Less is More: Using Buffer Nodes to Reduce Excessive Majority Node Influence in Class Imbalance Graphs

Qian Wang
National University of Singapore

Zemin Liu
Zhejiang University

Zhen Zhang
Nanjing University

Bingqiao Luo
National University of Singapore

Bingsheng He
National University of Singapore

Abstract

Graph Neural Networks (GNNs), despite success in node classification, struggle with class-imbalanced graphs, leading to minority node misclassification. Existing methods that synthesize minority nodes often overlook how majority nodes propagate misleading information through majority-minority edges; our analysis confirms this negative impact. To address this, we propose BufferGraph, a framework that inserts buffer nodes on such edges. These nodes act as controlled bottlenecks to reduce excessive majority node influence. And we theoretically demonstrate they reduce minority node feature distortion. Experiments on five real-world datasets show BufferGraph improves accuracy by up to 2% over state-of-the-art methods, excelling in imbalanced settings and for minority classes with high heterophily. Code is available at https://github.com/Persdre/BufferGraph.

1 Introduction

Recent years have witnessed the rapid development of graph representations, especially GNNs [1–6]. Among various graph tasks, node classification has proven important and socially beneficial in real-world applications, such as identifying influencers on social networks [7–9] and detecting fraudsters in financial activities [10–12]. Naturally, the misclassification of minority classes can have harmful societal impacts. For example, minor fraudulent accounts that are not identified will continue to affect the reputation of e-commerce platforms and lead to loss of platform users [13, 14]. Despite the critical importance of accurate minority class prediction in real-world applications, current GNN approaches struggle with class-imbalanced graphs, creating a significant gap between model capabilities and practical needs.

Standard GNN models assume class balance and neighborhood homogeneity when aggregating features [1–3]. However, real-world graphs typically exhibit significant class imbalances—fraudulent accounts form a tiny minority in the Ethereum network [12, 15], and computer vision papers greatly outnumber computer architecture papers in citation networks [16]. These imbalances bias model performance toward majority classes [4, 17, 18], resulting in poor minority class prediction when GNNs are applied directly to imbalanced graphs.

To address this problem, previous methods have synthesized fake minority nodes and connected them to the original graph [19–24]. These approaches include techniques like GraphSMOTE [19], GraphENS [20], TAM [21], and GraphSHA [22], which generate synthetic minority nodes through various interpolation and mixing strategies. However, these approaches overlook a critical issue: **majority nodes continue to propagate information to minority nodes through majority-minority edges**, which we define as *misleading edges*. These edges distort minority node representations during message passing. Despite generating new nodes to balance classes globally, local neighborhoods often remain imbalanced, with minority nodes surrounded by majority nodes. As shown in Figure 1,

Wang et al., Less is More: Using Buffer Nodes to Reduce Excessive Majority Node Influence in Class Imbalance Graphs. *Proceedings of the Fourth Learning on Graphs Conference (LoG 2025)*, PMLR 269, Hybrid Event, December 10–12, 2025.

higher percentages of majority neighbors correlate with lower prediction accuracy for minority classes, which restricts the performance of baselines.

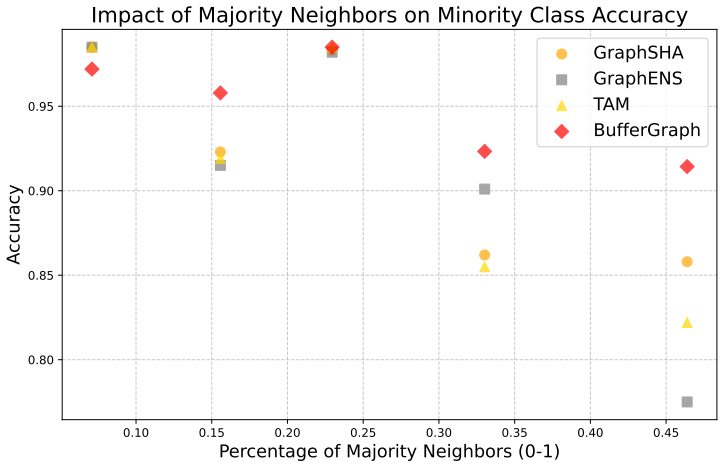

Figure 1: Impact of majority-class neighborhood composition on minority class accuracy on the Amazon-Computers dataset. Each point represents a minority class, plotting its classification accuracy against the percentage of its majority-class neighbors. While baseline methods struggle with accuracy degradation as majority-class neighbor percentage increases, **our proposed BufferGraph maintains consistently superior performance across all neighborhood compositions, effectively addressing the challenge of majority-class interference in graph-based classification.**

To address this fundamental limitation, we propose BufferGraph—a framework that explicitly regulates cross-class edge message propagation through adaptive edge intervention.

Our approach enhances minority class classification by decreasing message propagation along *misleading edges*—connections often dominated by majority nodes, which can distort the inherent features of minority instances as shown in Figure 2.

Figure 2: Misleading edge example.

To achieve this goal, we first use predicted labels from the GNN pre-training stage to identify potential *misleading edges*. Edges connecting to predicted minority nodes are classified as non-problematic as their predictions have shown high precision [22]. For other potentially *misleading edges*, we insert buffer nodes to modulate message passing, allowing information to flow through both the original path and a new path via the buffer node. These unlabeled buffer nodes act as message passing modulators. The proportion of message passing through the original edge is determined by the difference in predicted labels of connected nodes, helping maintain prediction accuracy for majority-majority node pairs while reducing majority nodes' misleading influences on minority nodes. We provide the overview of our method in Figure 3.

We evaluate BufferGraph on five real-world datasets: Amazon-Photos, Amazon-Computers [25], Coauthor-CS, Coauthor-Physics [16], and WikiCS [26]. Experiments are conducted under both natural and artificially imbalanced settings to assess BufferGraph's effectiveness across varying imbalance ratios, comparing against state-of-the-art baselines.

Experiment results demonstrate that BufferGraph consistently outperforms the baseline models [19–22] in most configurations and datasets. For instance, under the random splitting scenario, BufferGraph exhibits its superiority by achieving a 2% increase in accuracy (from 88.39% to 90.22%), a 2% enhancement in balanced accuracy (from 89.23% to 91.85%), and a 2% boost in F1-score (from 87.55% to 89.30%) on the Amazon-Computers dataset compared to the second-best outcomes.

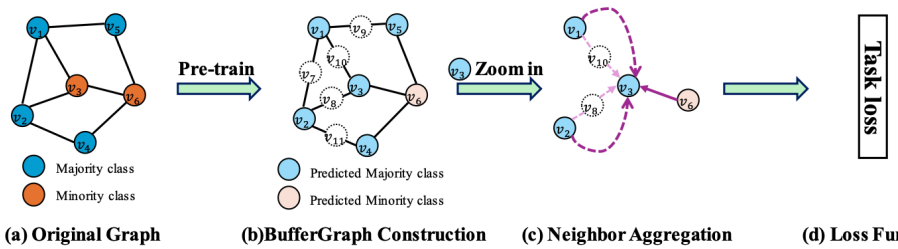

Figure 3: BufferGraph overview where v_1, v_2, v_4, v_5 are of the major class and v_3, v_6 are of the minor class. The input graph is shown in **(a)**. After the pre-training using the GCN model, the nodes are predicted into majority or minority classes. Subsequently, we introduce a buffer node into each edge except those minority nodes' neighboring edges, as depicted in **(b)**. The feature of the buffer node is a mixup of the features from the two nodes connected by the edge. Then, we zoom in v_3 to show the neighbor aggregation of BufferGraph in **(c)**. For edges with buffer nodes, messages pass both through the buffer node and directly to v_3, with direct edge weights determined by the predicted label differences between nodes. Loss calculation during BufferGraph neighbor aggregation is shown in **(d)**.

