UNREAL:Unlabeled Nodes Retrieval and Labeling for Heavily-imbalanced Node Classification

Liang Yan [1 *] **Shengzhong Zhang** [1 *] **Bisheng Li** [1] **Min Zhou** [2] **Zengfeng Huang** [1]

Abstract

Extremely skewed label distributions are common in real-world node classification tasks. If not dealt with appropriately, it significantly hurts the performance of GNNs in minority classes. Due to its practical importance, there have been a series of recent research devoted to this challenge. Existing over-sampling techniques smooth the label distribution by generating "fake" minority nodes and synthesizing their features and local topology, which largely ignore the rich information of unlabeled nodes on graphs. In this paper, we propose UNREAL, an iterative over-sampling method. The first key difference is that we only add unlabeled nodes instead of synthetic nodes, which eliminates the challenge of feature and neighborhood generation. To select which unlabeled nodes to add, we propose geometric ranking to rank unlabeled nodes. Geometric ranking exploits unsupervised learning in the node embedding space to effectively calibrates pseudo-label assignment. Finally, we identify the issue of geometric imbalance in the embedding space and provide a simple metric to filter out geometrically imbalanced nodes. Extensive experiments on real-world benchmark datasets are conducted, and the empirical results show that our method significantly outperforms current state-of-the-art methods consistent on different datasets with different imbalance ratios.

1. Introduction

Node classification is ubiquitous in real-world scenarios e.g., social network and commercial graph analysis. Generally, real-world data come with an imbalanced class distribution (Mohammadrezaei et al., 2018; Wang et al., 2020b). A classifier trained using an imbalanced dataset is prone to have biased accuracy on under-represented classes. While GNNs have achieved superior performance on node classification, training a fair GNN model for handling highly-imbalanced class distributions remains a challenging task (Liu et al., 2018; Zhao et al., 2021).

Several recent studies have been devoted to solving the imbalanced node classification problem (Zhao et al., 2021; Shi et al., 2020; Chen et al., 2021; Park et al., 2021; Song et al., 2022). One strategy is to adapt over-sampling, a widely used technique in other domains, to graph data. However, it is a non-trivial task, since one needs to additionally generate topological information for newly synthesized nodes. Different approaches to synthesize new nodes together with their feature and relational information have been proposed, e.g., (Zhao et al., 2021; Shi et al., 2020; Park et al., 2021). Adding synthetic data introduces additional noise (which harms classification accuracy) and extra computation burden, especially in graph learning. On the other hand, abundant unlabeled nodes are often available in large-scale graph learning tasks. *Can we just add unlabeled nodes instead of synthesizing a large amount of "fake" nodes?* Thus far, the value of unlabeled nodes in imbalanced semi-supervised classification has not been explored.

In this work, we propose a novel imbalanced node classification method: **u**nlabeled **n**ode **re**trieval **a**nd **l**abeling (UNREAL). At a high level, UNREAL is an oversampling approach similar to self-training (ST) (Yarowsky, 1995; Lee et al., 2013), which adds unlabeled nodes together with their pseudo-labels (predictions made by a previous-learned model) to the training set. Since there is no need for synthesizing node features and topology, it overcomes critical shortcomings of existing oversampling approaches.

It is noteworthy that ST has shown to be effective in dealing with label sparsity in node classification (Li et al., 2018; Zhou et al., 2019; Sun et al., 2020; Wang et al., 2021c). Extensive experiments in this work show that ST is effective for imbalanced learning as well but fails to achieve satisfactory performance in heavily-imbalanced scenarios. This is mainly because the bias in the original training set results in unreliable predictions, which makes the pseudo-labels used in ST highly noisy. In this paper, we introduce effective techniques to overcome the initial bias in ST to further boost

[1]Fudan University, Shanghai, China [2]Huawei Technologies Co., Ltd Shengzhen, China. Correspondence to: Zengfeng Huang <huangzf@fudan.edu.cn>.

the performance on heavily-imbalanced node classification. The key idea is to explore the geometric structure in the embedding space to calibrate the bias in pseudo-labels.

This is partially inspired by the work of (Kang et al., 2019) from computer vision, where they hypothesize and verify empirically that the classifier is the only under-performed component in the model when trained on an imbalanced training set. In UNREAL, after the preliminary training step in ST, we retrieve node embeddings from the output layer (before the classification layer) and use unsupervised clustering methods for label prediction, which compensate for the prediction made by the supervised model. As in (Chen et al., 2021), we wish to add nodes that are closer to their class centers, and unsupervised learning in the embedding space also provides a natural way to rank the closeness of nodes to their (predicted) class centers. As in standard ST, multiple rounds of pseudo-label fitting will be applied. Extensive experimental studies show that our framework outperforms existing imbalanced node classification methods by a large margin in most settings.

We summarize our contribution as follows: 1) As far as we know, this is the first work that systematically studies the performance of ST as an over-sampling technique for imbalanced node classification; 2) from our empirical results, we identify deficiencies of ST in heavily-imbalanced scenarios, and propose to apply unsupervised methods in the embedding space to overcome the drawbacks; 3) we introduce several simple yet effective techniques in the implementation to optimize the performance, e.g., we identify the Geometric Imbalance (GI) issue in the embedding space and propose a metric to measure GI and discard imbalanced nodes; 4) we conduct comprehensive experiments on multiple benchmarks which demonstrates the superiority of our framework.

2. Preliminaries

2.1. Notations and Definitions

In this work, we mainly focus on semi-supervised node classification on an undirected and unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$. Here, $\mathcal{V}$ is the node set, $\mathcal{E}$ is the edge set, and $\mathcal{L} \subset \mathcal{V}$ denotes the set of labeled nodes. So the set of unlabeled nodes is $\mathcal{U} = \mathcal{V} - \mathcal{L}$. Let $\mathcal{X} \in \mathbb{R}^{n \times f}$ be the feature matrix (where $n = |\mathcal{V}|$ and f is the feature dimension). We use $A \in \{0, 1\}^{n \times n}$ to denote the adjacency matrix and $\mathcal{N}(v)$ the set of 1-hop neighbors of node v. The labeled sets for all classes are denoted by $(\mathcal{C}_1, \mathcal{C}_2, \cdots, \mathcal{C}_k)$, where k is the number of different classes. We use imbalance ratio, defined as $\rho := \frac{\max_i(|\mathcal{C}_i|)}{\min_i(|\mathcal{C}_i|)}$, to measure the level of imbalance in a dataset. We summarize the notation in Appendix F.

