ2:** How does G-NodeMixup perform compared to existing augmentation methods with extremely limited labels available?
- **RQ3:** What is the impact of each proposed component on the overall performance?
- **RQ4:** How sensitive is G-NodeMixup to hyperparameters such as the balancing factors in the loss function?

6.1. Experimental setup

Hardware and Software. G-NodeMixup[4] is implemented based on the Torch Geometric library [50] and PyTorch 2.0.1 with Intel(R) Core(TM) i9-10980XE CPU @ 3.00 GHz and 2 NVIDIA TITAN RTX GPUs with 24GB of memory.

Dataset. We evaluate G-NodeMixup on the node classification task using five commonly used datasets and provide the details in Table 1.

[4] https://anonymous.4open.science/r/G-NodeMixup-CD70/

Table 1
Dataset statistics.

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
Coauthor CS	18,333	81,894	6,805	15
Coauthor Physics	34,493	247,962	8,415	5

Cora, Citeseer, and Pubmed [51][5] are citation networks in which each publication is defined by a 0/1-valued word vector indicating the presence or absence of the corresponding dictionary word. **Coauthor CS and Coauthor Physics** [52][6] are co-authorship graphs extracted from the Microsoft Academic Graph. Authors are represented as nodes, which are connected by an edge if two authors are co-authors of the same paper.

Baseline GNNs. We choose *six* baseline GNNs as follows:

- **GCN** [1] introduces a convolutional neural network architecture for graphs by aggregating feature information from local neighborhoods.
- **GAT** [2] incorporates attention mechanisms into graph neural networks, allowing nodes to weigh the importance of their neighbors' features during the aggregation process.
- **SuperGAT** [53] improves upon GAT by introducing self-supervised attention mechanisms to better identify informative neighbor nodes for message passing.
- **GraphSAGE** [6] is an inductive framework that generates node embeddings by sampling and aggregating features from a node's local neighborhood, enabling scalable learning on large graphs.
- **UniMP** [54] adopts a Graph Transformer to combine feature and label propagation.
- **APPNP** [5] combines GCN and PageRank to derive an improved propagation scheme.

Comparing Augmentation Techniques. We choose two SOTA mixup-based augmentation methods and a representative structural augmentation technique as follows:

- **NodeMixup** [55] performs labeled-unlabeled mixup operations to address under-reaching.
- **IGraphMix** [56] proposes an input-level graph mixup method to augment graph data.
- **DropEdge** [57] randomly deletes parts of edges to address over-fitting and over-smoothing for GNNs.

Training Configurations. In our experiments, we utilize six distinct GNN backbone models, as previously introduced, which serve as the foundation for all data augmentation methods. All models are trained using the **Adam optimizer [60]**. For baseline methods, we follow the specific hyperparameter settings reported in their original papers where applicable, to ensure a fair comparison of the augmentation techniques themselves. Each trial consists of 500 epochs, we conduct 10 independent runs, and early stopping patience is set to 100. For all these GNN backbone models, a hidden channel size of 256 is used. In G-NodeMixup, the Intra/Inter-class mixup weights (α^+ and α^-) are fixed at 0.5, and the pseudo-labeling confidence threshold (γ) is set to 0.9. Other key hyperparameters are selected via a grid search that optimizes for the best accuracy on the validation set of each benchmark. The search spaces for these are detailed below:

- **Baseline GNNs**:
  - Number of Layers: $\{2, 3, 4\}$
  - Dropout Rate: $\{0, 0.05, 0.1, \ldots, 0.8\}$
  - Learning Rate: $\{0.001, 0.005, 0.01\}$
  - Weight Decay Rate: $\{5\times10^{-10}, 5\times10^{-8}, 5\times10^{-6}, 5\times10^{-4}, 5\times10^{-2}\}$
- **G-NodeMixup:**
  - Hop Parameter: $k = \{1, 2, 3, 4, 5\}$
  - Loss Balancing Coefficients: $\alpha_{\text{CRML}}, \alpha_{\text{LLmix}} = \{0.5, 1, 1.5, 2\}$
  - NLD Sampling Weights: $\beta_s, \beta_d = \{0.5, 1, 1.5, 2\}$
  - Mixup Beta Parameter: $\alpha = \{1, 2, 3\}$

For a complete list of all configurations and implementation details, please refer to our source code.

6.2. Generalizability study (RQ1)

In this section, we conduct experiments to demonstrate the generalizability of G-NodeMixup across multiple GNN backbones in extremely limited label settings. The results are presented in Table 2. For all the adopted datasets, we randomly choose one and two labeled nodes per class (# T = 1 and 2) for training, respectively, in order to evaluate the model's performance under label-scarce settings. For detailed parameter settings, please refer to Section 6.1. The better performance and larger improvement between NodeMixup and G-NodeMixup w.r.t each backbone and # T are marked in **bold**.

For each backbone, G-NodeMixup outperforms NodeMixup by significant margins, particularly when only one or two labels are available. For instance, with GCN as the backbone, G-NodeMixup achieves improvements of up to 17.44 % over GCN and up to 9.66 % over NodeMixup. Additionally, with GraphSAGE, G-NodeMixup achieves improvements of up to 24.47 % over the baseline model, showing its effectiveness in reinforcing learning even in complex, low-label environments. Similarly, improvements with G-NodeMixup are observed across all backbones, highlighting its versatility in boosting the classification accuracy regardless of the underlying model.

These improvements can be attributed to G-NodeMixup's ability to (1) leverage multi-set pairing, which effectively enhances communication between labeled and unlabeled nodes, even in sparsely labeled scenarios; (2) preserve the local structural information of the graph while enhancing communication between nodes; (3) enforce smooth transitions between node predictions through consistency regularization to reduce the negative impact of noisy labels.

