AGMixup: Adaptive Graph Mixup for Semi-supervised Node Classification

Weigang Lu[1], Ziyu Guan[1*], Wei Zhao[1], Yaming Yang[1], Yibing Zhan[2], Yiheng Lu[1], Dapeng Tao[3]

[1]School of Computer Science and Technology, Xidian University, Xi'an, China
[2]JD Explore Academy, Xidian University, Beijing, China
[3]School of Information Science and Engineering, Yunnan University, Kunming, China
{wglu@.stu., zyguan@, ywzhao@mail., yym@,}xidan.edu.cn, zhanyibing@jd.com, dapeng.tao@gmail.com

Abstract

Mixup is a data augmentation technique that enhances model generalization by interpolating between data points using a mixing ratio λ in the image domain. Recently, the concept of mixup has been adapted to the graph domain through *node-centric* interpolations. However, these approaches often fail to address the complexity of interconnected relationships, potentially damaging the graph's natural topology and undermining node interactions. Furthermore, current graph mixup methods employ a *one-size-fits-all* strategy with a randomly sampled λ for all mixup pairs, ignoring the diverse needs of different pairs. This paper proposes an Adaptive Graph Mixup (`AGMixup`) framework for semi-supervised node classification. `AGMixup` introduces a subgraph-centric approach, which treats each subgraph similarly to how images are handled in Euclidean domains, thus facilitating a more natural integration of mixup into graph-based learning. We also propose an adaptive mechanism to tune the mixing ratio λ for diverse mixup pairs, guided by the contextual similarity and uncertainty of the involved subgraphs. Extensive experiments across seven datasets on semi-supervised node classification benchmarks demonstrate `AGMixup`'s superiority over state-of-the-art graph mixup methods.

Introduction

Graph Neural Networks (GNNs) have emerged as a dominant paradigm for learning on graph-structured data, driving significant advances in various graph-based tasks, notably node classification. Despite these successes, a fundamental challenge remains in how to effectively generalize from limited or sparsely labeled data, a common scenario in node classification. Mixup (Zhang et al. 2017), an effective data augmentation method originally devised for Euclidean data, enriches the training dataset by linearly interpolating between pairs of labeled samples (a mixup pair) and their corresponding labels with a mixing coefficient λ. While mixup has achieved notable success in Euclidean data domains characterized by regularity, its application to the complex, non-Euclidean nature of graph data encounters two significant questions.

Question 1: How can the mixup concept be seamlessly integrated into graphs? Existing graph mixup methods, focusing on node-centric interpolations within a graph, extend the mixup method from Euclidean spaces to graphs by blending features, labels, and connections of node pairs (Lu et al. 2024). However, these methods overlook the complexity of interconnected relationships, potentially altering the graph's natural topology and undermining the node interactions. In contrast to the isolation of images in mixup as shown in Fig. 1 (***left***), where interpolations do not influence other data points, graph data's interconnected nature demands an approach to preserve its topological integrity.

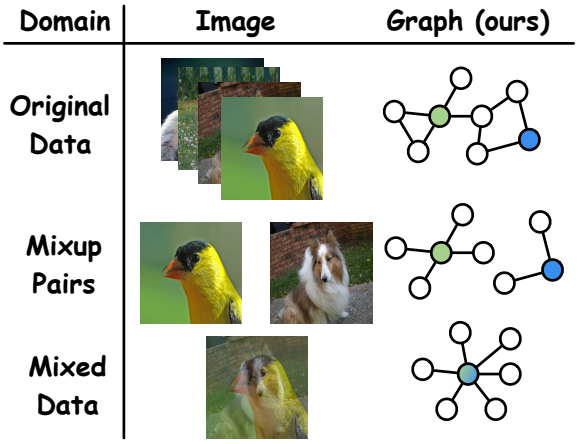

Figure 1: Seamlessly integrating mixup from image domain into graph domain with our `AGMixup`.

Question 2: How can the mixing ratio (λ) be adaptively controlled for different mixup pairs? Prevailing graph mixup techniques often employ a one-size-fits-all strategy to control the mixing ratio, applying a randomly sampled λ across all mixup operations. This approach can inadvertently generate synthetic nodes that starkly contrast with realistic distributions within the graph, especially when merging highly dissimilar node pairs with a mid-range λ. Moreover, this static strategy fails to encourage the model to explore less represented data areas, critical for more confident predictions.

Present Work. In response to these challenges, we propose an Adaptive Graph Mixup (**`AGMixup`**) framework for semi-supervised node classification. `AGMixup` innovates a

*Corresponding Author

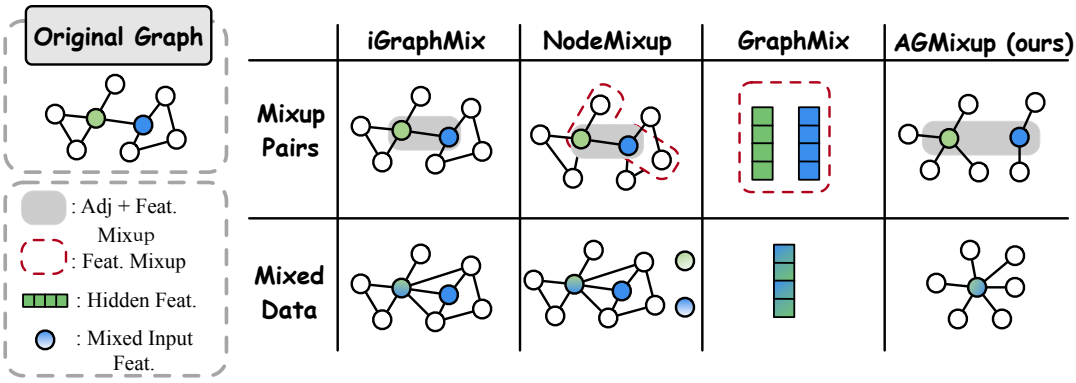

Figure 2: Difference between AGMixup and SOTA graph mixup methods.