2.2. Message Passing Neural Networks for Node Classification

A standard MPNN consists of three components, a message function m_l, an information aggregation function θ_l, and a node feature update function ψ_l. The feature of each node is updated iteratively. Let h_v^l be the feature of node v in the l-th layer, then in the $(l + 1)$-th layer the feature is:

$$h_v^{(l+1)} = \psi_l \left(h_v^{(l)}, \theta_l \left(\left\{ m_l \left(h_v^{(l)}, h_u^{(l)}, e_{v,u} \right) \mid u \in \mathcal{N}(v) \right\} \right) \right) \tag{1}$$

where $e_{v,u}$ is the edge weight between v and u. For the classic GCN model (Kipf & Welling, 2016), $h_v^{(l+1)}$ is computed as: $h_v^{(l+1)} = \Phi^l \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{e_{v,u}}{\sqrt{\hat{d}_u \hat{d}_v}} h_u^{(l)}$, where Φ^l is the parameter matrix of the l-th layer and $\hat{d}_v = 1 + \sum_{u \in \mathcal{N}(v)} e_{v,u}$. For node classification, a classification layer is concatenated after the last layer of a GNN.

3. Imbalanced Learning with Unlabeled Data

In this work, we try to harness the positive value of unlabeled nodes for imbalance learning. We first conduct extensive experiments to confirm the effectiveness of ST in boosting imbalanced learning on graph-structured data. Our experimental results show that as the imbalance ratio increase, standard ST is more likely to add nodes with wrong pseudo-labels, which leads to performance degradation.

3.1. ST Improves the Performance of GNN in Imbalanced Learning

Yang & Xu (2020) considered a binary classification problem with the data generating distribution is a mixture of two Gaussians and assumed that a base classifier is trained on imbalanced data. Their analysis of this simple model shows: (1) ST boosts imbalanced learning and more unlabeled data is always helpful. (2) data imbalance affects the probability of obtaining a good estimation. Graph learning is more complicated and such a simple model could miss some critical aspects of graph learning.

We first conduct extensive experiments to confirm the effectiveness of ST in boosting imbalanced learning on graph-structured data. We repeat each experiment five times and report the average experiment results on Cora under different imbalance ratios in Figure 1(a).[1] It can be observed that across different ratios, ST consistently outperforms vanilla model by a large margin, which verifies the positive value of the unlabeled samples of graph-structured data. However, as imbalance ratio increases, it is observed that the performance of ST degrades rapidly, which renders that ST is insufficient for high imbalance ratios. It can be observed

[1]More results on different datasets and models can be found in Appendix A.

that UNREAL consistently outperforms ST by a large margin, and as imbalance ratio increases, the gap of the F1 scores between ST and our method becomes larger. We believe the main reason is that, due to imbalance in the original training, ST adds low-quality nodes into the training set in the early stages. We conduct experiments to verify this in Section 3.2.

3.2. Pseudo-label Misjudgment Augmentation Problem in Imbalanced Learning

As we mentioned, since ST adds pseudo-labels to the training set and trains the model iteratively, misjudgements in the early stages will cause the method to fail badly. Here, we hypothesize that as the imbalance ratio of the dataset becomes larger, the pseudo-labels obtained by ST-based methods are less credible.

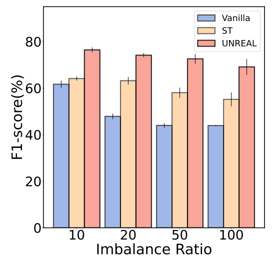
Experimental Setup. We first conduct experiments to test the accuracy of pseudo-label for unlabeled nodes on class-imbalanced graphs. ST based on GCN are trained on Cora. We process the dataset to make it imbalanced following Zhao et al. (2021); Park et al. (2021); Song et al. (2022). The imbalance ratio ρ is set as 1, 5, 10, 20, 50, 100. We test the accuracy of pseudo-labels for unlabeled nodes which are newly added to the training set. More specifically, we examine 100 nodes that join the majority class and 100 nodes that join the minority class. We repeat each experiment five times and report the average experiment results.

Pseudo-label Misjudgment Augmentation Problem. The accuracy of pseudo-labels for unlabeled nodes which are selected into the minority class and the majority class are reported in Figure 1(b).[2] We can observe that as ρ becomes larger, the accuracy of pseudo labels for unlabeled nodes selected into the minority class decreases quickly. For unlabeled nodes selected into the majority class, the accuracy of pseudo-labels is stable at a high level, which confirms the heavy bias of the base classifier trained on highly imbalanced data. As a reference, for both majority class or minority class, UNREAL consistently outperforms ST in pseudo-label accuracy.

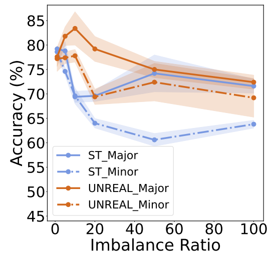
3.3. More Elaboration and Experiments.

In Appendix A, we provide more experimental details and more results on different benchmarks and base models to strengthen our finds in Section 3.1 and Section 3.2.

[2]More results on different datasets and models can be found in Appendix A.



(a) Cora-GCN



(b) Cora-GCN

Figure 1. (a) The experimental results on Cora under different imbalance scenarios (ρ = 10, 20, 50, 100). We compare the F1-score (%) with the standard errors of ST and UNREAL. (b) The experimental results on Cora under various imbalance scenarios are presented. We choose 100 unlabeled nodes that have recently been added to the training set using ST & UNREAL, and we evaluate their performance using GCN by comparing their accuracy (%) with the standard errors of these nodes' pseudo labels. Minor means that we only test unlabeled nodes from the minority classes, and Major means that we only test unlabeled nodes from the majority classes.

4. UNREAL

In this section, we provide the details of the proposed method. UNREAL iteratively adds unlabeled nodes (with predicted labels) to the training set and retrains the model. We propose three complementary techniques to enhance the unlabeled node selection and labeling. More specifically, in Section 4.1, we describe Dual Pseudo-tag Alignment Mechanism (DPAM) for effective node filtering, the key idea of which is to use unsupervised clustering in the embedding space to obtain a node ranking, namely *geometric ranking*. In Section 4.2, we show how to combine geometric rank from unsupervised learning and confidence ranking from supervised learning to reorder unlabeled nodes according to their closeness to the class centers (Node-Reordering). Finally, in Section 4.3, we identify the issue of geometric node imbalance (GI) and define a new metric to measure GI, which is then used to filter out nodes with high GI. The overall pipeline of UNREAL is illustrated in Figure 2. The pseudo-code of the full algorithm is provided in Appendix E (Algorithm 1).