6.3. Effectiveness study (RQ2)

In this section, we compare G-NodeMixup with three graph augmentation techniques, i.e., NodeMixup, IGraphMix, and DropEdge under different label settings (from 1 to 5 labels per class) on five benchmark datasets using GCN as the backbone model. The average performance in terms of classification accuracy is presented in Fig. 6.

G-NodeMixup consistently outperforms all other methods across various label settings and datasets. This superior performance can be attributed to the enhancements introduced by G-NodeMixup, specifically the generalization of mixup strategies to incorporate MSP. By expanding the mixup capabilities beyond L-U pairs,G-NodeMixup enables richer and more robust feature learning, especially in extremely limited label settings. This approach not only helps the model generalize better to unlabeled nodes but also ensures that labeled data itself is effectively used to reinforce the learning process. This is evident in the accuracy improvements achieved by G-NodeMixup and other baselines, particularly at lower label counts, where the impact of improved feature representation and broader mixup strategies is most pronounced.

NodeMixup, while effective in improving performance compared to standard GCN, relies heavily on pseudo-labels, which introduce noise into the learning process, especially when the confidence in pseudo-labels is low. This dependency on pseudo-labels is a key limitation of NodeMixup, which can lead to suboptimal feature representations and hinder its effectiveness in challenging settings with very limited labeled data. In contrast, G-NodeMixup mitigates this issue by introducing consistency regularization, which encourages smooth transitions between

[5] https://linqs.soe.ucsc.edu/data
[6] https://www.kdd.in.tum.de/gnn-benchmark

Table 2
Node classification results using extremely limited labels (# T means # Label per Class).