Conversely, within the imbalanced splitting framework, BufferGraph continues to excel, marking a 2% improvement in accuracy, a 1% gain in balanced accuracy, and a 1.5% increase in F1-score on the Amazon-Computers dataset relative to the runner-up results. These findings underscore BufferGraph's adaptability and efficacy in addressing class imbalances. We make the following contributions:

- ❶ **Heterophily-Aware Buffer Framework**: We identify a critical issue overlooked by previous work - the negative impact of majority-minority edges on message passing in imbalanced graphs. To address this, we propose BufferGraph, which strategically introduces buffer nodes to regulate information flow through misleading connections.
- ❷ **Theoretical Justification**: We provide an intuitive theoretical justification grounded in the GNN message-passing mechanism. We demonstrate how inserting a buffer node transforms a strong, direct 1-hop influence from a majority neighbor into a dampened 2-hop path, thereby reducing feature distortion and preserving the minority node's original information.
- ❸ **Experimental Validation**: We conduct comprehensive experiments on multiple real-world datasets. Results demonstrate BufferGraph's consistent superiority over state-of-the-art methods, particularly in improving performance on minority classes across varying imbalance ratios. We also conduct comprehensive ablation studies on BufferGraph to assure each component is necessary.

2 Proposed Method

Figure 3 illustrates our approach, which consists of three key components: (1) pre-training a GCN to identify potential misleading edges, (2) inserting buffer nodes into these edges to create controlled information bottlenecks, and (3) implementing an adaptive message-passing mechanism that adjusts information flow based on node similarity.

2.1 Background

Notations. We address the challenge of class-imbalanced node classification on an unweighted, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \cdots, v_N\}$ represents the set of N nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the edges. The graph structure is captured by an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$, with $\boldsymbol{A}_{ij} = 1$ indicating an edge between nodes v_i and v_j. Node features are represented by a matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, where each row $\boldsymbol{X}_i \in \mathbb{R}^d$ corresponds to the d-dimensional features of node v_i. Each node v is labeled with one of C classes, $\boldsymbol{Y}(v) \in \{1, \cdots, C\}$, with $\boldsymbol{Y}_c$ encompassing all nodes in class c. The training subset $\mathcal{V}^L \subset \mathcal{V}$, particularly in imbalanced settings, is characterized by class size disparities, quantified by the imbalance ratio. We provide notations used in Table 7.

Definition. *Majority-minority edges*, also referred to as *misleading edges*, are defined as edges connecting nodes from a majority class to nodes from a minority class. In the context of multi-

class graphs, we define **majority and minority groups** (as detailed in Section 3.1), where multiple classes can belong to each group. Formally, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, let $\mathcal{V}_{maj}$ be the set of nodes belonging to any majority class and $\mathcal{V}_{min}$ be the set of nodes belonging to any minority class. An edge $e = (v_i, v_j) \in \mathcal{E}$ is a majority-minority edge if ($v_i \in \mathcal{V}_{maj}$ and $v_j \in \mathcal{V}_{min}$) or ($v_i \in \mathcal{V}_{min}$ and $v_j \in \mathcal{V}_{maj}$).

2.2 Buffer Node Generation

Buffer nodes represent our key innovation for addressing class imbalance in GNNs by modulating message passing between majority and minority classes through strategically introduced buffer nodes, rather than generating new minority nodes.

A buffer node, denoted as v_{buf}, does not possess a label but is characterized by a feature vector $\boldsymbol{X}_{\text{buf}}$. This vector is derived by interpolating the features of two adjacent nodes, v_{a} and v_{b}, connected by an undirected edge. The interpolation is governed by the equation [27]:

$$\boldsymbol{X}_{\text{buf}} = \alpha \boldsymbol{X}_{\text{a}} + (1 - \alpha)\boldsymbol{X}_{\text{b}}, \quad \alpha \in [0, 1]. \tag{1}$$

In this formula, α acts as the mixup coefficient, influencing the degree to which the features of node v_{a} and node v_{b} affect the buffer node's features. A lower value of α biases the feature vector $\boldsymbol{X}_{\text{buf}}$ towards the features of v_{b}, whereas a higher value biases it towards those of v_{a}.

The buffer nodes integrated into edges serve primarily to regulate message passing rather than to enhance feature representation. Given this focused role, we initially set $\alpha = 1/2$ uniformly across all edges, prioritizing methodological consistency in our preliminary investigation. A comprehensive analysis of α's impact through systematic parameter variation is presented in Section 3.5.

2.3 BufferGraph Framework

To address the challenges of message passing across majority-minority edges within class-imbalanced graphs, BufferGraph implements a dynamic message passing mechanism that precisely modulates the flow of information. We illustrate the details in Figure 3 as follows:

Pre-training Stage. We first pre-train a GCN model using training sets label information to predict the labels of all nodes in the original graph. These predicted labels serve as a guide for identifying potential problematic edges, which are then targeted for buffer node insertion. According to our algorithm, edges connected to predicted minority nodes are considered non-problematic and left unchanged, as previous work [22] has shown high precision in minority node predictions.

Difference Score. After pre-training, we compute a difference score $s_{(v_i,v_j)}$ for each edge $e_{(v_i,v_j)}$, based on the Manhattan distance between the predicted label distributions of the connected nodes:

$$s_{(v_i,v_j)} = \sum_k |\hat{y}_{i,k} - \hat{y}_{j,k}| \tag{2}$$

where $\hat{y}_{i,k}$ and $\hat{y}_{j,k}$ are the predicted label distributions for nodes v_i and v_j respectively. This score quantifies the dissimilarity between nodes' predicted class distributions.

Adaptive Message Passing. For edges in the original graph, we modulate the message passing by adjusting the edge weight $w_{(v_i,v_j)}$ according to:

$$w_{(v_i,v_j)} = 1 - s_{(v_i,v_j)} \tag{3}$$

For edges connected to buffer nodes, we maintain their weights at 1. This design ensures that when there is a large label distribution difference (high $s_{(v_i,v_j)}$) between connected nodes, more information flows through the buffer node path rather than the direct edge. This mechanism effectively reduces the direct influence of majority nodes on their minority neighbors while preserving necessary information exchange through the buffer nodes. For instance, on an edge connecting two majority-class nodes, their predicted label distributions are often similar, resulting in a low difference score $s_{(v_i,v_j)}$ and a direct edge weight $w_{(v_i,v_j)}$ close to 1. This design preserves strong homophilic connections while primarily targeting the misleading influence on heterophilic majority-minority edges.

Neighbor Aggregation. The GNN neighbor aggregation mechanism we utilize follows the standard form [22]:

$$\boldsymbol{H}_t^{(l)} \leftarrow \mathbf{Transform}\left(\underset{\forall v_s \in \mathcal{N}_t}{\mathbf{Propagate}}\left(\boldsymbol{H}_s^{(l-1)}; \boldsymbol{H}_t^{(l-1)}\right)\right), \tag{4}$$

where $\boldsymbol{H}_t^{(l)}$ represents node embedding of node v_t in the l-th layer.

A GCN can be formalized with the following operations [1]:

$$\textbf{Propagate:} \quad M_t^{(l)} = \hat{A} H_t^{(l-1)}, \tag{5}$$

$$\textbf{Transform:} \quad H_t^{(l)} = \text{ReLU}(M_t^{(l)} W_t^{(l)}), \tag{6}$$

where $W^{(l)}$ is the learnable weight matrix of the l-th layer, and $\text{ReLU}(\cdot)$ is the activation function.