4.1. Dual Pseudo-tag Alignment Mechanism for Node Filtering DPAM

Motivating Example. As we verified above, in heavily-imbalanced scenarios, the pseudo-labels given by the classifier are much less reliable. Kang et al. (2019) hypothesize and verify empirically that the classifier is the only under-performed component in the model when trained on an imbalanced training set. We conduct experiments to verify

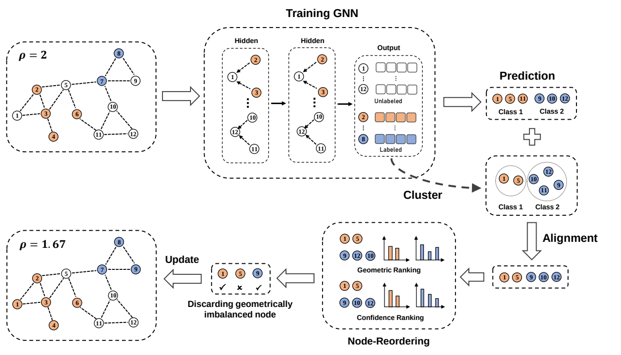


Figure 2. Overall pipeline of our UNREAL. Colored nodes denote labeled nodes. Parameters in the GNN Model and the classifier are trained together using the current training set.

this on imbalanced node classification.

Labeled

Let d be the embedding dimension. We use $H^L \in \mathbb{R}^{|\mathcal{L}|\times d}$ and $H^U \in \mathbb{R}^{|\mathcal{U}|\times d}$ unlabel to denote the embedding matrix of labeled and unlabeled nodes respectively. Each row of the embedding matrix is the embedding of a node u (denoted as h_u^L and h_u^U), which is considered as a point in the d-dimension Euclidean space. We apply an unsupervised clustering algorithm, f_{cluster}, which partitions the embeddings of unlabeled nodes into k' clusters and produces k' corresponding cluster centers, where k' is usually larger than k, the number of classes.

$$f_{\text{cluster}}(H^U) \implies \{\mathcal{K}_1, c_1, \mathcal{K}_2, c_2, \cdots, \mathcal{K}_{k'}, c_{k'}\} \tag{2}$$

where $\mathcal{K}_i$ is the i-th cluster and c_i is the i-th cluster center. We use ~~vanilla k-means in our implementation. We also~~ compute the embedding center of each class in the training set

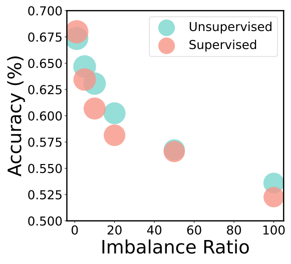
$$c_i^{\text{train}} = M(\{h_u^L \mid y_u \in \mathcal{C}_i\}). \tag{3}$$

Since we use k-means in our experiments, $M(\cdot)$ is simply the mean function. We next assign a pseudo-label $\tilde{y}_m$ to each cluster $\mathcal{K}_m$:

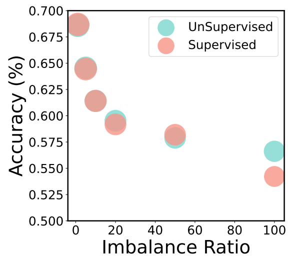
$$\tilde{y}_i = \arg\min_j \text{distance}(c_j^{\text{train}}, c_i). \tag{4}$$

We then combine clusters with the same pseudo-label m as $\tilde{\mathcal{U}}_m$, and $\mathcal{U} = \bigcup_{m=1}^{k} \tilde{\mathcal{U}}_m$. On the other hand, the supervised GNN model gives each node u in $\mathcal{U}$ a prediction $\hat{y}_u$, and we put unlabeled nodes whose prediction is m into the set $\mathcal{U}_m$, and again we have $\mathcal{U} = \bigcup_{m=1}^{k} \mathcal{U}_m$.

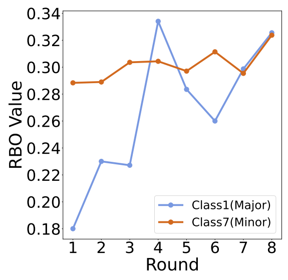
Analysis. Thus far, we obtain two pseudo-labels for each unlabeled node, which come from unsupervised and supervised methods respectively. We conduct experiments on Cora to compare the accuracy of the two pseudo-labels. The experiment settings are the same as Section 3.2. As shown in Figure 3(a) and Figure 3(b), it can be found that the embeddings learned by the GNN encoder are still of high quality to get an effective prediction based on unsupervised algorithms. Especially in heavily-imbalanced scenarios, the reliability of unsupervised algorithms is even higher than the supervised classifier. Based on this, we proposed DPAM. [3]



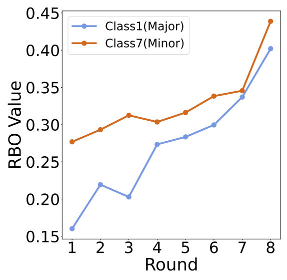
(a) Cora-GCN



(b) Cora-GAT



(c) Cora-GCN



(d) Cora-GAT

Figure 3. (a, b) The partial experimental results on Cora under different imbalance scenarios(ρ = 1, 5, 10, 20, 50, 100). We compare the accuracy of the two pseudo-labels(predictions) from unsupervised algorithms and supervised classifiers respectively for all unlabeled nodes. (c, d) Fluctuation of RBO values (similarity) of two rankings as iterations progress.

Dual Pseudo-tag Alignment Mechanism (DPAM). The pseudo-labels produced by applying an unsupervised algorithm on the embeddings provide an alternative and potentially less biased prediction, which may compensate for the bias introduced by the imbalanced training set. At the same time, the overall accuracy of the unsupervised algorithm is inferior to supervised methods when the training set gradually becomes balanced, and thus it is sub-optimal to rely solely on the pseudo-labels from clustering. As a

[3] We elaborate on the experimental details and conduct more experiments to validate our idea and the effectiveness of DPAM in Appendix C.1 and Appendix C.2.

result, DPAM only keeps unlabeled nodes whose two labels align, i.e., those belong to the intersection of $\tilde{\mathcal{U}}_m$ and $\mathcal{U}_m$ for ~~each $m \in \{1, 2, \cdots, k\}$; and each node in $\tilde{\mathcal{U}}_m \cap \mathcal{U}_m$ gets a pseudo label m.~~

4.2. Node-Reordering

Now DPAM has selected a pool of candidate nodes: $\mathcal{Z} = \bigcup_{i=m}^{k} (\tilde{\mathcal{U}}_m \cap \mathcal{U}_m)$. In this section, we present Node-Reordering, a method that re-orders nodes in $\mathcal{Z}$ according to the closen~~ess of each node to its class center. Node-~~ Reordering combines the geometric rankings from the unsupervised method and confidence rankings from model prediction. We first give the definitions of two rankings.