GNN +MIXUP	CORA		CITESEER		PUBMED		COAUTHOR CS		COAUTHOR PHYSICS	
	# T = 1	# T = 2	# T = 1	# T = 2	# T = 1	# T = 2	# T = 1	# T = 2	# T = 1	# T = 2
GCN [1]	41.81$_{\pm 5.3}$	56.05$_{\pm 4.8}$	33.67$_{\pm 7.2}$	46.13$_{\pm 7.3}$	55.71$_{\pm 6.0}$	61.81$_{\pm 5.1}$	60.39$_{\pm 7.8}$	77.83$_{\pm 3.8}$	76.37$_{\pm 13.8}$	87.72$_{\pm 2.7}$
+NODEMIXUP	45.21$_{\pm 8.8}$	57.35$_{\pm 5.4}$	36.10$_{\pm 8.8}$	47.80$_{\pm 6.2}$	57.55$_{\pm 6.4}$	62.87$_{\pm 5.2}$	69.56$_{\pm 6.5}$	81.25$_{\pm 3.7}$	79.98$_{\pm 10.3}$	87.99$_{\pm 2.6}$
Improv. (%)	8.13↑	2.32↑	7.22↑	3.62↑	3.30↑	1.71↑	15.18↑	4.39↑	4.73↑	0.31↑
+G-NODEMIXUP	**49.10**$_{\pm 6.8}$	**59.16**$_{\pm 5.6}$	**39.49**$_{\pm 4.9}$	**48.76**$_{\pm 5.1}$	**58.54**$_{\pm 6.7}$	**65.65**$_{\pm 4.5}$	**73.74**$_{\pm 6.2}$	**84.53**$_{\pm 2.7}$	**83.75**$_{\pm 4.4}$	**89.32**$_{\pm 1.5}$
Improv. (%)	**17.44↑**	**5.55↑**	**17.29↑**	**5.70↑**	**5.08↑**	**6.21↑**	**22.11↑**	**8.61↑**	**9.66↑**	**1.82↑**
GAT [2]	47.84$_{\pm 8.5}$	60.27$_{\pm 5.3}$	34.96$_{\pm 7.9}$	48.22$_{\pm 5.8}$	54.10$_{\pm 7.2}$	60.69$_{\pm 5.0}$	77.00$_{\pm 4.6}$	84.42$_{\pm 3.1}$	77.54$_{\pm 14.3}$	87.25$_{\pm 2.6}$
+NODEMIXUP	47.95$_{\pm 6.6}$	60.96$_{\pm 4.4}$	37.85$_{\pm 8.4}$	49.07$_{\pm 5.9}$	57.20$_{\pm 6.3}$	61.95$_{\pm 5.1}$	77.62$_{\pm 6.3}$	85.48$_{\pm 3.7}$	80.92$_{\pm 7.9}$	88.95$_{\pm 1.9}$
Improv. (%)	0.23↑	1.14↑	8.27↑	1.76↑	5.73↑	2.08↑	0.81↑	1.26↑	4.36↑	1.95↑
+G-NODEMIXUP	**49.29**$_{\pm 7.6}$	**62.40**$_{\pm 4.5}$	**39.74**$_{\pm 8.4}$	**49.30**$_{\pm 5.8}$	**57.30**$_{\pm 8.6}$	**62.57**$_{\pm 4.8}$	**78.60**$_{\pm 4.8}$	**86.36**$_{\pm 2.8}$	**82.16**$_{\pm 5.1}$	**89.61**$_{\pm 3.2}$
Improv. (%)	**3.03↑**	**3.53↑**	**13.67↑**	**2.24↑**	**5.91↑**	**3.10↑**	**2.08↑**	**2.30↑**	**5.96↑**	**2.70↑**
SUPERGAT [53]	46.85$_{\pm 8.8}$	60.59$_{\pm 5.4}$	36.11$_{\pm 8.0}$	48.39$_{\pm 6.1}$	56.61$_{\pm 7.8}$	60.69$_{\pm 4.9}$	78.15$_{\pm 3.5}$	84.19$_{\pm 2.6}$	78.30$_{\pm 9.4}$	87.48$_{\pm 3.3}$
+NODEMIXUP	47.00$_{\pm 8.0}$	61.42$_{\pm 5.5}$	37.36$_{\pm 7.6}$	49.19$_{\pm 6.1}$	56.76$_{\pm 5.2}$	63.11$_{\pm 5.0}$	78.89$_{\pm 6.0}$	84.69$_{\pm 3.6}$	80.96$_{\pm 7.6}$	87.64$_{\pm 2.9}$
Improv. (%)	0.32↑	1.37↑	3.46↑	1.47↑	0.26↑	3.99↑	0.95↑	0.59↑	3.40↑	0.18↑
+G-NODEMIXUP	**49.68**$_{\pm 5.6}$	**62.46**$_{\pm 4.6}$	**38.10**$_{\pm 8.2}$	**49.68**$_{\pm 7.4}$	**57.72**$_{\pm 6.1}$	**63.78**$_{\pm 3.6}$	**79.84**$_{\pm 2.3}$	**86.30**$_{\pm 2.2}$	**82.41**$_{\pm 4.6}$	**89.74**$_{\pm 2.8}$
Improv. (%)	**6.04↑**	**3.09↑**	**5.51↑**	**2.67↑**	**1.96↑**	**5.09↑**	**2.16↑**	**2.51↑**	**5.25↑**	**2.58↑**
GRAPHSAGE [6]	36.61$_{\pm 6.1}$	49.77$_{\pm 5.6}$	32.37$_{\pm 7.3}$	44.42$_{\pm 4.2}$	53.29$_{\pm 7.5}$	59.50$_{\pm 4.8}$	65.63$_{\pm 8.0}$	81.98$_{\pm 2.6}$	75.33$_{\pm 13.2}$	86.37$_{\pm 2.8}$
+NODEMIXUP	42.74$_{\pm 4.9}$	56.58$_{\pm 5.0}$	37.12$_{\pm 7.8}$	46.00$_{\pm 6.7}$	55.14$_{\pm 6.1}$	59.73$_{\pm 5.1}$	71.41$_{\pm 5.0}$	82.61$_{\pm 2.8}$	76.05$_{\pm 11.1}$	86.91$_{\pm 2.3}$
Improv. (%)	16.74↑	13.68↑	14.67↑	3.56↑	3.47↑	0.39↑	8.81↑	0.77↑	0.96↑	0.63↑
+G-NODEMIXUP	**45.57**$_{\pm 5.2}$	**56.81**$_{\pm 4.4}$	**38.06**$_{\pm 7.5}$	**46.81**$_{\pm 6.0}$	**56.91**$_{\pm 5.9}$	**60.50**$_{\pm 4.9}$	**73.04**$_{\pm 4.0}$	**83.23**$_{\pm 2.6}$	**80.35**$_{\pm 4.9}$	**86.96**$_{\pm 1.9}$
Improv. (%)	**24.47↑**	**14.15↑**	**17.58↑**	**5.38↑**	**6.79↑**	**1.68↑**	**11.29↑**	**1.52↑**	**6.66↑**	**0.68↑**
UNIMP [54]	40.19$_{\pm 6.4}$	52.61$_{\pm 6.9}$	35.49$_{\pm 6.9}$	46.68$_{\pm 5.3}$	53.45$_{\pm 6.0}$	60.01$_{\pm 4.4}$	69.41$_{\pm 8.6}$	79.41$_{\pm 4.6}$	75.16$_{\pm 13.7}$	86.32$_{\pm 3.1}$
+NODEMIXUP	42.27$_{\pm 4.1}$	53.63$_{\pm 5.1}$	35.60$_{\pm 6.2}$	46.73$_{\pm 7.1}$	55.09$_{\pm 5.4}$	60.04$_{\pm 4.7}$	71.43$_{\pm 6.4}$	82.01$_{\pm 3.4}$	79.53$_{\pm 9.0}$	87.30$_{\pm 2.5}$
Improv. (%)	5.18↑	1.94↑	0.31↑	0.11↑	3.07↑	0.05↑	2.91↑	3.27↑	5.81↑	1.14↑
+G-NODEMIXUP	**44.93**$_{\pm 4.8}$	**54.61**$_{\pm 5.1}$	**37.15**$_{\pm 7.2}$	**47.14**$_{\pm 5.1}$	**56.17**$_{\pm 5.8}$	**60.54**$_{\pm 4.4}$	**71.68**$_{\pm 8.0}$	**83.01**$_{\pm 2.5}$	**81.76**$_{\pm 3.7}$	**87.99**$_{\pm 1.8}$
Improv. (%)	**11.79↑**	**3.80↑**	**4.68↑**	**0.99↑**	**5.09↑**	**0.88↑**	**3.27↑**	**4.53↑**	**8.78↑**	**1.93↑**
APPNP [5]	51.04$_{\pm 10.1}$	64.34$_{\pm 7.8}$	33.19$_{\pm 10.9}$	48.34$_{\pm 8.4}$	58.72$_{\pm 9.3}$	65.36$_{\pm 5.4}$	67.22$_{\pm 7.7}$	81.83$_{\pm 3.0}$	83.34$_{\pm 4.9}$	88.69$_{\pm 2.9}$
+NODEMIXUP	55.67$_{\pm 10.7}$	**65.37**$_{\pm 5.3}$	39.04$_{\pm 10.1}$	49.93$_{\pm 6.7}$	60.24$_{\pm 7.9}$	**67.52**$_{\pm 3.9}$	72.53$_{\pm 3.3}$	82.59$_{\pm 3.6}$	85.42$_{\pm 4.1}$	90.08$_{\pm 2.5}$
Improv. (%)	9.07↑	**1.60↑**	17.63↑	3.29↑	2.59↑	**3.30↑**	7.90↑	0.93↑	2.50↑	1.57↑
+G-NODEMIXUP	**56.93**$_{\pm 7.6}$	65.31$_{\pm 5.9}$	**42.25**$_{\pm 8.9}$	**52.52**$_{\pm 7.2}$	**61.97**$_{\pm 7.6}$	67.01$_{\pm 4.2}$	**74.79**$_{\pm 6.5}$	**84.68**$_{\pm 2.8}$	**86.33**$_{\pm 3.4}$	**90.55**$_{\pm 1.3}$
Improv. (%)	**11.54↑**	1.52↑	**27.30↑**	**8.65↑**	**5.53↑**	2.52↑	**11.26↑**	**3.48↑**	**3.59↑**	**2.10↑**