Training Objective. During training, we calculate the validation loss every epoch and re-evaluate the difference scores if the current validation loss improves over the best previous loss. Our loss function combines two components:

$$\boldsymbol{\mathcal{L}}_{\text{total}} = \boldsymbol{\mathcal{L}}_{\text{pred}} + \lambda \cdot \boldsymbol{\mathcal{L}}_{\text{diff}}. \tag{7}$$

where λ is a hyperparameter that balances the prediction loss and difference loss. While λ can theoretically take any positive real value, we primarily consider $\lambda \in [0.1, 1.0]$ to maintain a balanced contribution between the two loss terms. We also explore larger values of λ in our parameter sensitivity analysis (Section 3.5).

In this equation, $\mathcal{L}_{\text{pred}}$ is the standard cross-entropy loss for node classification. The term $\mathcal{L}_{\text{diff}}$, which we refer to as the "difference loss", is a regularization component designed to improve the discriminative power of the learned node representations. This loss is computed over all edges connecting pairs of nodes in the training set. For each such edge (v_i, v_j), we first calculate the Manhattan distance between their output distributions (as defined in Equation 2), which serves as a dissimilarity score. We then frame a binary classification task where the goal is to predict whether the two nodes share the same ground-truth label based on this score. The $\mathcal{L}_{\text{diff}}$ is the Binary Cross-Entropy loss for this task, which penalizes the model for producing large distances between same-class nodes and small distances between different-class nodes. This encourages the formation of well-separated class clusters in the embedding space.

Algorithm 1 details the complete BufferGraph implementation. And complexity analysis of BufferGraph can be found in Appendix D.

2.4 Theoretical Justification

Here, we provide an intuitive justification for our framework, grounded in the GNN message-passing mechanism, to explain how BufferGraph mitigates the negative influence of misleading edges and preserves minority node features.

The Core Problem. In a standard GNN, a majority-class neighbor (v_{maj}) has a strong, direct 1-hop influence on a minority-class neighbor (v_{min}). During message passing, the features of v_{maj} are directly aggregated by v_{min}, which can overwhelm or "overwrite" the minority node's original, distinct features. This is the essence of a misleading edge.

The BufferGraph Mechanism. Our method directly targets this issue by altering the graph topology. By inserting a buffer node (v_{buf}) on the edge between v_{maj} and v_{min}, we transform the nature of their connection. The original, single 1-hop path is now complemented by a parallel 2-hop path: $v_{maj} \rightarrow v_{buf} \rightarrow v_{min}$.

The Dampening Effect. A message received over a 2-hop path is inherently weaker and more diffuse than a 1-hop message. This is because:

- At the first step, the features of v_{maj} are passed to v_{buf} and are averaged with the features of v_{min} to form the buffer node's representation.
- At the second step, v_{min} aggregates from this already-averaged and smoothed representation at v_{buf}.

This "average of an average" process acts as a natural **dampening mechanism**. The strong, direct, and potentially misleading signal from the majority neighbor is effectively smoothed and weakened before it reaches the minority node.

The Consequence: Feature Preservation. By routing a portion of the influence through this dampened 2-hop channel (and adaptively down-weighting the direct 1-hop path), we reduce the direct, distorting impact of majority neighbors. This helps to **preserve the original features of the minority**

node during the GNN's propagation process, preventing them from being "washed out" and leading to more robust and accurate classification of minority nodes.

3 Experiments

This section focuses on answering the following research questions (RQs): **(RQ1)** How does BufferGraph's performance in node classification on naturally class-imbalanced graphs compare to that of existing baseline models (§3.2)? **(RQ2)** How effectively does BufferGraph outperform other baseline models in node classification across graphs with varying class-imbalance ratios (§3.3)? **(RQ3)** How does each component contribute to the overall performance of BufferGraph (§3.4)? **(RQ4)** How does BufferGraph's performance change when different hyperparameter settings are used (§3.5)? **(RQ5)** Does our inserting ratio affect the performance of BufferGraph (§3.6)?

3.1 Experimental Setup

Datasets. To comprehensively evaluate BufferGraph, we conduct experiments across five naturally class-imbalanced datasets: Amazon-Photos, Amazon-Computers [25], Coauthor-CS, Coauthor-Physics [16], and WikiCS [26]. Table 5 summarizes the key statistics of these datasets. Table 5 summarizes the key statistics of these datasets. A detailed breakdown of the per-class node counts for all datasets, as promised during our review, is provided in Appendix 6. The Max/Min ratio represents the number of samples in the largest majority class to that in the smallest minority class, highlighting the natural class imbalance in these datasets.

Majority/Minority Group Definition. To facilitate our analysis of neighborhood effects in a multi-class context (e.g., in Figure 1), we define "majority" and "minority" class groups. For each dataset, all classes are ranked in descending order based on the number of nodes in the training set. The top 50% of classes in this ranking are designated as the **majority group**, while the bottom 50% are designated as the **minority group**. For instance, in the 10-class Amazon-Computers dataset, the five largest classes constitute the majority group, and the five smallest form the minority group.

Baselines. We compare BufferGraph against three categories of GCN-backboned baselines: (1) **Loss management strategies** including Reweight [28], PC SoftMax [29], Cross Entropy, and Balanced SoftMax [30]; (2) **Class-imbalanced node classification** methods including GraphSMOTE [19], GraphENS [20], TAM [21], and GraphSHA [22]; (3) **Heterophilic GCN models** including MixHop [31] and H2GCN [32], which handle heterophily similar to majority-minority edges. More details can be found in Appendix F.

Evaluation Metrics. We use four widely adopted metrics for class-imbalanced node classification: **Accuracy (Acc.)** measuring overall classification accuracy across all nodes; **Balanced Accuracy (BAcc.)** as the average of per-class accuracy giving equal weight to each class; **Macro F1 Score (F1)** balancing precision and recall across all classes; and **Standard Deviation** reported as mean±std across five runs with different random seeds.

Implementation Details. For all experiments, we use the following settings: (1) **Random Splitting** with dataset division into training/validation/testing sets (6:2:2 ratio); (2) **Model Architecture** featuring three hidden layers (256 hidden dimensions each); (3) **Training Parameters** including learning rate 0.01, dropout rate 0.4, and up to 5000 epochs with early stopping; (4) **BufferGraph-Specific Parameters** with mixup coefficient $\alpha = 0.5$ and difference loss weight $\lambda = 1.0$. For the imbalanced setting, we follow [20] by downsampling the last half of classes in the training set to achieve an imbalance ratio of 10, while maintaining the original distribution in validation and testing sets. For hardware, we provide the information in Appendix E.

3.2 RQ1: Performance on Naturally Imbalanced Graphs

Random Splitting Results. Tables 1 and 2 demonstrate BufferGraph's consistent superiority over baselines across five datasets. On Amazon-Computers, BufferGraph achieves 90.22% accuracy (+2% vs 88.39%), 91.85% balanced accuracy (+2% vs 89.23%), and 89.30% F1-score (+2% vs 87.55%). The framework shows particular strength on minority classes (evidenced by high balanced accuracy) while maintaining majority class performance, with consistent improvements across datasets of varying characteristics, confirming its robustness. To further validate BufferGraph's effectiveness, we present a comparative analysis in Figure 1. **The results reveal that BufferGraph significantly**

improves the performance of minority classes, particularly those with a high percentage of majority-class neighbors, while maintaining strong performance across all classes. This balanced improvement demonstrates BufferGraph's ability to effectively address the challenges of imbalanced node classification.