Definition 4.1. (Geometric Rankings.) Suppose $u \in \tilde{\mathcal{U}}_m \cap \mathcal{U}_m$, and let h_u^U be the embedding of u. We measure the distance between node u and its class center by

$$\delta_u = \mathsf{distance}\left(h_u^U, c_m^{\text{train}}\right) \tag{5}$$

where c_m^{train} is the class center of class m (see (3)). For each class m, we sort nodes in $\tilde{\mathcal{U}}_m \cap \mathcal{U}_m$ in the increasing order of their distance to the class center, so we obtain k sorted lists $\{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k\}$, which we call *geometric rankings*.

Definition 4.2. (Confidence Rankings.) For each node $u \in \tilde{\mathcal{U}}_m \cap \mathcal{U}_m$, we can get a classification *confidence* for the node from the output of the classifier as follow:

$$\mathsf{confidence} = \mathsf{max}\left(\mathsf{softmax}\left(logits\right)\right), \tag{6}$$

Here, $logits$ is the output of the neural network, usually a k (number of classes) dimensional vector. The pseudo-labels of u from the classifier are the index of the class with the highest prediction probability and the corresponding probability is its confidence. We sort nodes in $\tilde{\mathcal{U}}_m \cap \mathcal{U}_m$ in the decreasing order of their confidence, and obtain another k sorted lists $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_k\}$, which we call *confidence rankings*.

Rank Biased Overlap. In the fields of information retrieval and recommendation systems, a fundamental task is to measure the similarity between two rankings. Rank Biased Overlap (RBO) (Webber et al., 2010) compares two ordered lists and returns a numeric value between zero and one to quantify their similarity. An RBO value of zero indicates the lists are completely different, and an RBO of one means completely identical.

Node-Reordering. Section 3.2 has verified that the prediction results of unlabeled nodes given by the bias classifier are not reliable, and Section 4.1 has verified the high-quality of the embeddings. This means that the confidence rankings in early rounds are less unreliable than the geometric rankings.

For each class m, we calculate the RBO value between $\mathcal{S}_m$ and $\mathcal{T}_m$ and then use the RBO score as a weight and get the weighted combination of the two rankings. More specifically, we first compute $r_m = \text{RBO}(\mathcal{S}_m, \mathcal{T}_m)$, and then compute

$$\mathcal{N}_m^{New} = \max\{r_m, 1 - r_m\} \cdot \mathcal{S}_m + \min\{r_m, 1 - r_m\} \cdot \mathcal{T}_m, \tag{7}$$

We then select nodes according to the new ranking based on values in $\mathcal{N}_m^{New}$. Note that we always make the geometric rankings have the dominating influence in this step.

A natural question is why not filter nodes based solely on geometric rankings. UNREAL selects new nodes iteratively for multiple rounds, and as iterations progress, the training set becomes more and more balanced. So in the later stages, the confidence rankings given by the classifier are valuable. The effectiveness of Node-Reordering is empirically verified in Appendix C.4. At the same time, our experiments also show that as iterations progress, the RBO value (similarity) of the two rankings increases (Figure 3(c) and Figure 3(d)).

4.3. Geometric Imbalance

In this section, we consider the issue of Geometric Imbalance (GI) in the embedding space and define a simple and effective metric to measure GI.

Geometric Imbalance. In highly imbalanced scenarios, minority nodes often suffer from topology imbalance (Song et al., 2022; Chen et al., 2021), which means the labeled node stays near the boundary between a minority class and a majority class. The geometric ranking and DPAM introduced above partially alleviate this issue. However, when the class centers of two classes are very close in the embedding space, the problem may still exist. Consider two nodes u and v which belong to class 1 and 2 respectively. When the centers of class 1 and 2 are very close and v is on their boundary in the embedding space, u, v are likely both assigned with pseudo-label 1 and v has a higher geometric ranking than u w.r.t. class 1. We refer to this issue as the geometric imbalance in the embedding space.

Discarding Geometrically Imbalanced Nodes (DGIN). Intuitively, if a node is very close to the centers of more than one class simultaneously, it should not be selected as there is high uncertainty in the pseudo-label. Therefore, we define a simple and natural metric to measure the degree of GI. According to (5), δ_u refers to the distance between the embedding of u and the center of the class to which u is assigned (i.e., the closest class center among all classes). Similarly, we define β_u as the distance between the embedding of u and the second closest center to u. We have $\delta_u \leq \beta_u$ for all u, and intuitively, if $\delta_u \approx \beta_u$, then u is likely to have high degree of GI. We thus define the metric

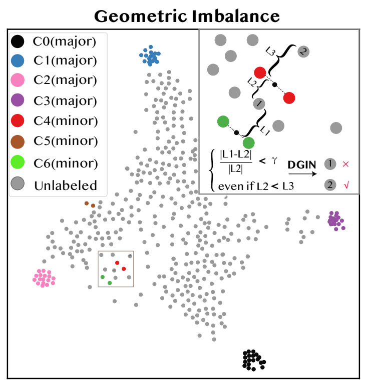


Figure 4. The diagram of GI and DGIN.

for measuring GI as

$$\mathrm{GI}_u = \frac{\beta_u - \delta_u}{\delta_u}. \tag{8}$$

We refer to the metric as the GI index. The GI issue is more serious on the node with a smaller GI index. So we set a threshold and discard all nodes with GI index below the threshold.

The effectiveness of DGIN is empirically verified in Appendix C.5.

4.4. Selecting New Nodes Iteratively

As in standard ST, we select nodes to join the training set in several rounds, and in each round, we retrain the model using the newly formed training set. In highly-imbalanced cases, we only add nodes from minority classes. In this way, the label distribution of the training set is gradually smoothed, and the imbalance issue is alleviated.