subgraph-centric mixup, extending the mixup concept beyond individual nodes. Given that a subgraph can encapsulate a node's local structure and semantic context, AGMixup treats a subgraph as analogous to an image, facilitating the transfer of the mixup idea from images to graphs, as shown in Fig. 1 (***right***). This subgraph-centric approach not only mitigates the potential negative impacts of traditional in-graph mixup by treating each subgraph as an independent sample, but also offers scalability and flexibility in handling graphs of varying sizes. Furthermore, AGMixup introduces an adaptive mechanism for tuning the mixing ratio, guided by the contextual similarity and uncertainty associated with the subgraphs involved. This strategy ensures that λ is optimally adjusted for diverse mixup pairs, leading to more realistic and exploratory synthetic graph data generation. Our main contributions are summarized as follows:

- We extend the mixup concept beyond the level of individual nodes to subgraphs. This methodology treats each subgraph as an analogous unit to an image in Euclidean domains, enabling a seamless integration of mixup into the graph domain.
- We introduce an adaptive mechanism for dynamically tuning the mixing coefficient, based on the contextual similarity and uncertainty associated with the subgraphs involved.
- We have conducted extensive experiments demonstrating that AGMixup outperforms existing state-of-the-art (SOTA) graph mixup methods. Besides, we provide comprehensive empirical analysis to understand the behavior of AGMixup.

Related Work

Mixup on Images. Mixup (Zhang et al. 2017), introduced by Zhang *et al.*, generates virtual training data by linearly interpolating the features and labels of image pairs using a ratio λ sampled from a Beta distribution. This concept has been expanded into the hidden space by methods like ManifoldMixup (Verma et al. 2019) and PatchUp (Faramarzi et al. 2022), which apply interpolation to hidden representations to enhance sample diversity. Building upon this foundation, adaptive mixing policies such as PuzzleMix (Kim, Choo, and Song 2020), SaliencyMix (Uddin et al. 2021), AutoMix (Liu et al. 2022), SuperMix (Dabouei et al. 2021), and Decoupled Mixup (Liu et al. 2024) have been developed to refine the generation of mixed samples, tailoring the process to the specific characteristics of input data. Theoretical and empirical investigations (Zhang et al. 2021, 2022) into mixup's effects on model generalization and calibration have further demonstrated its benefits, offering deep insights into mixup methods.

Mixup on Graphs. Extending mixup to the graph domain has opened new avenues for enhancing graph-level tasks, with several studies (Ma et al. 2024; Crisostomi et al. 2022a; Han et al. 2022; Guo and Mao 2021; Navarro and Segarra 2022; Park, Shim, and Yang 2022; Crisostomi et al. 2022b) introducing mixup techniques to facilitate these tasks. For graph-level tasks, the integrity of the graph structure is not tied to specific nodes but to the overall graph. As a results, these methods might not scale efficiently when applied directly to node-level tasks since each node requires consideration of its unique neighborhood, which isn't necessarily the focus of whole-graph mixup methods. To address this challenge, (Wang et al. 2021) proposes a method that mixes the input features of pairs of nodes and then blends their aggregated representations at each GNN layer. GraphMixup (Wu et al. 2021) incorporates a reinforcement mechanism to dynamically adjust the scale of mixup pairs, aiming to alleviate the class-imbalanced node classification problem. GraphMix (Verma et al. 2021) applies mixup to the hidden representations of nodes and feeds the mixed nodes into fully connected layers. NodeMixup (Lu et al. 2024) introduces inter- and intra-class mixup techniques for both labeled and unlabeled nodes, addressing the issue of under-reaching. iGraphMix (Jeong et al. 2024) focuses on mixing the features and connections of labeled nodes on the input graph.

Comparison to SOTA Graph Mixup. In Fig. 2, we illustrate the comparison between three SOTA graph mixup methods, i.e., iGraphMix, NodeMixup, GraphMix, and our AGMixup. Both iGraphMix and NodeMixup extend mixup operations to node connections within the graph, risking alterations to the graph's natural topology and potentially impacting node interactions negatively. Furthermore, these

methods might introduce too many edges into graphs, easily inducing over-smoothing (Rong et al. 2019; Oono and Suzuki 2020; Lu et al. 2021), particularly in large-scale graphs. The manifold mixup of GraphMix avoids this challenge by interpolating nodes in the latent space. However, this approach requires additional fully connected layers and may result in a loss of structural information. Besides, its flexibility is limited since hidden representations are not always accessible for linear GNNs (e.g., SGC (Wu et al. 2019)) either determinable for deep GNNs with layers larger than 2. Furthermore, all of these methods employ randomly sampled λ values for all mixup pairs, disregarding the distinct needs of different mixup pairs. Our `AGMixup` addresses these limitations in the following ways: (1) employing subgraph-centric mixup, treating each subgraph as an independent sample; (2) introducing an adaptive mechanism to tune the mixing ratio for diverse mixup pairs.

Methodology

In this section, we begin by revisiting Graph Neural Networks (GNNs) and the mixup technique. Subsequently, we introduce our proposed method, `AGMixup`, which comprises three key modules: subgraph-centric mixup, contextual similarity-aware λ initialization, and uncertainty-aware λ adjustment. Finally, we present a comprehensive case study to evaluate the effectiveness of `AGMixup` and analyze its computational complexity.

Preliminaries

Notations. We define an undirected graph with self-loops, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ represents the set of N nodes, and $\mathcal{E}$ denotes the edge set with element e_{ij} indicating a connection between node i and node j. The input feature matrix $X \in \mathbb{R}^{N \times F}$, with its i-th row vector $\mathbf{x}_i$, where F represents the input dimensionality. Besides, the label matrix $Y \in \mathbb{R}^{N \times C}$, where C is the number of classes, includes $\mathbf{y}_i$ as the one-hot encoding of node i's label.