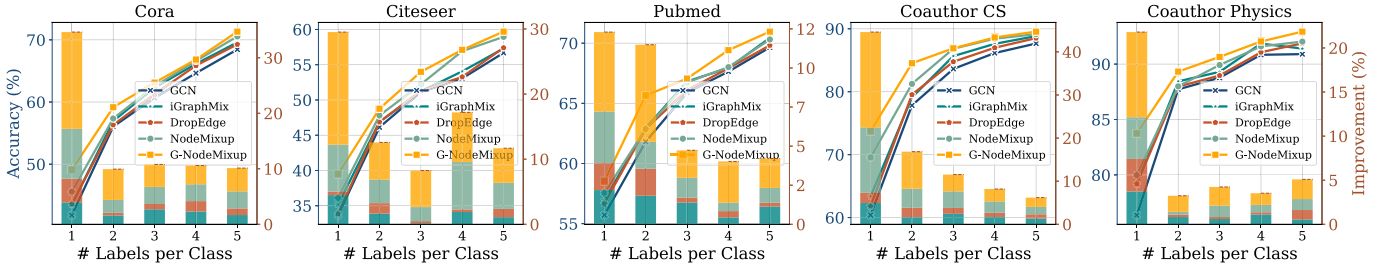


Fig. 6. Performance comparison of various graph augmentation methods using GCN as the backbone across five benchmark datasets, with the number of labels per class ranging from 1 to 5. The plot utilizes a dual y-axis scheme for comprehensive analysis: the line plots depict model accuracy and correspond to the blue y-axis on the left, while the stacked bar charts represent the percentage improvement over the GCN baseline, corresponding to the red y-axis on the right. This visualization highlights both the absolute performance and relative gains of each method.

node predictions without directly relying on potentially noisy pseudo-labels. This not only reduces the negative impact of pseudo-label noise but also strengthens the generalization capabilities of the model.

DROPEDGE and IGRAPHMIX also show improvements over GCN, but their overall gains are less pronounced compared to both NODEMIXUP and G-NODEMIXUP. DROPEDGE, which stochastically drops edges during training to improve generalization, fails to effectively address the under-reaching problem, especially under extreme label scarcity. IGRAPHMIX, which applies mixup in the input space, improves the feature representations but lacks the capability to enhance the interaction between labeled and unlabeled nodes as effectively as G-NODEMIXUP.

Fig. 6 clearly illustrates the superiority of our proposed G-NODEMIXUP over the baseline methods. Across all five datasets and under all label settings, G-NODEMIXUP consistently achieves the highest absolute accuracy, as indicated by its line plot occupying the top position. Furthermore, it delivers the largest percentage improvement over the GCN baseline, represented by the largest segment in the stacked bar charts. This performance advantage is particularly significant in the most challenging, label-scarce scenarios (e.g., when #T = 1 or 2), which strongly demonstrates the effectiveness and robustness of our framework.

6.4. Ablation study (RQ3)

In this section, we conduct an ablation study to investigate the effectiveness of the key components of G-NODEMIXUP with # T = 1 and 2 using GCN as the backbone model. Specifically, we examine the impact of removing different modules from G-NODEMIXUP, including L-L mixup (w/o LL), L-U mixup (w/o LU), U-U mixup (w/o UU), edge mixup (w/o EM), feature mixup (w/o FM), and consistency regularization-based

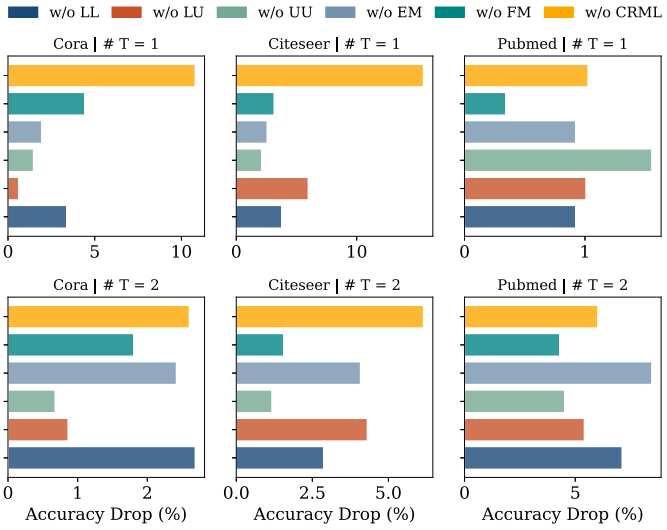


Fig. 7. Ablation study results on three datasets with G-NodeMixup using GCN as the backbone model, comparing accuracy drops (%) caused by removing different components of G-NodeMixup (**w/o LL**, **w/o LU**, **w/o UU**, **w/o EM**, **w/o FM**, **w/o CRML**) under extremely limited label settings (# $T = 1$ and # $T = 2$).

mixup loss (w/o CRML). The experimental results for three benchmark datasets are shown in Fig. 7.