5. Experiment

5.1. Experimental Setups

Datasets. We validate the advantages of our method on five benchmark datasets under different imbalance scenarios, in which the step imbalance scheme given in (Zhao et al., 2021; Park et al., 2021; Song et al., 2022) is adopted to construct class imbalanced datasets. More specifically, we choose half of the classes as minority classes and convert randomly picked labeled nodes into unlabeled ones until the imbalance ratio of the training set reaches ρ. For Flickr, in the public split, the training set is already imbalanced, and thus we directly use this split and do not make any changes. For the three citation networks (Cora, CiteSeer, Pubmed), we use the standard splits from Yang et al. (2016) as our initial splits when the imbalance ratio is 10, 20. To create a larger imbalance ratio, 20 labeled nodes per class is not enough, and we use a random split as the initial split for creating an imbalance ratio of 50 and 100. The detailed experimental settings such as evaluation protocol and implementation details of our algorithm are described in Appendix D.

Baselines. We compare UNREAL with several classic techniques (cross-entropy loss with re-weighting (Japkowicz & Stephen, 2002), PC Softmax (Hong et al., 2021) and Balanced Softmax (Ren et al., 2020)) and state-of-the-art methods for imbalanced node classification, including GraphSMOTE (Zhao et al., 2021), GraphENS (Park et al., 2021), ReNode (Chen et al., 2021), and TAM (Song et al., 2022). Among them, GraphSMOTE and GraphENS are representative over-sampling methods for node classification, and ReNode and TAM are loss function modification approaches. For TAM, we test its performances when combined with different base models, including GraphENS, ReNode, and Balanced Softmax, following (Song et al., 2022). The implementation details of baselines are described in Appendix D.5.

5.2. Main Results

Experimental Results Under Different Imbalance Ratios. In Table 1 and Table 3, we report the averaged balanced accuracy (bAcc.) and F1 score with standard errors for the baselines and UNREAL on four class-imbalanced node classification benchmark datasets under different imbalance ratios ($\rho = 10, 20$). The results demonstrate the advantage of UNREAL. Our method consistently outperforms existing state-of-the-art approaches across four datasets, three base models, and two imbalance ratios (except for GraphSAGE on Amazon-Computers with imbalance ratio 10). In many cases the margin is significant. To evaluate the performance on very skewed label distribution, we also test in more imbalanced settings ($\rho = 50, 100$) and present the results in Table 4 and Table 5 of Appendix B.1. Similarly, our method outperforms all other methods consistently and often by a notable margin. We remark that since GraphSMOTE (Zhao et al., 2021) synthesizes nodes within the minority class, it is not applicable when there is only one node in some classes, which is the case when $\rho = 20, 50, 100$ in our setup.

Experimental Results When Unlabeled Data is Imbalanced. We also validate our model on two naturally imbalanced dataset, Flickr ($\rho \approx 10.8$) and Computers-Random ($\rho \approx 17.7$), whose unlabeled data is also imbalanced (See Table 14). The construction of the training set, validation set, and testing set is elaborated on Table D. We found that existing over-sampling methods use too much memory due to synthetic node generation, and cannot handle Flickr on

Table 1. Experimental results of our method UNREAL and other baselines on four class-imbalanced node classification benchmark datasets with $\rho = 10$. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on the GCN architecture.