Imbalanced Splitting Results. Tables 3 and 4 show BufferGraph's performance under artificially imbalanced settings with an imbalance ratio of 10. BufferGraph maintains its superior performance, achieving approximately 2% improvement in accuracy, 1% in balanced accuracy, and 1.5% in F1-score on Amazon-Computers compared to the runner-up results. While baseline methods like Reweight show good balanced accuracy but lower overall accuracy (indicating they improve minority class performance at the cost of majority class accuracy), BufferGraph achieves strong performance across all metrics, demonstrating its ability to effectively balance minority and majority class prediction.

Analysis. BufferGraph's superior performance stems from two key innovations: First, buffer nodes transform direct one-hop neighbors into two-hop neighbors, reducing the negative influence of majority nodes on minority nodes. Second, adaptive message passing modulates information flow based on node similarity, preserving minority class features while maintaining majority class performance.

Table 1: Random splitting experiment results of BufferGraph and other baselines on Amazon-Photos, Amazon-Computers, and Coauthor-CS. We report all metrics with the standard deviation errors for five repetitions. The best result is highlighted by bold text. The runner-up result is highlighted by the underline.

	Dataset	Amazon-Photos			Amazon-Computers			Coauthor-CS		
	Random Splitting	Acc.	BAcc.	F1	Acc.	BAcc.	F1	Acc.	BAcc.	F1
Methods	Vanilla	92.44±0.16	90.41±0.62	91.20±0.27	87.71±0.36	82.34±1.36	83.03±1.86	92.89±0.41	89.97±0.45	90.70±0.63
	Reweight	92.91±0.36	92.51±0.41	91.72±0.25	86.21±0.71	<u>89.23</u>±0.19	85.09±0.68	92.86±0.03	90.86±0.13	91.09±0.04
	PC Softmax	91.83±0.33	91.83±0.34	90.31±0.48	87.13±1.49	87.60±0.67	85.09±1.71	92.95±0.12	91.87±0.11	91.21±0.11
	Balanced Softmax	93.46±0.05	91.19±0.01	92.05±0.05	86.28±0.02	87.42±0.29	83.91±0.02	93.46±0.05	91.19±0.01	92.05±0.05
	Cross Entropy	93.40±0.20	92.13±0.21	92.30±0.23	86.99±0.22	82.26±0.75	84.21±0.50	93.05±0.14	90.85±0.37	91.49±0.32
	MixHop	92.85±0.42	91.77±1.36	91.20±0.47	85.02±0.26	75.17±1.58	75.09±1.13	91.97±0.58	88.50±0.49	89.41±0.43
	H2GCN	93.06±0.53	92.47±0.27	91.79±0.21	<u>88.39</u>±0.55	88.01±0.32	<u>87.55</u>±1.18	<u>94.24</u>±0.14	<u>92.77</u>±0.34	<u>92.75</u>±0.60
	GraphSmote	89.72±0.45	90.69±0.57	88.90±0.48	85.36±0.72	84.79±1.22	85.22±0.98	87.44±0.24	85.08±0.63	84.26±0.52
	GraphENS	93.37±0.42	92.18±0.36	91.63±0.46	86.35±0.71	87.66±0.54	85.81±0.47	91.65±0.23	90.72±0.39	89.53±0.36
	TAM	90.13±0.33	90.98±0.36	89.15±0.49	85.46±0.11	88.51±0.67	84.52±0.26	92.41±0.04	90.84±0.01	91.35±0.02
	GraphSHA	<u>93.63</u>±0.23	<u>92.61</u>±0.66	<u>92.60</u>±0.38	82.98±0.17	77.73±1.90	79.10±2.22	92.68±0.59	91.00±0.37	90.94±0.51
	BufferGraph	**94.47**±0.10	**94.28**±0.10	**93.12**±0.07	**90.22**±0.48	**91.85**±0.34	**89.30**±0.69	**94.90**±0.28	**93.88**±0.39	**93.70**±0.41

Table 2: Random splitting experiment results of BufferGraph on Coauthor-Physics and WikiCS.

	Dataset	Coauthor-Physics			WikiCS		
	Random Splitting	Acc.	BAcc.	F1	Acc.	BAcc.	F1
Methods	Vanilla	96.22±0.24	94.60±0.42	93.49±0.03	83.20±0.23	80.34±0.41	80.63±0.07
	Reweight	95.70±0.02	95.06±0.05	94.52±0.01	82.66±0.17	<u>82.82</u>±0.09	80.92±0.20
	PC Softmax	96.14±0.07	95.36±0.11	95.07±0.11	82.76±0.32	81.94±0.50	80.46±0.45
	Balanced Softmax	96.16±0.03	<u>95.52</u>±0.06	95.05±0.02	<u>83.83</u>±0.49	82.15±0.51	<u>81.74</u>±0.72
	Cross Entropy	<u>96.50</u>±0.14	95.30±0.13	<u>95.25</u>±0.18	83.15±0.23	82.63±0.74	81.57±0.30
	MixHop	94.77±0.69	92.32±0.86	93.15±1.09	77.07±1.11	67.96±2.02	68.57±1.95
	H2GCN	95.17±0.33	92.33±0.25	93.25±0.81	79.77±2.23	69.69±2.62	72.90±2.67
	GraphSmote	95.09±0.53	93.01±0.66	93.42±0.76	82.94±0.70	80.42±1.14	80.65±0.75
	GraphENS	95.46±0.09	95.32±0.04	94.27±0.06	81.78±0.06	80.87±0.12	79.80±0.08
	TAM	95.35±0.19	95.04±0.08	94.04±0.20	80.73±0.34	79.02±0.21	79.02±0.16
	GraphSHA	96.27±0.14	95.51±0.17	95.05±0.11	81.83±1.22	80.76±0.54	78.95±1.53
	BufferGraph	**96.78**±0.07	**96.18**±0.07	**95.34**±0.06	**84.47**±0.22	**84.34**±0.21	**81.92**±0.08

3.3 RQ2: Performance Across Varying Imbalance Ratios

Experimental Setup. To evaluate BufferGraph's robustness to different levels of class imbalance, we artificially adjust the imbalance ratio to 15, 20, and 25 on the Amazon-Computers dataset.

Results and Analysis. Figure 4 shows that BufferGraph maintains consistently superior balanced accuracy across all imbalance ratios, outperforming the second-best baseline by at least 3% in each

Table 3: Imbalanced splitting experiment results of BufferGraph and other baselines on Amazon-Photos, Amazon-Computers, and Coauthor-CS.