GNN Revisit. Given a node i with its feature vector $\mathbf{x}_i$, a GNN computes the node's embedding $\mathbf{h}_i$ through a series of layer-wise propagation rules. At each layer l, the embedding $\mathbf{h}_i^{(l)}$ is updated as $\mathbf{h}_{i+1}^{(l)} = \sigma\left(W^{(l)} \cdot \text{AGGREGATE}(\{\mathbf{h}_i^{(l)} | j \in \mathcal{N}_i\})\right)$, where σ is a non-linear activation function, $W^{(l)}$ is the learnable parameter of l-th layer, and AGGREGATE is a function that combines the embeddings of node i's neighbors $\mathcal{N}_i$. The initial embedding $\mathbf{h}_i^{(0)}$ is set to the node features, i.e., $\mathbf{h}_i^{(0)} = \mathbf{x}_i$. Upon completion of L layers of propagation, the final embedding $\mathbf{h}_i^{(L)}$ is then passed through a classification layer to obtain the predictions for each node $\mathbf{p}_i = \text{softmax}(W^{(L)}\mathbf{h}_i^{(L)})$, where parameters $W^{(L)}$ maps the final embedding into the prediction space and the softmax function ensures the output $\mathbf{p}_i$ represents a probability distribution over the C classes.

Mixup Revisit. The traditional mixup method is a data augmentation technique that creates virtual training samples by linearly interpolating between pairs of examples. Given two mixup candidates $\mathbf{x}_i$ and $\mathbf{x}_j$ from the labeled data $\mathcal{D}$, the mixup data $\tilde{\mathbf{x}}_{ij}$ can be generated as follows:

$$\tilde{\mathbf{x}}_{ij} = \mathcal{M}(\mathbf{x}_i, \mathbf{x}_j, \lambda) = \lambda \mathbf{x}_i + (1 - \lambda)\mathbf{x}_j, \tag{1}$$

where $\mathcal{M}(.)$ is a mixup function. Here, λ is a mixing coefficient drawn from a Beta distribution $\text{Beta}(\alpha, \alpha)$ with $\alpha > 0$. The value of λ controls the degree to which they are mixed. In the context of a general classification task, we regard the mixed data as additional training data with mixed labels.

Proposed Method: AGMixup

Subgraph-Centric Mixup. Inspired by image mixup techniques, we treat subgraphs induced from nodes as independent samples since a subgraph encapsulates a node's local structure and semantic context. For a node i, its r-ego graph $\mathcal{G}_i^{(r)} = \{\mathcal{V}_i^{(r)}, \mathcal{E}_i^{(r)}\}$ includes all nodes and edges within r hops. Our `AGMixup` innovatively proposes the mixing of such subgraphs, yielding a mixed graph $\tilde{\mathcal{G}}_{ij}$ formulated as:

$$\tilde{\mathcal{G}}_{ij} = \mathcal{M}_g(\mathcal{G}_i^{(r)}, \mathcal{G}_j^{(r)}, \lambda) = \{\tilde{\mathcal{V}}_{ij}^{(r)}, \tilde{\mathcal{E}}_{ij}^{(r)}\}, \tag{2}$$

where $\mathcal{M}_g(.)$ is a mixup function for the graph. The mixed node set $\tilde{\mathcal{V}}_{ij}^{(r)}$ contains all the nodes from both $\mathcal{G}_i^{(r)}$ and $\mathcal{G}_j^{(r)}$, excluding $\mathbf{x}_i$ and $\mathbf{x}_j$. Instead, a virtual node $\tilde{\mathbf{x}}_{ij} = \mathcal{M}(\mathbf{x}_i, \mathbf{x}_j, \lambda_{ij})$ is introduced, where λ_{ij} is dynamically computed through our adaptive mechanism (detailed later). The mixed edge set $\tilde{\mathcal{E}}_{ij}^{(r)}$ includes connections from the virtual node $\tilde{\mathbf{x}}_{ij}$ to the neighbors of both $\mathbf{x}_i$ and $\mathbf{x}_j$. It preserves the structural connectivity inherent to each original subgraph but also facilitates the integration of node features from the two distinct subgraphs. By doing so, `AGMixup` maintains the graph's structural integrity and leverages the combined features to enhance learning efficacy, mirroring how traditional image mixup enhances model generalization by interpolating between distinct image samples.

Contextual Similarity-Aware λ Initialization. A fixed λ drawn from $\text{Beta}(\alpha, \alpha)$ applies the same degree of interpolation across all sample pairs, potentially limiting the model's ability to generalize from the mixed samples without an appropriate α. A too small α predominantly yields λ values near the extremes (close to one or zero), generating virtual samples overly similar to original samples. On the contrary, a larger α biases λ towards 0.5. However, such uniform blending could result in unrealistic feature combinations when mixing highly dissimilar samples, diverging from the natural data distribution. To this end, we propose a similarity-aware λ initialization mechanism:

$$\lambda_{ij}^{(0)} = 0.5 * \exp(-\gamma \|\bar{\mathbf{h}}_i^{(r)} - \bar{\mathbf{h}}_j^{(r)}\|^2), \tag{3}$$

where $\bar{\mathbf{h}}_i^{(r)}$ is the mean encoded embedding vector by a GNN model of all the nodes in $\mathcal{G}_i^{(r)}$ and $\gamma > 0$ adjusts the sensitivity of mixup to contextual similarity. Eq. (3) enables more exploratory mixing only for subgraphs that are similar in hidden space by initializing λ_{ij} value close to 0.5.