	Dataset	Cora		CiteSeer		PubMed		Amazon-Computers	
	Imbalance Ratio ($\rho = 10$)	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Vanilla	62.82 ± 1.43	61.67 ± 1.59	38.72 ± 1.88	28.74 ± 3.21	65.64 ± 1.72	56.97 ± 3.17	80.01 ± 0.71	71.56 ± 0.81
	Re-Weight	65.36 ± 1.15	64.97 ± 1.39	44.69 ± 1.78	38.61 ± 2.37	69.06 ± 1.84	64.08 ± 2.97	80.93 ± 1.30	73.99 ± 2.20
	PC Softmax	68.04 ± 0.82	67.84 ± 0.81	50.18 ± 0.55	46.14 ± 0.14	72.46 ± 0.80	70.27 ± 0.94	81.54 ± 0.76	73.30 ± 0.51
	BalancedSoftmax	69.98 ± 0.58	68.68 ± 0.55	55.52 ± 0.97	53.74 ± 1.42	73.73 ± 0.89	71.53 ± 1.06	81.46 ± 0.74	74.31 ± 0.51
	GraphSMOTE	66.39 ± 0.56	65.49 ± 0.93	44.87 ± 1.12	39.20 ± 1.62	67.91 ± 0.64	62.68 ± 1.92	79.48 ± 0.47	72.63 ± 0.76
	Renode	67.03 ± 1.41	67.16 ± 1.67	43.47 ± 2.22	37.52 ± 3.10	71.40 ± 1.42	67.27 ± 2.96	81.89 ± 0.77	73.13 ± 1.60
	GraphENS	70.89 ± 0.71	70.90 ± 0.81	56.57 ± 0.98	55.29 ± 1.33	72.13 ± 1.04	70.72 ± 1.07	82.40 ± 0.39	74.26 ± 1.05
	BalancedSoftmax+TAM	69.94 ± 0.45	69.54 ± 0.47	56.73 ± 0.71	56.15 ± 0.78	74.62 ± 0.97	72.25 ± 1.30	82.36 ± 0.67	72.94 ± 1.43
	Renode+TAM	68.26 ± 1.84	68.11 ± 1.97	46.20 ± 1.17	39.96 ± 2.76	72.63 ± 2.03	68.28 ± 3.30	80.36 ± 1.19	72.51 ± 0.68
	GraphENS+TAM	71.69 ± 0.36	72.14 ± 0.51	58.01 ± 0.68	56.32 ± 1.03	74.14 ± 1.42	72.42 ± 1.39	81.02 ± 0.99	70.78 ± 1.72
	UNREAL	**78.33 ± 1.04**	**76.44 ± 1.06**	**65.63 ± 1.38**	**64.94 ± 1.38**	**75.35 ± 1.41**	**73.65 ± 1.43**	**85.08 ± 0.38**	**75.27 ± 0.23**
	Δ	**+6.64**	**+4.30**	**+7.62**	**+8.62**	**+1.21**	**+1.23**	**+2.68**	**+0.96**
GAT	Vanilla	62.33 ± 1.56	61.82 ± 1.84	38.84 ± 1.13	31.25 ± 1.64	64.60 ± 1.64	55.24 ± 2.80	79.04 ± 1.60	70.00 ± 2.50
	Re-Weight	66.87 ± 0.97	66.62 ± 1.13	45.47 ± 2.35	40.60 ± 2.98	68.10 ± 2.85	63.76 ± 3.54	80.38 ± 0.66	69.99 ± 0.76
	PC Softmax	66.69 ± 0.79	66.04 ± 1.10	50.78 ± 1.66	48.56 ± 2.08	72.88 ± 0.83	71.09 ± 0.89	79.43 ± 0.94	71.33 ± 0.86
	BalancedSoftmax	67.89 ± 0.36	67.96 ± 0.41	54.78 ± 1.25	51.83 ± 2.11	72.30 ± 1.20	69.30 ± 1.79	82.02 ± 1.19	72.94 ± 1.54
	GraphSMOTE	66.71 ± 0.32	65.01 ± 1.21	45.68 ± 0.93	38.96 ± 0.97	67.43 ± 1.23	61.97 ± 2.54	79.38 ± 1.97	69.76 ± 2.31
	Renode	67.33 ± 0.79	68.08 ± 1.16	44.48 ± 2.06	37.93 ± 2.87	69.93 ± 2.10	65.27 ± 2.90	76.01 ± 1.08	66.72 ± 1.42
	GraphENS	70.45 ± 1.25	69.87 ± 1.32	51.45 ± 1.28	47.98 ± 2.08	73.15 ± 1.24	71.90 ± 1.03	81.23 ± 0.74	71.23 ± 0.42
	BalancedSoftmax+TAM	69.16 ± 0.27	69.39 ± 0.37	56.30 ± 1.25	53.87 ± 1.14	73.50 ± 1.24	71.36 ± 1.99	75.54 ± 2.09	66.69 ± 1.44
	Renode+TAM	67.50 ± 0.67	68.06 ± 0.96	45.12 ± 1.41	39.29 ± 1.79	70.66 ± 2.13	66.94 ± 3.54	74.30 ± 1.13	66.13 ± 1.75
	GraphENS+TAM	70.15 ± 0.18	70.00 ± 0.40	56.15 ± 1.13	54.31 ± 1.68	73.45 ± 1.07	72.10 ± 0.36	81.07 ± 1.03	71.27 ± 1.98
	UNREAL	**78.91 ± 0.59**	**75.99 ± 0.47**	**64.10 ± 1.49**	**63.44 ± 1.47**	**74.68 ± 1.43**	**72.78 ± 0.89**	**85.62 ± 0.44**	**75.34 ± 0.99**
	Δ	**+8.46**	**+5.99**	**+7.80**	**+9.13**	**+1.23**	**+0.68**	**+3.60**	**+2.40**
SAGE	Vanilla	61.82 ± 0.97	60.97 ± 1.07	43.18 ± 0.52	36.66 ± 1.25	68.68 ± 1.51	64.16 ± 2.38	72.36 ± 2.39	64.32 ± 2.21
	Re-Weight	63.94 ± 1.07	63.82 ± 1.30	46.17 ± 1.32	40.13 ± 1.68	69.89 ± 1.60	65.71 ± 2.31	76.08 ± 1.14	65.76 ± 1.40
	PC Softmax	65.79 ± 0.70	66.04 ± 0.92	50.66 ± 0.99	47.48 ± 1.66	71.49 ± 0.94	70.23 ± 0.67	74.63 ± 3.01	66.44 ± 4.04
	BalancedSoftmax	67.43 ± 0.61	67.66 ± 0.69	51.74 ± 2.32	49.01 ± 3.16	71.36 ± 1.37	69.66 ± 1.81	73.67 ± 1.11	65.23 ± 2.44
	GraphSMOTE	61.65 ± 0.34	60.97 ± 0.98	42.73 ± 2.87	35.18 ± 1.75	66.63 ± 0.65	61.97 ± 2.54	71.85 ± 0.98	68.92 ± 0.73
	Renode	66.84 ± 1.78	67.08 ± 1.75	48.65 ± 1.37	44.25 ± 2.20	71.37 ± 1.33	67.78 ± 1.38	77.37 ± 0.74	68.42 ± 1.81
	GraphENS	68.74 ± 0.46	68.34 ± 0.33	53.51 ± 0.78	51.42 ± 1.19	70.97 ± 0.78	70.00 ± 1.22	82.57 ± 0.50	71.95 ± 0.51
	BalancedSoftmax+TAM	69.03 ± 0.92	69.03 ± 0.97	51.93 ± 2.19	48.67 ± 3.25	72.28 ± 1.47	71.02 ± 1.31	77.00 ± 2.93	70.85 ± 2.28
	Renode+TAM	67.28 ± 1.11	67.15 ± 1.11	48.39 ± 1.76	43.56 ± 2.31	71.25 ± 1.07	68.69 ± 0.98	74.87 ± 2.25	66.87 ± 2.52
	GraphENS+TAM	70.45 ± 0.74	70.40 ± 0.75	54.69 ± 1.12	53.56 ± 1.86	73.61 ± 1.35	72.50 ± 1.58	82.17 ± 0.93	**72.46 ± 1.00**
	UNREAL	**75.99 ± 0.98**	**73.63 ± 1.23**	**66.45 ± 0.39**	**65.83 ± 0.30**	**74.78 ± 1.30**	**72.80 ± 0.54**	**83.21 ± 1.50**	70.81 ± 1.70
	Δ	**+5.44**	**+3.23**	**+11.76**	**+12.77**	**+1.07**	**+0.30**	**+0.64**	**-1.65**

Table 2. Experimental results of our method UNREAL and other baselines on Computers-Random. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures.

Dataset (Computers-Random)	GCN		GAT		SAGE	
Imbalance Ratio($\rho = 25.50$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
Vanilla	78.43 ± 0.41	77.14 ± 0.39	71.35 ±1.18	69.60 ± 1.11	65.30 ± 1.07	64.77 ± 1.19
Re-Weight	80.49 ± 0.44	75.07 ± 0.58	71.95 ± 0.80	70.67 ± 0.51	66.50 ± 1.47	66.10 ± 1.46
PC Softmax	81.34 ± 0.55	75.17 ± 0.57	70.56 ± 1.46	67.26 ± 1.48	69.73 ± 0.53	67.03 ± 0.6
BalancedSoftmax	81.39 ± 0.25	74.54 ± 0.64	72.09 ± 0.31	68.38 ± 0.69	73.80 ± 1.06	69.74 ± 0.60
GraphSMOTE	80.50 ± 1.11	73.79 ± 0.14	71.98 ± 0.21	67.98 ± 0.31	72.69 ± 0.82	68.73 ± 1.01
Renode	81.64 ± 0.34	76.87 ± 0.32	72.80 ± 0.94	71.40 ± 0.97	70.94 ± 1.50	70.04 ± 1.16
GraphENS	82.66 ± 0.61	76.55 ± 0.17	75.25 ± 0.85	71.49 ± 0.54	77.64 ± 0.52	72.65 ± 0.53
BalancedSoftmax+TAM	81.64 ± 0.48	75.59 ± 0.83	74.00 ± 0.77	70.72 ± 0.50	73.77 ± 1.26	71.03 ± 0.69
Renode+TAM	80.50 ± 1.11	75.79 ± 0.14	71.98 ± 0.21	70.98 ± 0.31	72.69 ± 0.82	70.73 ± 1.01
GraphENS+TAM	82.83 ± 0.68	76.76 ± 0.39	75.81 ± 0.72	72.62 ± 0.57	**78.98 ± 0.60**	**73.59 ± 0.55**
UNREAL	**85.32 ± 0.22**	**80.43 ± 0.56**	**82.52 ± 0.35**	**78.90 ± 0.38**	75.81 ± 1.86	71.86 ± 1.86
Δ	**+2.49**	**+3.97**	**+6.71**	**+6.28**	**-3.17**	**-1.73**