The ablation results reveal that removing any component from G-NodeMixup generally leads to a decline in model performance, demonstrating that all components contribute to its effectiveness. Among the different components, the removal of CRML results in the most significant drop in performance, particularly in the Citeseer dataset, where the accuracy drops by 15.57 % and 5.72 % for # $T = 1$ and # $T = 2$, respectively. This highlights the importance of CRML, which addresses the reliance on noisy pseudo-labels by promoting smooth transitions between node predictions.

Removing L-L mixup also results in notable accuracy drops, especially for Cora and Pubmed datasets. This is because Labeled-Labeled Mixup directly enhances the intra-class cohesion by leveraging reliable labeled nodes to reinforce feature learning. On the other hand, L-U and U-U mixup also contribute to the overall effectiveness of G-NodeMixup by promoting interactions between labeled and unlabeled nodes and enforcing consistency within the unlabeled space, respectively.

The edge and feature mixup are crucial for augmenting the graph structure and features, respectively. Removing these mixup components leads to consistent performance drops, indicating that both structural and feature-level mixups are necessary to improve communication and feature representation in extremely limited label settings.

Overall, the ablation results highlight the effectiveness of each component of G-NodeMixup. MSP, SGMixup, and CRML collectively contribute to the robustness and generalization capabilities of G-NodeMixup, leading to significant improvements in classification accuracy even under challenging label-scarce scenarios.

6.5. Hyperparameter sensitivity study (RQ4)

In this section, we investigate the sensitivity of G-NodeMixup to its key hyperparameters. We focus on two main aspects: the hop parameter k, which defines the size of the k-hop subgraph that serves as the local region for our mixup operations, and the loss-balancing coefficients α_{LLmix} and α_{CRML}.

To investigate the impact of the hop parameter k and determine its optimal value, we conduct a sensitivity analysis. We evaluate the performance of G-NodeMixup on five datasets by varying k within the range $\{1, 2, 3, 4, 5\}$ and keeping other hyperparameters fixed. The results, presented in Fig. 8, show a clear and consistent trend: the model's accuracy on all datasets first increases and then declines as k grows, with performance typically peaking at $k = 2$ or $k = 3$.

This observed trend highlights a critical trade-off: while increasing the hop parameter from $k = 1$ to $k = 2$ is beneficial for expanding the receptive field and mitigating the under-reaching issue, further increases ($k \geq 3$) can be detrimental. This is likely because larger values of k introduce noise from irrelevant higher-order neighbors, which dilute the meaningful local structural information crucial for classification accuracy. Moreover, increasing k also leads to a significant rise in computational complexity, as shown in Fig. 9. Therefore, our choice of $k = 2$ for the main experiments represents a robust and empirically validated balance among enhancing node reachability, preserving local structure, and maintaining computational efficiency.

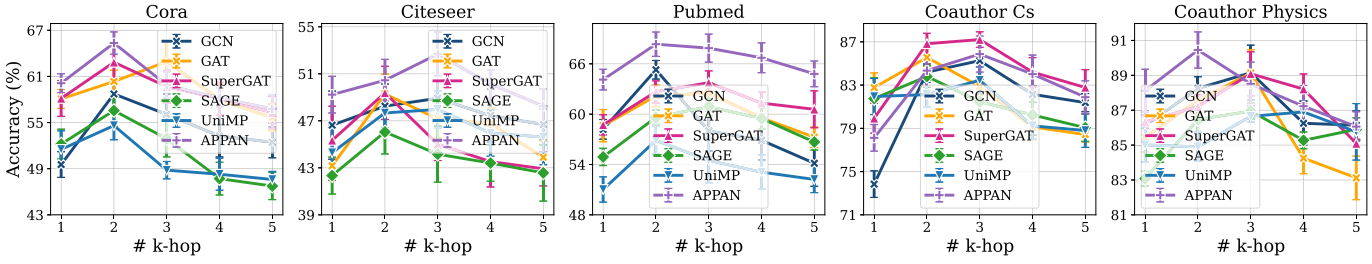


Fig. 8. Sensitivity of the model's accuracy to the hop parameter k.

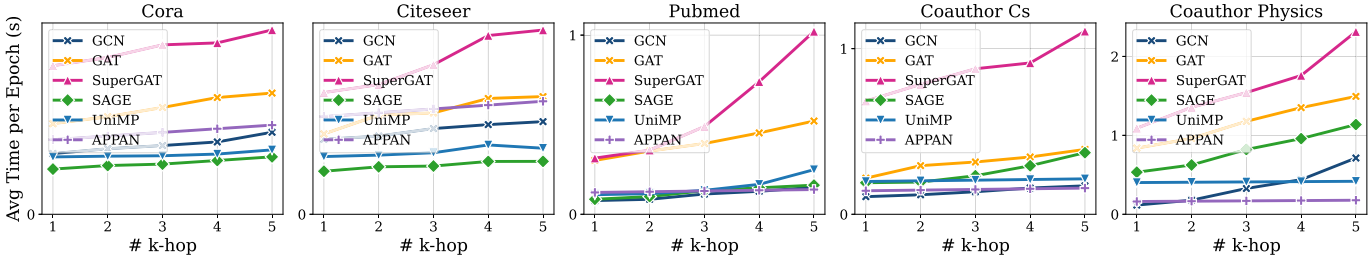


Fig. 9. Sensitivity of the model's running time to the hop parameter k.