	Dataset	Amazon-Photos			Amazon-Computers			Coauthor-CS		
	ρ=10	Acc.	BAcc.	F1	Acc.	BAcc.	F1	Acc.	BAcc.	F1
Methods	Vanilla	92.20±0.54	89.60±0.05	90.41±0.14	83.40±0.29	69.71±0.28	70.79±0.43	92.54±0.55	89.86±0.68	90.53±0.63
	Reweight	92.65±0.36	92.34±0.17	90.79±0.36	86.46±0.20	89.26±0.08	85.33±0.14	93.23±0.12	91.74±0.07	91.86±0.03
	PC Softmax	84.51±0.86	88.69±1.27	84.01±2.57	70.48±1.09	84.92±1.21	70.50±0.46	92.78±0.02	93.16±0.06	91.23±0.14
	Balanced Softmax	92.81±0.20	93.33±0.04	91.44±0.19	87.55±0.24	89.31±0.16	86.95±0.02	93.99±0.01	93.24±0.03	92.35±0.02
	Cross Entropy	91.67±0.16	87.85±0.40	89.93±0.35	87.46±0.18	83.49±0.53	85.19±0.53	94.04±0.07	92.03±0.03	92.38±0.04
	MixHop	91.57±0.84	90.46±0.54	89.53±0.72	84.56±1.15	75.48±2.35	75.30±2.52	88.65±0.51	80.59±1.24	83.19±1.03
	H2GCN	92.49±0.70	91.04±0.29	91.55±0.24	85.22±0.46	85.19±0.30	85.55±0.83	93.98±0.36	92.61±0.36	92.33±0.34
	GraphSmote	88.31±0.63	88.15±1.53	87.27±0.28	85.30±0.66	84.66±0.27	84.35±0.23	88.95±0.19	83.96±0.99	85.56±0.78
	GraphENS	92.55±0.07	91.66±0.37	91.07±0.02	85.50±0.58	89.21±0.14	85.05±0.69	92.12±0.03	90.49±0.01	89.21±0.16
	TAM	91.08±0.03	91.70±0.07	90.15±0.07	85.79±0.18	88.21±0.69	85.21±0.42	92.53±0.04	90.45±0.13	90.67±0.13
	GraphSHA	93.56±0.04	92.46±0.30	92.59±0.02	85.24±0.52	83.77±0.55	83.31±0.59	92.42±0.16	90.43±0.46	90.21±0.22
	BufferGraph	**93.91**±0.18	**93.40**±0.20	**92.90**±0.07	**89.51**±0.35	**90.54**±0.57	**88.14**±0.40	**94.06**±0.02	**93.78**±0.03	**92.72**±0.35

Table 4: Imbalanced splitting experiment results of our model BufferGraph and other baselines on two class-imbalanced node classification benchmark datasets.

	Dataset	Coauthor-Physics			WikiCS		
	ρ=10	Acc.	BAcc.	F1	Acc.	BAcc.	F1
Methods	Vanilla	95.65±0.04	93.76±0.12	94.19±0.17	81.30±1.00	75.16±1.53	77.42±1.55
	Reweight	96.35±0.04	95.12±0.20	95.16±0.09	81.16±0.13	81.48±0.28	79.54±0.34
	PC Softmax	95.18±0.09	95.47±0.08	93.87±0.13	76.01±2.24	80.30±1.42	73.85±2.13
	Balanced Softmax	**96.46**±0.05	95.46±0.03	95.32±0.04	82.14±0.04	82.45±0.21	80.10±0.08
	Cross Entropy	96.12±0.01	94.53±0.11	94.93±0.03	82.44±0.23	78.07±0.51	80.06±0.10
	MixHop	93.88±0.32	91.06±0.82	91.98±0.27	76.91±1.41	66.24±0.49	66.90±0.94
	H2GCN	93.09±0.17	88.60±0.14	90.02±0.56	77.85±1.83	66.30±2.75	68.01±0.93
	GraphSmote	92.64±0.36	92.79±0.11	94.42±0.53	74.96±1.07	69.43±2.17	70.82±1.93
	GraphENS	95.35±0.19	95.04±0.08	94.04±0.20	80.73±0.30	79.94±0.27	77.83±0.45
	TAM	95.25±0.06	94.11±0.19	93.66±0.16	81.12±0.04	80.29±0.03	79.32±0.12
	GraphSHA	96.27±0.14	95.14±0.04	94.94±0.14	82.60±0.30	80.34±0.31	80.00±0.52
	BufferGraph	96.26±0.17	**96.13**±0.14	**96.15**±0.27	**83.42**±0.52	**84.21**±0.35	**81.58**±0.45

case. While all methods show some performance degradation as the imbalance ratio increases, BufferGraph exhibits the most stable performance, demonstrating its robustness to severe class imbalance. This stability can be attributed to BufferGraph's buffer node mechanism, which effectively modulates message passing between majority and minority classes regardless of their relative proportions.

3.4 RQ3: Ablation Study

Experimental Setup. To assess the contribution of each component in BufferGraph, we conduct ablation studies on Amazon-Computers and WikiCS datasets using the GCN backbone under the random splitting setting. BufferGraph uniquely combines buffer node insertion with a specialized message passing scheme and a difference loss. We examine four variants: the **Complete BufferGraph** (the full model); **- DL**, which ablates the difference loss component; **- CMP** (Concrete Message Passing), which removes the adaptively weighted direct message path between the original connected nodes, thereby routing all interaction through the buffer node pathway; and **- UMP** (Update Message Passing via Buffer), which ablates the message pathway through the inserted buffer node, relying only on the adaptively weighted direct path between original nodes.

Results and Analysis. Figure 5 shows that removing the update message passing mechanism (- UMP) leads to the most significant performance drop across all metrics, particularly in F1 scores. This confirms that adaptive message passing is the core component of BufferGraph, as it directly controls how information flows between majority and minority nodes. The removal of other components also notably impacts model performance, demonstrating that each component plays an essential role in BufferGraph's effectiveness. The complete BufferGraph consistently outperforms all ablated variants, confirming the synergistic effect of combining all components.

3.5 RQ4: Parameter Sensitivity Analysis

Experimental Setup. We investigate the sensitivity of BufferGraph to two critical hyperparameters: **(1) Mixup coefficient** α controlling feature interpolation for buffer nodes (tested values: 0.1, 0.3, 0.5, 0.7, 0.9), and **(2) Difference loss weight** λ balancing prediction loss and difference loss (tested values: 0.1, 0.5, 1, 2, 5).

Results and Analysis. Figure 6 reveals that: (1) For α, optimal performance occurs at 0.5 across both datasets, suggesting equal feature mixing is ideal for buffer nodes; (2) For λ, Amazon-Computers shows stable performance from 0.1 to 1 while WikiCS improves over this range, but both degrade significantly at $\lambda \geq 2$. These results demonstrate BufferGraph's robustness within moderate parameter ranges ($\alpha = 0.5$, $\lambda = 1.0$), while extreme values harm performance. The optimal settings balance feature mixing and loss weighting effectively across diverse datasets.

3.6 RQ5: Buffer Node Ratio Study

Experimental Setup. We investigate how the ratio of inserted buffer nodes affects model performance under the random splitting setting on Amazon-Computers and WikiCS datasets. We vary the number of buffer nodes in the graph to evaluate the sensitivity of our approach. Specifically, we randomly selected a certain percentage of all eligible edges to add buffer nodes varying from 0% to 100%.

Results and Analysis. Our experiments consistently show that higher insertion ratios lead to improved performance, with the best results achieved when buffer nodes are inserted for all potentially misleading edges (100% insertion ratio). This confirms that our strategy of comprehensive buffer node insertion is effective and that the benefits of modulating message passing outweigh any potential drawbacks from increased graph complexity. Detailed results are provided in Appendix G.5.

4 Related Work

Existing methods to tackle class imbalance in graphs can be categorized into two main approaches. The first approach focuses on generating synthetic nodes and edges [18–20, 22, 33, 34], where methods like DRGCN [33] and GraphSMOTE [19] generate minority nodes to balance classes, while GraphSHA [22] focuses on generating 'hard' nodes of minority classes to enlarge the margin between majority and minority classes. The second approach modifies the learning process itself, such as TAM [21] which introduces connectivity-aware margins. Recently, BAT [35] addresses class imbalance from a topological perspective by theoretically identifying two fundamental phenomena that amplify class-imbalance bias, serving as a plug-and-play module for existing class rebalancing methods.

However, existing methods either overlook the direct influence of majority classes on minority classes through majority-minority edges (empirically demonstrated in Figure 1), or function merely as auxiliary modules without providing a complete solution. In contrast, BufferGraph offers a comprehensive end-to-end model that fundamentally redesigns the message passing mechanism using buffer nodes to effectively address the class imbalance problem. We provide a comparison between previous works and BufferGraph to highlight our novelty in Table 4.