a 3090 GPU with 24GB memory. This include GraphENS (Park et al., 2021), GraphSMOTE (Zhao et al., 2021) and ReNode (Chen et al., 2021). We present the experimental results in Table 2 and Table 6. More importantly, on these two datasets, UNREAL consistently outperforms other approaches.

5.3. More Experiments and Further Analysis

We conduct more experiments on Cora and Amazon-Computers to validate our motivations for DPAM in Appendix C.1. Besides, more novel experiments are targeted at verifying the effectiveness of DPAM in Appendix C.2). More experiments are conducted to investigate the variation of the RBO value (similarity) of the two rankings as iterations progress in Appendix C.3. More ablation studies on Node-Reordering are provided in Appendix C.4. The effectiveness of DGIN is empirically verified in Appendix C.5). We provide sensitivity analysis on the hyperparameter k', which is the number of classes in the K-Means algorithm, and on the threshold γ of DGIN in Appendix C.6. We conduct additional ablation studies to analyze the benefit of each component in our method Appendix C.7.

6. Related work

Imbalanced Learning Most real-world data is naturally imbalanced. The major challenge in imbalanced scenarios is how to train a fair model which does not biased toward the majority classes. There are several commonly used approaches for alleviating this problem. Ensemble learning (Freund & Schapire, 1997; Liu et al., 2008; Zhou et al., 2020; Wang et al., 2020a; Liu et al., 2020; Cai et al., 2021) combines the results of multiple weak classifiers. Data resampling methods (Chawla et al., 2002; Han et al., 2005; Smith et al., 2014; Sáez et al., 2015; Kang et al., 2019; Wang et al., 2021a) smooth the label distribution in the training set by synthesizing or duplicating minority class samples. A third class of approaches alleviates the imbalance problem by modifying the loss function, which gives larger weights to minority classes or changes the margins of different classes (Zhou & Liu, 2005; Tang et al., 2008; Cao et al., 2019; Tang et al., 2020; Xu et al., 2020; Ren et al., 2020; Wang et al., 2021b). Methods based on post-hoc correction compensate minority classes during the inference step, after model training is complete (Kang et al., 2019; Tian et al., 2020; Menon et al., 2020; Hong et al., 2021). Although these techniques have been widely applied on the i.i.d. data, it is not a trivial task to extend them to graph-structured data.

Imbalanced Learning in Node Classification Recently, a series of research (Shi et al., 2020; Wang et al., 2020c; Zhao et al., 2021; Liu et al., 2021; Qu et al., 2021; Chen et al., 2021; Park et al., 2021; Song et al., 2022) explicitly tackle the challenges brought by the topological structures of graph data when handling imbalanced node classification. GraphSMOTE (Zhao et al., 2021) synthesizes minority nodes in embedding space by interpolating two minority nodes using the SMOTE (Chawla et al., 2002) algorithm and infers the neighborhoods of new nodes with link prediction algorithms. ImGAGN (Qu et al., 2021) generates the features of minority nodes with all of the minority nodes according to the learned weight matrix and synthesizes the neighborhoods of new nodes based on weights. (Qu et al., 2021) only consider binary classification, and it is computationally expensive to build a generator for each class on multi-classification tasks. GraphENS (Park et al., 2021) works for multi-class node classification, which synthesizes the whole ego network for minority nodes by interpolating the ego networks of two nodes based on their similarity. (Chen et al., 2021) identifies topology imbalance as a main source of difficulty when handling imbalance on node classification tasks; they propose ReNode, which mitigates topology imbalance by adjusting the weights of nodes according to their distance to class boundaries. TAM (Song et al., 2022) adjusts the scores of different classes in the Softmax function based on local topology and label statistics. To obtain label information of unlabeled nodes, TAM trains the model using the original imbalanced training set and takes the model predictions as proxies for ground-truth labels.

Pseudo-labeling Methods in GNNs Recent studies have just focused on leveraging pseudo-labeling techniques to train GNNs given limited labeled information. Co-training (Li et al., 2018), in particular, uses Parwalks (Wu et al., 2012) to provide confident pseudolabels to help train GNNs, whereas self-training (Li et al., 2018) expands the label set by obtaining pseudo-labels provided by previously trained GNNs. Furthermore, M3S (Sun et al., 2020) employs a clustering technique to filter out pseudo-labels that do not match the clustering assignments, thereby improving pseudo-labeling accuracy.

7. Conclusion

In this work, we observe that selecting unlabeled nodes instead of generating synthetic nodes in oversampling-based methods for imbalanced node classification is much simpler and more effective. We propose a novel iterative unlabeled node selection and retraining framework, which effectively selects high-quality new samples from the unlabeled sets to smooth the label distribution of the training set. Moreover, we propose to exploit the geometric structure in the node embedding space to compensate for the bias in the model predictions. Extensive experimental results show that UNREAL consistently outperforms existing state-of-the-art approaches by large margins.

References

Cai, J., Wang, Y., and Hwang, J.-N. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 112–121, 2021.

Cao, K., Wei, C., Gaidon, A., Arechiga, N., and Ma, T. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

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

Freund, Y. and Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

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

Han, H., Wang, W.-Y., and Mao, B.-H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pp. 878–887. Springer, 2005.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Hong, Y., Han, S., Choi, K., Seo, S., Kim, B., and Chang, B. Disentangling label distribution for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6626–6636, 2021.