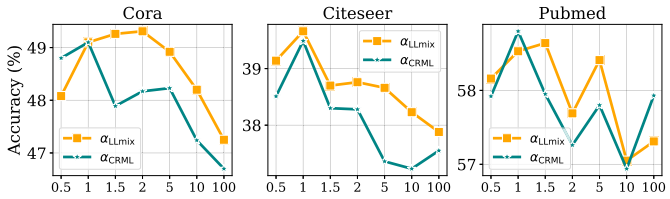


Fig. 10. The impact of varying hyperparameters α_{LLmix} and α_{CRML} on the classification accuracy of G-NodeMixup under # $T = 1$ label setting.

We evaluate the impact of different values of α_{LLmix} and α_{CRML} on G-NodeMixup. This experiment is conducted for label setting # $T = 1$. The goal is to assess how varying the contributions of $\mathcal{L}_{\text{CRML}}$ and $\mathcal{L}_{\text{LLmix}}$ affect the model.

The results, as shown in Fig. 10, reveal several important observations regarding the sensitivity of G-NodeMixup to these hyperparameters. For α_{LLmix}, the accuracy of all datasets shows a peak around the middle values, with a notable decline as α_{LLmix} becomes very small or very large. This suggests that an optimal contribution from L-L mixup is crucial to achieve the best performance, as an imbalance can either weaken its positive influence or cause overfitting due to over-emphasizing this mixup term. In the case of α_{CRML}, the performance also exhibits a similar trend where moderate values generally yield better results, while very high or low values negatively impact the accuracy. This indicates that both L-U and U-U mixup play important roles in stabilizing the predictions, but excessive regularization may lead to restricted flexibility in learning feature representations.

7. Conclusion

In this paper, we propose G-NodeMixup, a novel architecture-agnostic framework designed to address the challenges of extremely limited label settings in semi-supervised node classification. Compared to our previously proposed NodeMixup, we introduce three key innovations: Multi-set Pairing, which enables more comprehensive data augmentation through diverse pairwise mixups; Subgraph-based Mixup, which mitigates the structural disruptions caused by global edge modifications; and Consistency Regularization-based Mixup Loss, which reduces the dependence on potentially noisy pseudo-labels by enforcing smooth transitions in node predictions. Extensive experiments across multiple GNN backbones and benchmark datasets demonstrate that G-NodeMixup consistently outperforms existing methods in extremely limited label settings. Our approach provides a simple yet effective augmentation framework that can be seamlessly integrated into various GNN architectures without significant computational overhead. This makes G-NodeMixup a practical solution for real-world applications.

CRediT authorship contribution statement

Ziyu Guan: Writing – review & editing, Methodology, Conceptualization. **Beilei Ling:** Writing – original draft, Software, Methodology. **Weigang Lu:** Supervision, Conceptualization. **Meng Yan:** Software, Data curation. **Yaming Yang:** Software, Data curation. **Wei Zhao:** Supervision, Funding acquisition. **Yibing Zhan:** Investigation. **Dapeng Tao:** Visualization, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62133012, 62425605, and 62303366, in part by the Key Research and Development Program of Shaanxi under Grants 2024CY2-GJHX-15, 2022ZDLGY01-10, and 2025CY-YBXM-041, and in part by the Xidian University Specially Funded Project for Interdisciplinary Exploration under Grant TZJHF202506, and the Fundamental Research Funds for the Central Universities via the Innovation Fund of Xidian University under Grant YJSJ25012.

Data availability

I have shared the link to my data/code in the attached file step.