Uncertainty-aware λ Adjustment. During the training with labeled nodes sparsely distributed in the graph, some regions of the feature space might be underrepresented, leading to higher uncertainty in those areas. An adaptive mixup

strategy that accounts for uncertainty can ensure that mixed samples from these underrepresented areas are more frequently generated, contributing meaningfully to the learning process. An adaptive λ adjustment, biased towards the node with higher uncertainty, can help the model focus on learning from the underrepresented data, improving its predictive performance across a more diverse set of nodes. As training progresses and the model becomes more confident in certain regions of the feature space (low uncertainty), adjusting λ_{ij} to favor less mixing (closer to 0 or 1) for these samples encourages exploitation. This refinement phase allows the model to learn representations that generalize better to unseen data, enhancing the confidence of models. Based on this, we adjust λ_{ij} based on the uncertainties of mixup candidates:

$$\lambda_{ij} = \lambda_{ij}^{(0)} + \Delta\lambda_{ij} = \lambda_{ij}^{(0)} + \beta(\frac{\bar{u}_i - \bar{u}_j}{U_{max}}), \tag{4}$$

where β is a scaling factor that controls the influence of uncertainty difference on λ_{ij}. Given the mean predicted probability vector $\bar{p}_i^{(r)}$ for all the nodes from $\mathcal{G}_i^{(r)}$, the uncertainty $\bar{u}_i$ is calculated as: $\bar{u}_i = -\sum_c^C \bar{p}_{ic}^{(r)} \log \bar{p}_{ic}^{(r)}$, where $\bar{p}_{ic}^{(r)}$ is the predicted probability over class c. Here, $U_{max} = \log C$ is a normalization term representing the maximum possible uncertainty difference, ensuring that the adjustment factor falls within a reasonable range. To ensure the calculated λ_{ij} is within valid bounds, we apply a clipping operation on it: $\lambda_{ij} = \text{Clip}(\lambda_{ij}, 0, 1)$.

Training with AGMixup. In node classification tasks, the relationship between graph data $\mathcal{G}$ and node labels $\{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$ is modeled through a graph neural network (e.g., GCN), denoted as $g_\theta : \mathcal{G} \mapsto \{\mathbf{y}_1, \cdots, \mathbf{y}_N\}$, with θ representing the network parameters. The optimization is guided by minimizing $\mathcal{L}_G$ as follows:

$$\mathcal{L}_G = \frac{1}{|\mathcal{V}_L|}\ell(g_\theta(\mathcal{G}), \mathbf{y}), \tag{5}$$

where ℓ is the cross-entropy loss. To regularize the training process and encourage the model to learn from these mixed graphs effectively, `AGMixup` employs a mixup loss function:

$$\begin{aligned}\mathcal{L}_M &= \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}} \tilde{\ell}(\tilde{\mathcal{G}}_{ij}, \mathbf{y}_i, \mathbf{y}_j) \\ &= \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}} \lambda_{ij}\ell(g_\theta(\tilde{\mathcal{G}}_{ij}), \mathbf{y}_i) + (1-\lambda_{ij})\ell(g_\theta(\tilde{\mathcal{G}}_{ij}), \mathbf{y}_j),\end{aligned} \tag{6}$$

where $\mathcal{P}$ is the mixup pair index set. *In practice, AGMixup generates a virtual graph $\tilde{\mathcal{G}}$ that encompasses a set of virtual disjoint subgraphs, each subgraph $\tilde{\mathcal{G}}_{ij}$ corresponding to a mixup between pairs of labeled-node-induced graphs from the original graph.* Then, we combine Eq. (5) and Eq. (6) with a balance factor $\mu \in [0, 1]$ as follows:

$$\mathcal{L} = \mathcal{L}_G + \mu\mathcal{L}_M. \tag{7}$$

This mixup loss $\mathcal{L}_M$ serves as a regularization term to be added to the classification loss, guiding the model to not only fit the original data but also to generalize across the mixed data samples. By adaptively tuning the mixing parameter for each mixup pair, `AGMixup` enhances the model's interpolation capability and confidence.

Case Study: What is AGMixup doing?

In this section, we conduct a comprehensive empirical investigation into `AGMixup` on Cora, Pubmed, and Coauthor CS datasets. We benchmark `AGMixup` against various methodologies, including two variants of `AGMixup` itself (*R*-GMixup, which randomly samples λ from a Beta distribution, and *F*-GMixup, employing a fixed $\lambda = 0.5$ for all mixup pairs), two prevalent graph mixup methods (GraphMix and iGraphMix), and a baseline vanilla GCN model.

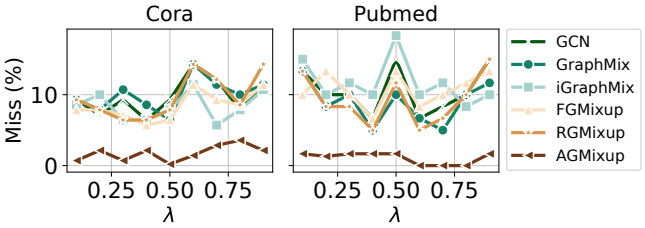

Figure 3: Prediction errors in-between training data. Our `AGMixup` significantly reduces the rate of prediction misses, indicating superior interpolation capability.

Improving Interpolation Capability. Mixup (Zhang et al. 2017) encourages models to behave linearly in-between training samples. In the context of the mixup method and its evaluation, a "miss" refers to an instance where the prediction for a mixed sample does not match either of the labels of the original samples. Fewer misses imply that the model is better at interpolating between classes, providing smooth transitions, and understanding the variations in data that might not be explicitly represented in the training set. In Fig. 3, we synthesize test subgraphs $\tilde{\mathcal{G}}_{ij}$ with various λ. While all the baselines exhibit higher misses around the $\lambda = 0.5$ point, our `AGMixup` shows a noticeable reduction in misses. For example, on the Cora dataset, `AGMixup` dramatically reduces the prediction misses to as low as 0.20%, a stark contrast to the other methods, where misses even exceed 10%. It demonstrates `AGMixup`'s superior ability to enhance the model's ability to interpolate between training examples, thereby improving its interpolation capability. By tuning the mixing parameter adaptively, our `AGMixup` tailors the generation of mixed samples in a way that aligns more closely with the inherent data distribution, effectively reducing the incidence of prediction misses.

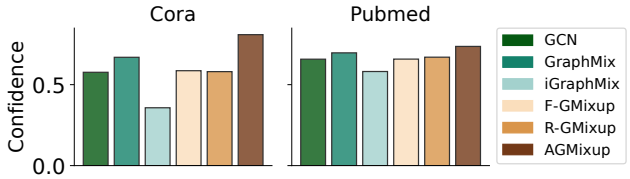

Figure 4: Confidence comparison. Our `AGMixup` makes the model more "confident" in the predictions by paying more attention to those underrepresented samples.