Method	Handle Majority-Minority Edges	Label-Free Node Generation	Message Propagation Adjustment	No Complex Parameter Tuning
GraphSMOTE [19]	✗	✗	✗	✗
GraphENS [20]	✗	✗	✗	✗
TAM [21]	✗	✓	✓	✗
GraphSHA [22]	✗	✗	✗	✗
DRGCN [33]	✗	✗	✗	✗
BAT [35]	✗	✓	✗	✓
BufferGraph	✓	✓	✓	✓

5 Conclusion

We introduce BufferGraph, a lightweight framework that addresses class imbalance in graph neural networks through strategic buffer node insertion. Unlike previous approaches focusing on minority node generation, BufferGraph modulates message passing through potentially misleading edges, effectively reducing majority nodes' negative influence on minority nodes. Our experiments on five real-world datasets demonstrate BufferGraph's superior and robust performance, particularly for minority classes with high percentages of majority-class neighbors.

Acknowledgements

This research is supported by the National Research Foundation, Singapore, and the Infocomm Media Development Authority under its Trust Tech Funding Initiative. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Research Foundation, Singapore, and the Infocomm Media Development Authority.

References

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1, 5

[2] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 1

[4] Fenyu Hu, Liping Wang, Shu Wu, Liang Wang, and Tieniu Tan. Graph classification by mixture of diverse experts. *arXiv preprint arXiv:2103.15622*, 2021. 1

[5] Dingyi Zeng, Wanlong Liu, Wenyu Chen, Li Zhou, Malu Zhang, and Hong Qu. Substructure aware graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11129–11137, 2023.

[6] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 417–428, 2023. 1

[7] Xinyu Huang, Dongming Chen, Dongqi Wang, and Tao Ren. Identifying influencers in social networks. *Entropy*, 22(4):450, 2020. 1

[8] Jayati Bhadra, Amandeep Singh Khanna, and Alexei Beuno. A graph neural network approach for identification of influencers and micro-influencers in a social network:* classifying influencers from non-influencers using gnn and gcn. In *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)*, pages 66–71. IEEE, 2023.

[9] Alexander K Taylor, Nuan Wen, Po-Nien Kung, Jiaao Chen, Violet Peng, and Wei Wang. Where does your news come from? predicting information pathways in social media. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2511–2515, 2023. 1

[10] Dan Lin, Jiajing Wu, Qi Xuan, and K Tse Chi. Ethereum transaction tracking: Inferring evolution of transaction networks via link prediction. *Physica A: Statistical Mechanics and its Applications*, 600:127504, 2022. 1

[11] Kiarash Shamsi, Friedhelm Victor, Murat Kantarcioglu, Yulia Gel, and Cuneyt G Akcora. Chartalist: Labeled graph datasets for utxo and account-based blockchains. *Advances in Neural Information Processing Systems*, 35:34926–34939, 2022.

[12] Qian Wang, Zhen Zhang, Zemin Liu, Shengliang Lu, Bingqiao Luo, and Bingsheng He. Etgraph: A pioneering dataset bridging ethereum and twitter. *arXiv preprint arXiv:2310.01015*, 2023. 1

[13] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. Not all neighbors are friendly: Learning to choose hop features to improve node classification. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pages 4334–4338, 2022. 1

[14] Jianke Yu, Hanchen Wang, Xiaoyang Wang, Zhao Li, Lu Qin, Wenjie Zhang, Jian Liao, and Ying Zhang. Group-based fraud detection network on e-commerce platforms. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5463–5475, 2023. 1

[15] Yuan Li, Bingqiao Luo, Qian Wang, Nuo Chen, Xu Liu, and Bingsheng He. Cryptotrade: A reflective llm-based agent to guide zero-shot cryptocurrency trading. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1094–1106, 2024. 1

[16] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. 1, 2, 6

[17] Min Liu, Siwen Jin, Luo Jin, Shuohan Wang, Yu Fang, and Yuliang Shi. Imbalanced nodes classification for graph neural networks based on valuable sample mining. In *Proceedings of the 2022 6th international conference on electronic information technology and computer engineering*, pages 1957–1962, 2022. 1

[18] Mengting Zhou and Zhiguo Gong. Graphsr: A data augmentation algorithm for imbalanced node classification. *arXiv preprint arXiv:2302.12814*, 2023. 1, 9

[19] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 833–841, 2021. 1, 2, 6, 9, 14

[20] Joonhyung Park, Jaeyun Song, and Eunho Yang. Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *The Tenth International Conference on Learning Representations, ICLR 2022*. International Conference on Learning Representations (ICLR), 2022. 1, 6, 9, 14

[21] Jaeyun Song, Joonhyung Park, and Eunho Yang. Tam: topology-aware margin loss for class-imbalanced node classification. In *International Conference on Machine Learning*, pages 20369–20383. PMLR, 2022. 1, 6, 9, 14

[22] Wen-Zhi Li, Chang-Dong Wang, Hui Xiong, and Jian-Huang Lai. Graphsha: Synthesizing harder samples for class-imbalanced node classification. *arXiv preprint arXiv:2306.09612*, 2023. 1, 2, 4, 6, 9, 14, 15

[23] Meilun Shi, Enguang Zuo, Hanwen Qu, and Xiaoyi Lv. Graphuc: Predictive probability-based graph-structured data augmentation model for class imbalance node classification. In *2024 4th International Conference on Electronic Information Engineering and Computer Science (EIECS)*, pages 933–936. IEEE, 2024.

[24] Nan Chen, Zemin Liu, Bryan Hooi, Bingsheng He, Jun Hu, and Jia Chen. Nodeimport: Imbalanced node classification with node importance assessment. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 94–105, 2025. 1

[25] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018. 2, 6

[26] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020. 2, 6

[27] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 4

[28] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019. 6, 14

[29] Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang. Disentangling label distribution for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6626–6636, 2021. 6, 14

[30] Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020. 6, 14

[31] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019. 6, 14

[32] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33:7793–7804, 2020. 6, 14

[33] Min Shi, Yufei Tang, Xingquan Zhu, David Wilson, and Jianxun Liu. Multi-class imbalanced graph convolutional network learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2020. 9

[34] Liang Qu, Huaisheng Zhu, Ruiqi Zheng, Yuhui Shi, and Hongzhi Yin. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1390–1398, 2021. 9

[35] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. Class-imbalanced graph learning without class rebalancing. In *Forty-first International Conference on Machine Learning*, 2024. 9

[36] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *Advances in neural information processing systems*, 35:1362–1375, 2022. 14

A Dataset Details

We provide the statistics of datasets used in the paper in Table 5.

Table 5: Statistics of datasets used in the paper.

Dataset	#Nodes	#Edges	#Features	#Classes	#Max / Min
Amazon-Photos	7,650	119,081	745	8	5.86
Amazon-Computers	13,752	245,861	767	10	17.72
Coauthor-CS	18,333	163,778	6,805	15	35.05
Coauthor-Physics	34,493	247,962	8,415	5	6.33
WikiCS	11,701	216,123	300	10	9.08

Table 6: Per-class node counts for all experimental datasets. Counts are listed in descending order to illustrate the imbalance.