References

[1] N.T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.
[2] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[3] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, ICML (2019) 6861–6871.
[4] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, ICML (2020) 1725–1735.
[5] J. Klicpera, A. Bojchevski, S. Günnemann, Predict Then Propagate: Graph Neural Networks Meet Personalized Pagerank, ICLR, in, 2019.
[6] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, NeurIPS (2017).
[7] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum Chemistry, ICML (2017) 1263–1272.
[8] Q. Zheng, X. Xia, K. Zhang, E. Kharlamov, Y. Dong, On the distribution alignment of propagation in graph neural networks, AI Open 3 (2022) 218–228.
[9] Q. Sun, J. Li, H. Yuan, X. Fu, H. Peng, C. Ji, Q. Li, P.S. Yu, 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, 2022, pp. 1848–1857.
[10] F. Di Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lio, M. Bronstein, On over-squashing in message passing neural networks: the impact of width, depth, and topology, ICML (2023).
[11] H. Attali, D. Buscaldi, N. Pernelle, Curvature constrained MPNNS: improving message passing with local structural properties, Data Knowl. Eng. 156 (2025) 102382.
[12] K. Nguyen, N.M. Hieu, V.D. Nguyen, N. Ho, S. Osher, T.M. Nguyen, Revisiting over-smoothing and over-squashing using Ollivier-Ricci curvature, in: International Conference on Machine Learning, 2023, pp. 25956–25979.
[13] R. Brüel-Gabrielsson, M. Yurochkin, J. Solomon, Rewiring with positional encodings for graph neural networks, ICLR (2023).
[14] P. Barceló, E.V. Kostylev, M. Monet, J. Pérez, J. Reutter, J.-P. Silva, The logical expressiveness of graph neural networks, ICLR (2020).
[15] T.U. Do, V.C. Ta, Tackling under-reaching issue in beta-wavelet filters with mixup augmentation for graph anomaly detection, Expert Syst. Appl. (2025) 127033.
[16] C. Qian, A. Manolache, K. Ahmed, Z. Zeng, G.V.D. Broeck, M. Niepert, C. Morris, Probabilistically rewired message-passing neural networks, ICLR (2024).
[17] C. Qian, A. Manolache, C. Morris, M. Niepert, Probabilistic Graph Rewiring via Virtual Nodes, NeurIPS, in, 2024.
[18] Q. Li, Z. Han, X.-M. Wu, Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning, AAAI, in, 2018.
[19] W. Lu, Y. Zhan, B. Lin, Z. Guan, L. Liu, B. Yu, W. Zhao, Y. Yang, D. Tao, Skipnode: on alleviating performance degradation for deep graph convolutional networks, TKDE (2024).
[20] S. Stanovic, B. Gaüzère, L. Brun, Graph neural networks with maximal independent set-based pooling: mitigating over-smoothing and over-squashing, Pattern Recognit. Lett. 187 (2025) 14–20.
[21] Q. Zhang, J. Li, Q. Ye, Y. Lin, X. Chen, Y.-G. Fu, DWSSA: alleviating over-smoothness for deep graph neural networks, Neural Netw. 174 (2024) 106228.
[22] Q. Cheng, L. Long, J. Xu, M. Zhang, S. Han, C. Zhao, W. Feng, A universal strategy for smoothing deceleration in deep graph neural networks, Neural Netw. 185 (2025) 107132.
[23] U. Alon, E. Yahav, On the bottleneck of graph neural networks and its practical implications, ICML (2021).
[24] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: Beyond Empirical Risk Minimization, ICLR, in, 2018.
[25] L. Wu, J. Xia, Z. Gao, H. Lin, C. Tan, S.Z. Li, Graphmixup: improving class-imbalanced node classification by reinforcement mixup and self-supervised context prediction, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2022, pp. 519–535.
[26] V. Verma, M. Qu, K. Kawaguchi, A. Lamb, Y. Bengio, J. Kannala, J. Tang, Graphmix: improved training of GNNS for semi-supervised learning, AAAI 35 (2021) 10024–10032.
[27] Y. Wang, W. Wang, Y. Liang, Y. Cai, B. Hooi, Mixup for Node and Graph Classification, WWW, in, 2021, pp. 3663–3674.
[28] H. Ling, Z. Jiang, M. Liu, S. Ji, N. Zou, Graph mixup with soft alignments, in: International Conference on Machine Learning, PMLR, 2023, pp. 21335–21349.
[29] X. Ma, X. Chu, Y. Wang, Y. Lin, J. Zhao, L. Ma, W. Zhu, Fused Gromov-Wasserstein graph mixup for graph-level classifications, Adv. Neural Inf. Process. Syst. 36 (2023) 15252–15276.
[30] A. Zeng, L. Wang, W. Zhang, X. Lin, Efficient and effective augmentation framework with latent mixup and label-guided contrastive learning for graph classification, in: IEEE Transactions on Knowledge and Data Engineering, IEEE, 2024.

[31] X. Han, Z. Jiang, N. Liu, X. Hu, G-mixup: graph data augmentation for graph classification, in: International Conference on Machine Learning, PMLR, 2022, pp. 8230–8248.
[32] S. Zheng, H. Wang, X. Liu, Intramix: intra-class mixup generation for accurate labels and neighbors, Adv. Neural Inf. Process. Syst. 37 (2024) 75095–75124.
[33] E. Chien, J. Peng, P. Li, O. Milenkovic, Adaptive universal generalized pagerank graph neural network, ICML (2021).
[34] J. Li, G. Lu, Z. Wu, F. Ling, Multi-view representation model based on graph autoencoder, Inf. Sci. 632 (2023) 439–453.
[35] C.M. Bishop, N.M. Nasrabadi, Pattern Recognition and Machine Learning, vol. 4, Springer, 2006.
[36] D. Crisostomi, S. Antonelli, V. Maiorca, L. Moschella, R. Marin, E. Rodolà, Metric based few-shot graph classification, in: Learning on Graphs Conference, 2022, pp. 33–41.
[37] J. Park, H. Shim, E. Yang, Graph transplant: node saliency-guided graph mixup with local structure preservation, AAAI 36 (2022) 7966–7974.
[38] S.Y. Kim, C.P. Junghurn, S-MIXUP: Structural Mixup for Graph Neural Networks, CIKM, in, 2023.
[39] Y. Kong, J. Li, K. Zhang, J. Wu, Multi-scale self-attention mixup for graph classification, Pattern Recognit. Lett. 168 (2023) 100–106.
[40] T. Jia, H. Li, C. Yang, T. Tao, C. Shi, Graph invariant learning with subgraph co-mixup for out-of-distribution generalization, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, 2024, pp. 8562–8570.
[41] L. Weigang, Z. Guan, W. Zhao, Y. Yang, Y. Zhan, L. Yiheng, D. Tao, AGMIXUP: adaptive graph mixup for semi-supervised node classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, 2025, pp. 19143–19151.
[42] J. Topping, D.F. Giovanni, P.B. Chamberlain, X. Dong, M.M. Bronstein, Understanding over-squashing and bottlenecks on graphs via curvature, ICML (2022).
[43] H. Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function, J. Stat. Plan. Inference 90 (2) (2000) 227–244.
[44] L. Du, G. Lu, Y. Han, Z. Luo, G. Wen, L. Zhang, W. Chen, S. Zhang, Graphdhv: graph neural network with dual hybrid view on imbalanced node classification, in: International Computing and Combinatorics Conference, Springer, 2024, pp. 502–513.
[45] E. Buchnik, E. Cohen, Bootstrapped graph diffusions: exposing the power of nonlinearity, in: Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems, 2018, pp. 8–10.
[46] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. (2016).
[47] S. Kornblith, M. Norouzi, H. Lee, G. Hinton, Similarity of neural network representations revisited, ICML (2019) 3519–3529.
[48] J. Zhu, Y. Yan, M. Heimann, L. Zhao, L. Akoglu, D. Koutra, Heterophily and graph neural networks: past, present and future, IEEE Data Eng. Bull. 10 (2023).
[49] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, C.A. Raffel, Mixmatch: a holistic approach to semi-supervised learning, Adv. Neural Inf. Process. Syst. (2019).
[50] M. Fey, J.E. Lenssen, Fast graph representation learning with PYTORCH geometric, in: ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.
[51] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, ICML (2016) 40–48.
[52] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, arxiv:1811.05868, (2018), arXiv preprint.
[53] D. Kim, A. Oh, How to find your friendly neighborhood: graph attention design with self-supervision, ICML (2021).
[54] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun, Masked label prediction: unified message passing model for semi-supervised classification, ICML (2021).
[55] W. Lu, Z. Guan, W. Zhao, Y. Yang, L. Jin, Nodemixup: tackling under-reaching for graph neural networks, AAAI 38 (2024) 14175–14183.
[56] J. Jeong, H. Lee, H.G. Yoon, B. Lee, J. Heo, G. Kim, K.J. Seon, Igraphmix: input graph mixup method for node classification, in: ICLR, 2024.
[57] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: Towards Deep Graph Convolutional Networks on Node Classification, ICLR, in, 2020.
[58] C. Zou, L. Guangquan, D. Longqing, X. Zeng, S. Lin, Graph similarity learning for cross-level interactions, Info. Process. Manag. 62(1) 2025 103932. Publisher: Elsevier.
[59] Y. Han, L. Guangquan, S. Zhang, L. Zhang, C. Zou, G. Wen, A temporal knowledge graph embedding model based on variable translation, Tsinghua Sci. Technol. 29(5) 2024 1554–1565. Publisher: TUP.
[60] D.P. Kingma, B. Jimmy, Adam: A Method for Stochastic Optimization, ICLR, in, 2015.