Improving Model Confidence. Graphs often contain underrepresented regions when labeled nodes distribute sparsely in the graph (Lu et al. 2024), which can lead to higher uncertainty in model predictions for those areas. Our AGMixup offers a dynamic balance between exploring new areas of the feature space (especially those that are underrepresented and uncertain) and exploiting the knowledge that the model has confidently acquired. As the model becomes more confident in certain regions (indicating low uncertainty), the adaptive mechanism adjusts to encourage less mixing for those samples, shifting towards exploitation of the learned representations. This approach encourages the model to pay more attention to underrepresented feature space regions. Empirical observations from Fig. 4 show that AGMixup significantly improves the confidence[1] of GCN for test node. For instance, on the Cora dataset, AGMixup's confidence score surpasses 0.8, markedly higher than any baselines, suggesting that it successfully directs the model's focus towards previously underexplored areas, thereby enhancing overall prediction certainty.

Complexity Analysis

AGMixup's complexity is driven by three operations: subgraph extraction, mixup operation, and adaptive λ calculation. Subgraph extraction can be done offline, minimizing training overhead. The mixup operation, which involves interpolating features and connections, is computationally light due to parallelizable matrix addition. We assume that each subgraph contains n nodes and has a hidden dimensionality of d. Therefore, adaptive λ calculation, with a complexity of $O(n + nd)$ for similarity and $O(n + nC)$ for uncertainty, remains manageable. To ensure scalability, we introduce a shrink ratio $\epsilon \in (0, 1]$ to control the number of subgraphs processed per iteration, keeping the overall complexity at $O(\epsilon|\mathcal{D}_L|(n + C + 2))$, where $\mathcal{D}_L$ is the labeled node set.

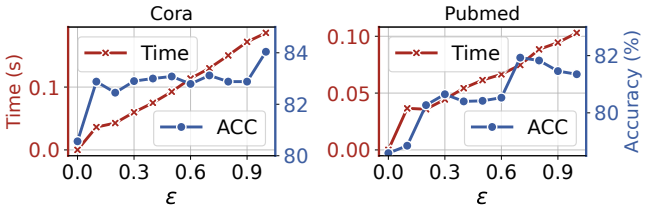

Figure 5: Efficacy analysis using GCN as backbone model.

In Fig. 5, we analyze the relationship between the time cost per epoch of mixup training and accuracy of AGMixup w.r.t different values of ϵ, ranging from 0 to 1 with an interval of 0.1. The results show a clear trend: as ϵ increases, computational time also rises, reflecting the additional overhead associated with using a larger proportion of data for mixup. This increase in computational cost is expected as more subgraphs are processed during training. On the other hand, model performance generally improves with higher ϵ values, peaking at certain points before plateauing or slightly declining. This suggests that while increasing ϵ enhances model generalization, there is an optimal range where performance gains are maximized without incurring excessive computational costs.

[1]The confidence is calculated by averaging the maximum prediction scores on nodes.

Experiment

We compare our AGMixup[2] with three state-of-the-art graph mixup methods targeting at the semi-supervised node classification problem: (1) GraphMix (Verma et al. 2021) that trains GNNs by interpolating nodes' hidden representations and corresponding labels; (2) NodeMixup (Lu et al. 2024) that chooses mixup pairs from labeled and unlabeled node sets; (3) iGraphMix (Jeong et al. 2024) that mixes the input features, labels, and connections of two labeled nodes. We choose four representative GNNs as our backbone models, i.e., GCN (Kipf and Welling 2017), GAT (Velickovic et al. 2018), JKNet (Xu et al. 2018), and GraphSAGE (Hamilton, Ying, and Leskovec 2017). As for the datasets, we choose seven graphs, i.e., Cora, Citeseer, Pubmed (Yang, Cohen, and Salakhudinov 2016), Coauthor CS, and Coauthor Physics (Shchur et al. 2018), and two large-scale graphs, i.e., ogbn-arxiv and ogbn-products (Hu et al. 2020).

Comparison

Classification Accuracy. The semi-supervised node classification accuracy results presented in Table 1 are obtained from ten different runs, ensuring reliable and consistent measurements. The results consistently demonstrate that AGMixup surpasses both the baseline models and other mixup methods in terms of classification accuracy across all datasets and models. Notably, AGMixup exhibits significant performance improvements, achieving the highest enhancements in accuracy across almost all tested GNNs. For instance, when applied to the GCN model on Cora, AGMixup achieves a remarkable peak accuracy of 83.61%, outperforming SOTA graph mixup methods by a significant margin. These notable improvements reveal the effectiveness of AGMixup in enhancing the model's ability to generalize from limited labeled data, as well as its versatility in being applied to various GNNs and datasets. The best accuracy is marked in **bold** according to each backbone model.

Generalization Gap. We explore the generalizability of our AGMixup by calculating the generalization gap by comparing the generalization gaps across different methods in Fig. 6. The generalization gap, defined as the difference between the test loss and training loss, serves as an indicator of a model's ability to generalize from training data to unseen test data. Lower values indicate better generalization, as the model is less likely to overfit the training data. The consistent outperformance of AGMixup across all datasets highlights its robustness and effectiveness in ensuring model generalization. The generalization advantage of AGMixup is due to its capacity to blend the strengths of mixup regularization with the specific challenges posed by graph data,