Dataset	**# Classes**	**Node Count per Class (Descending Order)**
Amazon-Photos	8	[2346, 1269, 1175, 962, 796, 521, 480, 101]
Amazon-Computers	10	[4610, 2125, 1729, 1319, 1196, 1139, 703, 563, 308, 60]
Coauthor-CS	15	[2564, 2228, 1968, 1735, 1478, 1373, 1261, 1239, 1170, 1081, 911, 715, 418, 160, 32]
Coauthor-Physics	5	[12994, 9152, 6311, 4426, 1610]
WikiCS	10	[2708, 1878, 1335, 1098, 971, 940, 893, 769, 714, 395]

And these datasets' licenses are as follows:

- **Amazon-Photos/Computers**: Sourced from Amazon product data under MIT license.
- **Coauthor-CS/Physics**: Academic collaboration networks from arXiv with CC-BY 4.0 license.
- **WikiCS**: Wikipedia article network dataset (v1.0) under CC-BY-SA 3.0 license.

B Notations in BufferGraph

We provide all notations we use in BufferGraph paper in Table 7.

C Algorithm

We provide the full algorithm in Algorithm 1.

Table 7: Key Notations Used in BufferGraph

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$
$\boldsymbol{A} \in \{0,1\}^{N \times N}$	Adjacency matrix
$\boldsymbol{X} \in \mathbb{R}^{N \times d}$	Node feature matrix
v_{buf}	Buffer node
$s_{(v_i, v_j)}$	Difference score between nodes v_i and v_j
$w_{(v_i, v_j)}$	Edge weight between nodes v_i and v_j

Algorithm 1: BufferGraph: Buffer Node Synthesis and Adaptive Message Passing

Input: Graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{A})$, training set nodes $\mathcal{V}^L$ and their labels $\boldsymbol{Y}^L$, number of classes C, mixup coefficient α, distance loss coefficient λ
Output: Trained GNN model f_θ

```
/* Buffer Node Generation                                          */
for each edge (v_src, v_tar) in G do
    X_buf ← αX_src + (1 − α)X_tar
    Insert buffer node v_buf with feature X_buf into G between v_src and v_tar
end
/* Pre-training Stage                                              */
Initialize vanilla GCN f_θ
Ŷ ← PreTrainGCN(G, V^L, Y^L)
Identify minority nodes based on Ŷ
Decide edges to insert buffer nodes in G (exclude edges connected to predicted minority nodes)
/* Training Stage                                                  */
best_val_loss ← ∞
Define warmup_epochs
for epoch ← 1 to N do
    Compute node prediction loss L_pred
    Compute difference loss L_diff
    L_total ← L_pred + λ · L_diff
    Update f_θ by minimizing L_total with standard backpropagation
    if epoch > warmup_epochs then
        if val_loss < best_val_loss then
            best_val_loss ← val_loss
            for each edge e_(v_i,v_j) ∈ G do
                s_(v_i,v_j) ← Σ_k |ŷ_i,k − ŷ_j,k|
                if edge e_(v_i,v_j) has a buffer node then
                    w_(v_i,v_j) ← 1 − s_(v_i,v_j)
                end
            end
        end
    end
end
return trained model f_θ
```

D Complexity Analysis

Given N as the total number of nodes and E as the total number of edges within the graph, the creation of buffer nodes correlates with the subset of the node and edge sets, leading to a computational complexity of $O(E)$. Generating augmented features for buffer nodes requires $O(E \cdot d)$ time, with d representing the dimensionality of the node features. Additionally, forming augmented edges for the buffer nodes incurs a time complexity of $O(E)$, engaging the whole graph's edge set. Hence, the total additional complexity introduced by our model is $O(E + E \cdot d + E)$. As demonstrated by

successful experiments on extensive datasets such as Coauthor-Physics and WikiCS, BufferGraph can work well on the large datasets.

E Experimental Environment

All models in our experiments were implemented using Pytorch 2.0.0 in Python 3.9.16, and run on a robust Linux workstation. This system is equipped with two Intel(R) Xeon(R) Gold 6226R CPUs, each operating at a base frequency of 2.90 GHz and a max turbo frequency of 3.90 GHz. With 16 cores each, capable of supporting 32 threads, these CPUs offer a total of 64 logical CPUs for efficient multitasking and parallel computing. The workstation is further complemented by a potent GPU setup, comprising eight NVIDIA GeForce RTX 3090 GPUs, each providing 24.576 GB of memory. The operation of these GPUs is managed by the NVIDIA-SMI 525.60.13 driver and CUDA 12.0, ensuring optimal computational performance for our tasks.

F Baselines

- **Loss Management Strategies:**
  1. Reweight [28]: This adjusts the weights of classes based on their propotion to the class samples in the dataset, aiming to counteract imbalance.
  2. PC SoftMax [29]: This improves the model's probability calibration for multi-class classification, ensuring more precise probabilistic predictions for minority classes.
  3. Cross Entropy: It acts as a core loss function by quantifying the divergence between predicted and actual distributions.
  4. Balanced Softmax [30]: It is designed to minimize the generalization bound in multi-class softmax regression.
- **Heterophilious GCNs**
  1. MixHop [31]: It extracts the features from multi-hop neighbors to get more information.
  2. H2GCN [32] [36]: It combines three key designs to address heterophily: (1) ego- and neighbor-embedding separation; (2) higher-order neighborhoods; (3) combination of intermediate representations.
- **Class-imbalanced Node Classification Approaches:**
  1. GraphSMOTE [19]: It synthesizes new instances in minority classes to directly tackle class imbalance.
  2. GraphENS [20]: It utilizes ensemble strategies to generate new instances in minority classes.
  3. TAM [21]: It introduces connectivity and distribution-aware margin to guide minority classes' classification.
  4. GraphSHA [22]: It focus on generating hard minority samples to enlarge the margin between minority and majority classes.

For general hyper-parameters such as learning rates and the number of layers, we explore layers of 2,3 and explore hidden dimensions of 64, 128, and 256. Among these, we select the layer and dimension that provide the best performance of each baseline on each dataset. For the learning rate, we explore the optimal settings by evaluating the performance at values of 0.001, 0.005, and 0.01. For the dropout rate, we explore the optimal settings for evaluating the performance of 0.1, 0.2, 0.3, 0.4, 0.5.

For GraphSmote, we opt for the GraphSmote$_O$ variant, which is tailored to predict discrete values without necessitating pretraining, showcasing superior performance among its various versions [19].

In the implementation of GraphENS, we set the feature masking rate k to 0.01 and the temperature τ to 1 according to the paper's configuration [20].

For TAM, we select the GraphENS-based iteration, which is identified as the most performant according to the findings reported in the corresponding paper. The default settings from the released code are utilized, with the coefficients for ACM α, ADM β, and classwise temperature ϕ set to 2.5, 0.5, and 1.2, respectively [21].

Specifically for GraphSHA, we follow the parameter configurations detailed in the original study, employing the PPR version with a setting of $\alpha = 0.05$ and $K = 128$ [22].

G Experiment Results

G.1 Imbalanced Splitting Experiment

In this section, we provide the results of the imbalanced splitting experiment in Table 3 and Table 4.

G.2 Imbalance Ratio Experiment Results

In this section, we provide the results of the imbalance ratio experiment on Amazon-Computers dataset in Figure 4.