Author biography

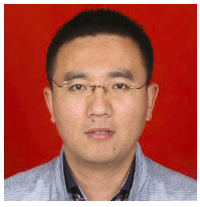
Ziyu Guan received the B.S. and Ph.D. degrees in Computer Science from Zhejiang University, Hangzhou China, in 2004 and 2010, respectively. He had worked as a research scientist in the University of California at Santa Barbara from 2010 to 2012, and as a professor in the School of Information and Technology of Northwest University, China from 2012 to 2018. He is currently a professor with the School of Computer Science and Technology, Xidian University. His research interests include attributed graph mining and search, machine learning, expertise modeling and retrieval, and recommender systems.

Beilei Ling received the B.S. and M.S. degrees in Computer Science and Technology from Xi'an University of Posts and Telecommunications, China, in 2015 and 2018, respectively. She is currently working towards a Ph.D. degree at the School of Computer Science and Technology, Xidian University. Her research interests include graph data mining and machine learning.

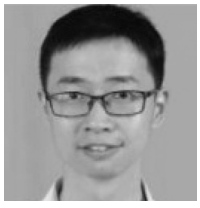
Weigang Lu is a Postdoctoral Fellow at the Department of Civil and Enviromental Engineering at The Hong Kong University of Science and Technology (HKUST). He received his Ph.D. in Computer Science from Xidian University. His research focuses on semi-supervised learning and graph neural networks, with work published in top-tier conferences and journals such as NeurIPS, SIGKDD, AAAI, ICDE, and IEEE TKDE. He also serves as a reviewer for leading academic venues, including IEEE TKDE, ACM SIGKDD, AAAI, The Web Conference (WebConf), Neural Networks, and ICDM.

Meng Yan received the B.S degree in information security from Guangxi University, Nanning,China, in 2019. She is currently working toward the Ph.D. degree in computer science with the School of Computer Science and Technology, Xidian University, Xi'an, China. Her current research interests include recommender systems and multi-modal learning.

Yaming Yang received the B.S. and Ph.D. degrees in Computer Science and Technology from Xidian University, China, in 2015 and 2022, respectively. He is currently a lecturer with the School of Computer Science and Technology at Xidian University. His research interests include data mining and machine learning on graph data.



Wei Zhao received the B.S., M.S. and Ph.D. degrees from Xidian University, Xi'an, China, in 2002, 2005 and 2015, respectively. He is currently a professor in the School of Computer Science and Technology at Xidian University. His research direction is pattern recognition and intelligent systems, with specific interests in attributed graph mining and search, machine learning, signal processing and precision guiding technology.



Yibing Zhan received a B.E. and a Ph.D. from the University of Science and Technology of China in 2012 and 2018, respectively. From 2018 to 2020, he was an Associate Researcher in the Computer and Software School at the Hangzhou Dianzi University. He is currently an algorithm scientist at the JD Explore Academy. His research interest includes graph neural networks and multimodal learning. He has published many papers on top conferences and journals, such as CVPR, ACM MM, AAAI, IJCV, and IEEE TMM.

Dapeng Tao received the B.E. degree from Northwestern Polytechnical University, Xi'an, China, and the Ph.D. degree from the South China University of Technology, Guangzhou, China. He is currently with the School of Information Science and Engineering, Yunnan University, Kunming, China, as an Engineer. He has authored or co-authored over 30 scientific articles. His current research interests include machine learning, computer vision, and cloud computing. Dr. Tao has served more than ten international journals, including the IEEE Transactions on Neural Networks and Learning Systems, the IEEE Transactions on Multimedia, the IEEE Signal Processing Letters, and PLOS-ONE.