[2]https://github.com/WeigangLu/AGMixup

	Cora	Citeseer	Pubmed	CS	Physics	Arxiv	**Avg. Gains**
GCN	$80.51_{\pm 1.39}$	$69.53_{\pm 0.96}$	$78.23_{\pm 1.64}$	$91.22_{\pm 0.83}$	$93.51_{\pm 0.34}$	$65.39_{\pm 0.13}$	0.00% ↑
+ GraphMix	$81.18_{\pm 1.03}$	$69.67_{\pm 0.96}$	$78.07_{\pm 1.99}$	$91.27_{\pm 0.97}$	$93.54_{\pm 0.73}$	$65.27_{\pm 0.29}$	0.12% ↑
+ NodeMixup	$81.21_{\pm 1.02}$	$69.45_{\pm 0.74}$	$78.98_{\pm 1.32}$	$92.03_{\pm 0.36}$	$93.86_{\pm 0.67}$	$66.82_{\pm 0.24}$	0.86% ↑
+ iGraphMix	$81.16_{\pm 0.95}$	$69.95_{\pm 0.84}$	$77.07_{\pm 1.61}$	$90.93_{\pm 0.68}$	$93.61_{\pm 0.36}$	$65.97_{\pm 0.57}$	0.10% ↑
+ **AGMixup** (***ours***)	$\mathbf{83.61}_{\pm 0.39}$	$\mathbf{70.63}_{\pm 1.54}$	$\mathbf{81.90}_{\pm 0.40}$	$\mathbf{92.35}_{\pm 0.92}$	$\mathbf{94.45}_{\pm 0.42}$	$\mathbf{67.96}_{\pm 0.21}$	**2.71%** ↑
GAT	$80.02_{\pm 1.15}$	$68.38_{\pm 1.35}$	$77.28_{\pm 1.80}$	$90.06_{\pm 0.80}$	$92.59_{\pm 0.64}$	$66.57_{\pm 0.27}$	0.00% ↑
+ GraphMix	$80.07_{\pm 1.11}$	$69.53_{\pm 0.76}$	$76.96_{\pm 1.85}$	$88.43_{\pm 1.49}$	$91.73_{\pm 1.47}$	$65.84_{\pm 0.18}$	0.41% ↓
+ NodeMixup	$81.46_{\pm 1.13}$	$69.21_{\pm 0.86}$	$78.03_{\pm 1.03}$	$90.86_{\pm 1.23}$	$92.84_{\pm 1.42}$	$66.72_{\pm 0.26}$	0.89% ↑
+ iGraphMix	$80.07_{\pm 1.20}$	$69.11_{\pm 0.79}$	$77.14_{\pm 1.35}$	$90.01_{\pm 0.99}$	$92.64_{\pm 0.69}$	$64.79_{\pm 0.51}$	0.28% ↓
+ **AGMixup** (***ours***)	$\mathbf{83.14}_{\pm 0.54}$	$\mathbf{69.72}_{\pm 1.51}$	$\mathbf{81.60}_{\pm 0.51}$	$\mathbf{91.92}_{\pm 1.41}$	$\mathbf{94.31}_{\pm 0.63}$	$\mathbf{67.48}_{\pm 0.72}$	**2.78%** ↑
JKNet	$79.48_{\pm 1.47}$	$67.45_{\pm 1.05}$	$77.52_{\pm 2.29}$	$89.93_{\pm 1.11}$	$92.59_{\pm 1.36}$	$67.39_{\pm 0.18}$	0.00% ↑
+ GraphMix	$80.14_{\pm 1.29}$	$67.75_{\pm 1.01}$	$78.11_{\pm 2.12}$	$90.97_{\pm 0.85}$	$92.68_{\pm 1.09}$	$66.59_{\pm 0.13}$	0.35% ↑
+ NodeMixup	$80.94_{\pm 1.03}$	$\mathbf{68.08}_{\pm 1.41}$	$78.96_{\pm 1.61}$	$91.71_{\pm 0.98}$	$92.44_{\pm 1.24}$	$67.98_{\pm 0.42}$	1.22% ↑
+ iGraphMix	$79.94_{\pm 1.03}$	$67.58_{\pm 0.99}$	$76.03_{\pm 2.10}$	$91.19_{\pm 0.58}$	$92.83_{\pm 0.86}$	$65.15_{\pm 0.81}$	0.46% ↓
+ **AGMixup** (***ours***)	$\mathbf{82.06}_{\pm 0.83}$	$68.00_{\pm 1.53}$	$\mathbf{81.34}_{\pm 0.44}$	$\mathbf{92.63}_{\pm 0.50}$	$\mathbf{93.27}_{\pm 0.62}$	$\mathbf{69.76}_{\pm 0.21}$	**2.70%** ↑
GraphSAGE	$79.28_{\pm 1.20}$	$68.36_{\pm 0.89}$	$75.72_{\pm 1.84}$	$91.95_{\pm 0.36}$	$93.18_{\pm 0.47}$	$66.00_{\pm 0.17}$	0.00% ↑
+ GraphMix	$79.65_{\pm 1.44}$	$68.44_{\pm 1.21}$	$75.88_{\pm 1.89}$	$91.71_{\pm 0.52}$	$92.93_{\pm 0.59}$	$66.73_{\pm 0.14}$	0.22% ↑
+ NodeMixup	$80.42_{\pm 1.23}$	$69.06_{\pm 0.58}$	$77.92_{\pm 1.72}$	$91.94_{\pm 0.38}$	$93.78_{\pm 0.68}$	$67.13_{\pm 0.21}$	1.28% ↑
+ iGraphMix	$79.72_{\pm 1.14}$	$68.30_{\pm 0.82}$	$75.70_{\pm 1.67}$	$91.70_{\pm 0.61}$	$92.80_{\pm 0.77}$	$64.66_{\pm 0.47}$	0.37% ↓
+ **AGMixup** (***ours***)	$\mathbf{82.67}_{\pm 1.16}$	$\mathbf{70.04}_{\pm 1.08}$	$\mathbf{79.84}_{\pm 0.74}$	$\mathbf{92.31}_{\pm 0.50}$	$\mathbf{94.28}_{\pm 0.64}$	$\mathbf{68.82}_{\pm 0.11}$	**3.00%** ↑

Table 1: Semi-supervised Node Classification Results (%).

creating a robust method that effectively bridges the gap between training and test performance.

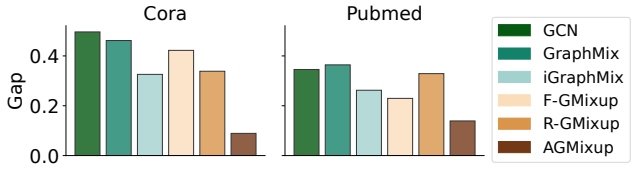

Figure 6: Generalization gap (|Test Loss - Train Loss |) comparison. AGMixup shows better generalization ability over vanilla GCN and other methods, keeping the model from being over-fitting.