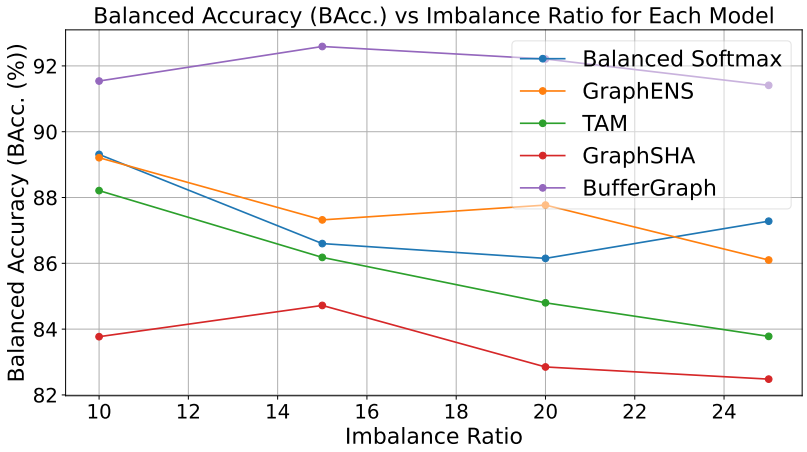

Figure 4: Balanced accuracy (BAcc.) with increasing imbalance ratios on Amazon-Computers. BufferGraph maintains superior performance even as imbalance becomes more severe.

G.3 Ablation Study Results

In this section, we provide the results of the ablation study on Amazon-Computers and WikiCS in Figure 5.

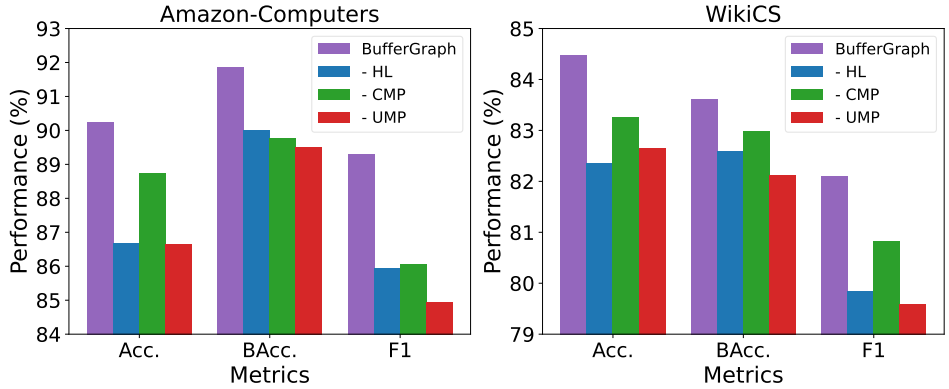

Figure 5: Ablation study on Amazon-Computers and WikiCS.

G.4 Parameter Sensitivity Experiment Results

In this section, we provide the results of the parameter sensitivity experiment on Amazon-Computers and WikiCS in Figure 6.

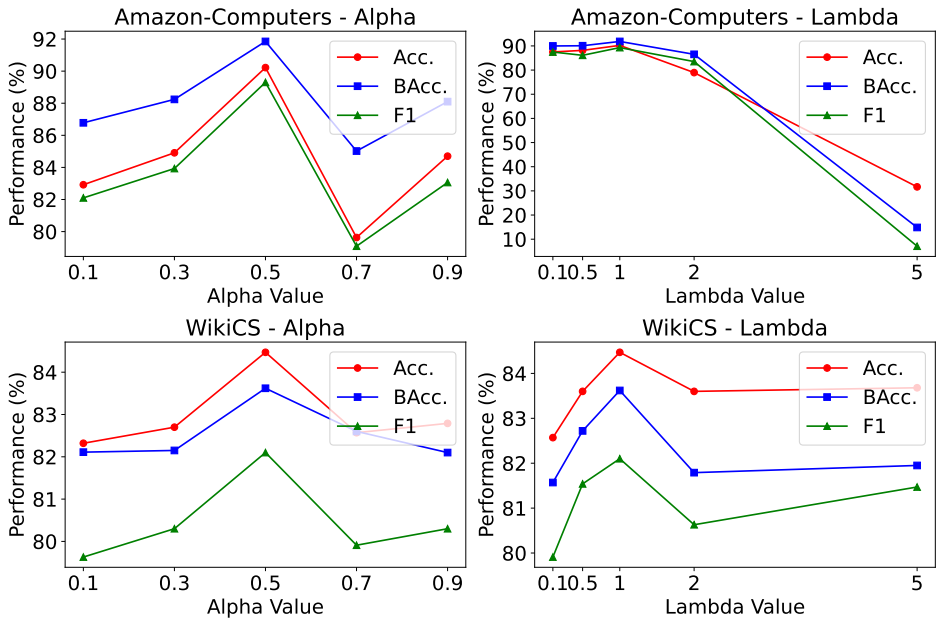

Figure 6: Parameter sensitivity analysis on Amazon-Computers and WikiCS.

G.5 Insertion Ratio Experiment Results

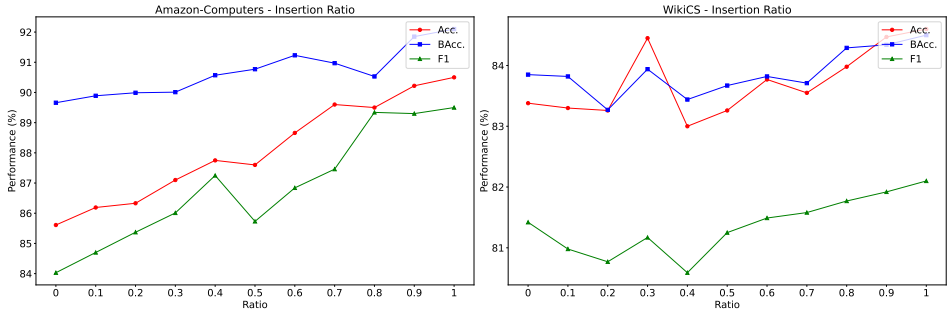

Figure 7: Insertion ratio experiments on Amazon-Computers and WikiCS.

We use the pre-training predicted results to identify the edges most likely to connect majority and minority nodes. We apply various ratios: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0 to select the top potential edges for inserting buffer nodes and analyze their performance. For instance, a 0.1 ratio means that only the top 10% of potential majority-minority edges, excluding edges neighboring minority nodes, would be inserted with buffer nodes. To quantify the likelihood of an edge connecting majority-minority nodes, we calculate a score for each edge using the following formula:

$$\text{score}(e) = \frac{1}{2}\Big(\text{prob\_minority}(v_i) \cdot \text{prob\_majority}(v_j) + \text{prob\_minority}(v_j) \cdot \text{prob\_majority}(v_i)\Big)$$

where $e = (v_i, v_j) \in \mathcal{E}$ is an edge, $\text{prob\_minority}(v)$ is the sum of the probabilities of node v belonging to minority classes, and $\text{prob\_majority}(v)$ is the sum of the probabilities of node v belonging to majority classes. The scores are then normalized between 0 and 1:

$$\text{normalized\_score}(e) = \frac{\text{score}(e) - \min(\text{score}(e))}{\max(\text{score}(e)) - \min(\text{score}(e))}$$

This edge scoring mechanism effectively identifies potential majority-minority connections, guiding the strategic insertion of buffer nodes. The experimental results show that higher insertion ratios consistently lead to better performance, validating the effectiveness of BufferGraph's design.

H Limitations and Broader Impacts

Limitations. BufferGraph has two main limitations: (1) using a fixed mixup coefficient for all buffer nodes may not be optimal across different graph structures, and (2) its effectiveness is currently only validated for node classification, leaving other graph tasks (e.g., edge prediction) for future work.

Broader Impact. By improving minority class prediction in imbalanced graphs, BufferGraph has potential applications in domains where fairness and equity are crucial, such as social network analysis, recommendation systems, and fraud detection. Addressing class imbalance can help reduce algorithmic bias against underrepresented groups in graph-based applications. Future work could explore dynamic buffer node generation strategies, adaptive mixup coefficients based on node characteristics, and extensions to other graph learning tasks beyond node classification.