Additional Results. Due to the page limit, we provide some results in our preprint version[3]: (1) a paired T-Test between AGMixup and other methods; (2) results on more advanced GNNs, i.e., GCNII (Chen et al. 2020), APPNP (Klicpera, Bojchevski, and Günnemann 2019), GPRGNN (Chien et al. 2021), and GRAND (Feng et al. 2020); (3) results on fewer labeled nodes; (4) results on ogbn-products; (5) performance comparison against other graph augmentation methods, i.e., DropEdge (Rong et al. 2019) and PairNorm (Zhao and Akoglu 2019).

Ablation Study

In this section, we ablate our AGMixup employing on GCN to investigate two important designs, i.e., subgraph-centric mixup and adaptive λ generation.

[3]https://arxiv.org/abs/2412.08144

Subgraph-Centric Mixup. In our comparative analysis illustrated in Fig. 7, we evaluate the performance of subgraph-centric against node-centric mixup across various graph datasets. The node-centric mixup, a variant of our AGMixup with $r = 0$, focuses solely on the central nodes. The results clearly demonstrate the advantages of the subgraph-centric approach, especially on the Pubmed dataset where it significantly outperforms the node-centric version. The subgraph-centric mixup's superiority can be attributed to its capacity to retain more complex structural information and contextual dependencies within graphs. Unlike the node-centric approach, which concentrates on individual node features and may result in considerable information loss, subgraph-centric mixup preserves the integrity of local graph structures. This method not only captures the broader graph topology but also enhances model robustness by incorporating a richer contextual understanding into the training process.

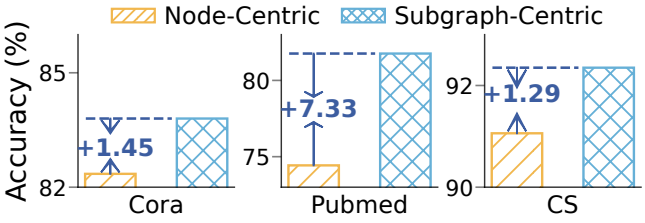

Figure 7: Ablation study of subgraph-centric mixup.

Adaptive λ Generation. Our analysis explores the effectiveness of adaptive λ in AGMixup by contrasting it with

R-GMixup, which randomly samples λ from a Beta distribution, and *F*-GMixup, which employs a fixed $\lambda = 0.5$. As illustrated in Fig. 8 (***upper***), AGMixup consistently outperforms these variants across all datasets. This superior performance underscores the significance of tailoring λ to the distinct characteristics of each mixup pair, thus optimizing the integration of subgraphs for enhanced GNNs' performance. The second part of the analysis in Fig. 8 (***bottom***), evaluate the contributions of the Contextual Similarity-aware λ Initialization (**S-Aware**) and the Uncertainty-aware λ Adjustment (**U-Aware**) modules. The removal of either module leads to a noticeable decrease in performance, with the impact being more severe in the absence of the S-Aware module. This is particularly evident in the Pubmed dataset, highlighting the S-Aware module's critical role in initializing λ in response to the contextual similarity between subgraphs. This analysis confirms the critical importance of both modules in our AGMixup.

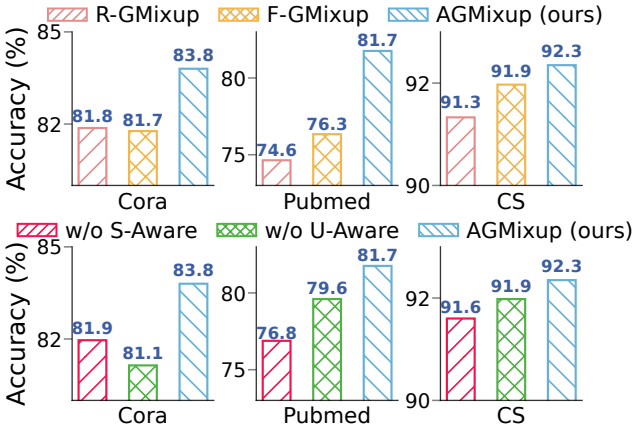

Figure 8: Ablation study of adaptive λ generation.

Hyperparameter Study

In this section, we explore the impact of key hyperparameters on the performance of AGMixup, i.e., subgraph size (controlled by r) and sensitivity to similarity and uncertainty (controlled by γ and β, respectively). We conduct experiments on Cora and Pubmed using GCN as the backbone model. More analysis can also be found in our preprint version.

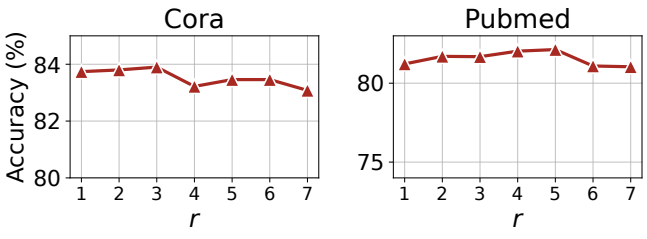

Figure 9: Hyperparameter study on r.

Subgraph Size. We investigate the effects of the subgraph size hyperparameter r on the performance of AGMixup, ranging from 1 to 7, as illustrated in Fig. 9. For the Cora dataset, AGMixup exhibits optimal performance at $r = 3$, achieving the highest accuracy of 83.90%, with performance declining as r increased. For the Pubmed dataset, AGMixup shows improved performance as r is increased up to a moderate level ($r = 5$), peaking at an accuracy of 82.13%. These trends suggest that while a larger r can beneficially expand the contextual scope of the node features, excessively broad neighborhoods may introduce noise and irrelevant information. *Based on these findings, to achieve an optimal balance between performance enhancement and computational efficiency, we recommend tuning r within the range of 2 to 5.*

Sensitivity to Similarity and Uncertainty. To evaluate the impact of γ and β, which adjust the sensitivity to contextual similarity and the influence of uncertainty differences in mixup pairs respectively, we conduct a comprehensive analysis in Fig. 10. The range of both hyperparameters is from 0 to 3 with an interval of 0.5. For both datasets, we observe that increasing β initially improves performance.

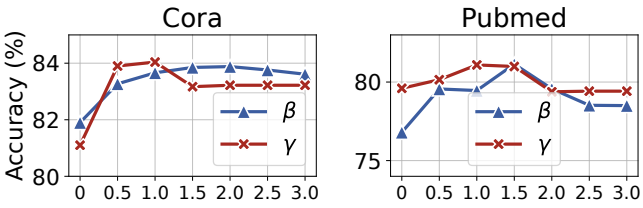

Figure 10: Hyperparameter study on γ and β.

However, this trend reverses beyond an optimal point, suggesting that excessive emphasis on uncertainty might cause the model to focus too much on uncertain data, integrating too much randomness into the training process. Similarly, γ also exhibits an initial increase in performance, reaching a peak before it begins to decline. This pattern indicates that while a moderate increase in γ can help the model avoid generating unrealistic mixed data for dissimilar subgraphs, an overly high γ potentially makes the mixup operation sensitive to dissimilarity. It limits the model's ability to generalize effectively to new, unseen data. *Based on these empirical observations, we recommend setting γ and β at the range of 0.5 to 2.* This range appears to strike an effective balance between enhancing model performance.

Conclusion and Future Work

Our AGMixup introduces significant advancements in applying adaptive mixup technique to the graph domain. Future work could focus on establishing a theoretical framework to understand how mixup methods affect GNNs and developing a generalized mixup method for different graph-based learning tasks, e.g., link prediction (Zhang and Chen 2018) and graph classification (Yang et al. 2022).

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grants 62133012, 61936006, 62425605, and 62303366, the Fundamental Research Funds for the Central Universities under Grants QTZX24072, and the Key Research and Development Program of Shaanxi under Grant 2024CY2-GJHX-15.

References

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

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

Crisostomi, D.; Antonelli, S.; Maiorca, V.; Moschella, L.; Marin, R.; and Rodolà, E. 2022a. Metric based few-shot graph classification. In *Learning on Graphs Conference*, 33–1. PMLR.

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

Dabouei, A.; Soleymani, S.; Taherkhani, F.; and Nasrabadi, N. M. 2021. Supermix: Supervising the mixing data augmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 13794–13803.

Faramarzi, M.; Amini, M.; Badrinaaraayanan, A.; Verma, V.; and Chandar, S. 2022. Patchup: A feature-space block-level regularization technique for convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 589–597.

Feng, W.; Zhang, J.; Dong, Y.; Han, Y.; Luan, H.; Xu, Q.; Yang, Q.; Kharlamov, E.; and Tang, J. 2020. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33: 22092–22103.

Guo, H.; and Mao, Y. 2021. ifMixup: Interpolating Graph Pair to Regularize Graph Classification. *arXiv e-prints*, arXiv–2110.

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

Han, X.; Jiang, Z.; Liu, N.; and Hu, X. 2022. G-mixup: Graph data augmentation for graph classification. In *International Conference on Machine Learning*, 8230–8248. PMLR.

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

Jeong, J.; Lee, H.; Yoon, H. G.; Lee, B.; Heo, J.; Kim, G.; and Seon, K. J. 2024. iGraphMix: Input Graph Mixup Method for Node Classification. In *The Twelfth International Conference on Learning Representations*.

Kim, J.-H.; Choo, W.; and Song, H. O. 2020. Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 5275–5285. PMLR.

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

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

Liu, Z.; Li, S.; Wang, G.; Wu, L.; Tan, C.; and Li, S. Z. 2024. Harnessing hard mixed samples with decoupled regularizer. *Advances in Neural Information Processing Systems*, 36.

Liu, Z.; Li, S.; Wu, D.; Liu, Z.; Chen, Z.; Wu, L.; and Li, S. Z. 2022. Automix: Unveiling the power of mixup for stronger classifiers. In *European Conference on Computer Vision*, 441–458. Springer.

Lu, W.; Guan, Z.; Zhao, W.; Yang, Y.; and Jin, L. 2024. Nodemixup: Tackling under-reaching for graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 14175–14183.

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

Ma, X.; Chu, X.; Wang, Y.; Lin, Y.; Zhao, J.; Ma, L.; and Zhu, W. 2024. Fused Gromov-Wasserstein Graph Mixup for Graph-level Classifications. *Advances in Neural Information Processing Systems*, 36.

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

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

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

Rong, Y.; Huang, W.; Xu, T.; and Huang, J. 2019. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*.

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

Uddin, A. S.; Monira, M. S.; Shin, W.; Chung, T. C.; and Bae, S. H. 2021. SALIENCYMIX: A SALIENCY GUIDED DATA AUGMENTATION STRATEGY FOR BETTER REGULARIZATION. In *9th International Conference on Learning Representations, ICLR 2021*.

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

Verma, V.; Lamb, A.; Beckham, C.; Najafi, A.; Mitliagkas, I.; Lopez-Paz, D.; and Bengio, Y. 2019. Manifold mixup:

Better representations by interpolating hidden states. In *International conference on machine learning*, 6438–6447. PMLR.

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

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

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

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

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

Yang, Y.; Guan, Z.; Zhao, W.; Lu, W.; and Zong, B. 2022. Graph substructure assembling network with soft sequence and context attention. *IEEE Transactions on Knowledge and Data Engineering*, 35(5): 4894–4907.

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

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

Zhang, L.; Deng, Z.; Kawaguchi, K.; Ghorbani, A.; and Zou, J. 2021. HOW DOES MIXUP HELP WITH ROBUSTNESS AND GENERALIZATION? In *9th International Conference on Learning Representations, ICLR 2021*.

Zhang, L.; Deng, Z.; Kawaguchi, K.; and Zou, J. 2022. When and how mixup improves calibration. In *International Conference on Machine Learning*, 26135–26160. PMLR.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Zhao, L.; and Akoglu, L. 2019. Pairnorm: Tackling oversmoothing in gnns. *arXiv preprint arXiv:1909.12223*.