Japkowicz, N. and Stephen, S. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

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

Liu, X.-Y., Wu, J., and Zhou, Z.-H. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., and He, Q. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pp. 3168–3177, 2021.

Liu, Z., Chen, C., Yang, X., Zhou, J., Li, X., and Song, L. Heterogeneous graph neural networks for malicious account detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 2077–2085, 2018.

Liu, Z., Wei, P., Jiang, J., Cao, W., Bian, J., and Chang, Y. Mesa: boost ensemble imbalanced learning with meta-sampler. *Advances in Neural Information Processing Systems*, 33:14463–14474, 2020.

Menon, A. K., Jayasumana, S., Rawat, A. S., Jain, H., Veit, A., and Kumar, S. Long-tail learning via logit adjustment. *arXiv preprint arXiv:2007.07314*, 2020.

Mohammadrezaei, M., Shiri, M. E., and Rahmani, A. M. Identifying fake accounts on social networks based on graph analysis and classification algorithms. *Security and Communication Networks*, 2018, 2018.

Park, J., Song, J., and Yang, E. Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *International Conference on Learning Representations*, 2021.

Qu, L., Zhu, H., Zheng, R., Shi, Y., and Yin, H. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1390–1398, 2021.

Ren, J., Yu, C., Ma, X., Zhao, H., Yi, S., et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33: 4175–4186, 2020.

Sáez, J. A., Luengo, J., Stefanowski, J., and Herrera, F. Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a resampling method with filtering. *Information Sciences*, 291:184–203, 2015.

Shi, M., Tang, Y., Zhu, X., Wilson, D., and Liu, J. Multi-class imbalanced graph convolutional network learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2020.

Smith, M. R., Martinez, T., and Giraud-Carrier, C. An instance level analysis of data complexity. *Machine learning*, 95(2):225–256, 2014.

Song, J., Park, J., and Yang, E. Tam: Topology-aware margin loss for class-imbalanced node classification. In *International Conference on Machine Learning*, pp. 20369–20383. PMLR, 2022.

Sun, K., Lin, Z., and Zhu, Z. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5892–5899, 2020.

Tang, K., Huang, J., and Zhang, H. Long-tailed classification by keeping the good and removing the bad momentum causal effect. *Advances in Neural Information Processing Systems*, 33:1513–1524, 2020.

Tang, Y., Zhang, Y.-Q., Chawla, N. V., and Krasser, S. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(1):281–288, 2008.

Tian, J., Liu, Y.-C., Glaser, N., Hsu, Y.-C., and Kira, Z. Posterior re-calibration for imbalanced datasets. *Advances in Neural Information Processing Systems*, 33:8101–8113, 2020.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, J., Lukasiewicz, T., Hu, X., Cai, J., and Xu, Z. Rsg: A simple but effective module for learning imbalanced datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3784–3793, 2021a.

Wang, T., Zhu, Y., Zhao, C., Zeng, W., Wang, J., and Tang, M. Adaptive class suppression loss for long-tail object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3103–3112, 2021b.

Wang, X., Lian, L., Miao, Z., Liu, Z., and Yu, S. X. Long-tailed recognition by routing diverse distribution-aware experts. *arXiv preprint arXiv:2010.01809*, 2020a.

Wang, X., Liu, H., Shi, C., and Yang, C. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, 34:23768–23779, 2021c.

Wang, Z., Wang, Q., Zhu, T., and Ye, X. Extending line for network embedding with completely imbalanced labels. *International Journal of Data Warehousing and Mining (IJDWM)*, 16(3):20–36, 2020b.

Wang, Z., Ye, X., Wang, C., Cui, J., and Philip, S. Y. Network embedding with completely-imbalanced labels. *IEEE Transactions on Knowledge and Data Engineering*, 33(11):3634–3647, 2020c.

Webber, W., Moffat, A., and Zobel, J. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

Wu, X.-M., Li, Z., So, A., Wright, J., and Chang, S.-F. Learning with partially absorbing random walks. *Advances in neural information processing systems*, 25, 2012.

Xu, Z., Dan, C., Khim, J., and Ravikumar, P. Class-weighted classification: Trade-offs and robust approaches. In *International Conference on Machine Learning*, pp. 10544–10554. PMLR, 2020.

Yang, Y. and Xu, Z. Rethinking the value of labels for improving class-imbalanced learning. *Advances in neural information processing systems*, 33:19290–19301, 2020.

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

Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pp. 189–196, 1995.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., and Prasanna, V. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Zhao, T., Zhang, X., and Wang, S. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 833–841, 2021.

Zhou, B., Cui, Q., Wei, X.-S., and Chen, Z.-M. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9719–9728, 2020.

Zhou, Z., Shi, J., Zhang, S., Huang, Z., and Li, Q. Effective semi-supervised node classification on few-labeled graph data. *arXiv preprint arXiv:1910.02684*, 2019.

Zhou, Z.-H. and Liu, X.-Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on knowledge and data engineering*, 18(1):63–77, 2005.

Supplementary Material

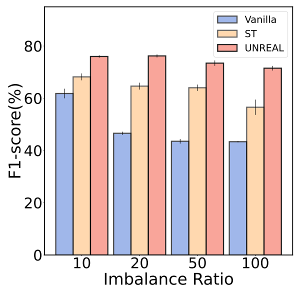
A. More Experiments About Imbalanced Learning with Unlabled Data (Section 3)

We provide complete evaluation results on more benchmark datasets for Section 3.1 and Section 3.2, where more basic models are included in addition to the reported results in the main paper.

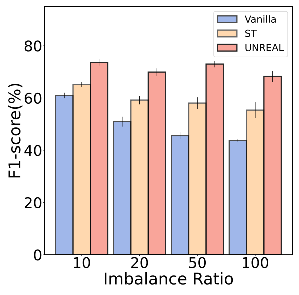
A.1. More Experiments for ST Improves the Performance of GNN in Imbalanced Learning (Section 3.1)

Details of Experimental Setup. We chose the three citation datasets, Cora, CiteSeer, and PubMed, to build scenarios with varying degrees of imbalance. To be more specific, we select half of the classes as minority classes and convert randomly selected labeled nodes into unlabeled nodes until the training set's imbalance ratio reaches ρ. We fix architecture as the 2-layer GNN (i.e. GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), GraphSAGE (Hamilton et al., 2017)) having 128 hidden dimensions and train models for 2000 epochs. For the ST method, the size of added nodes for each class is a hyperparameter that we tune based on the validation set's accuracy. We repeat each experiment five times and report the average experiment results under different imbalance ratios in Figure 5.

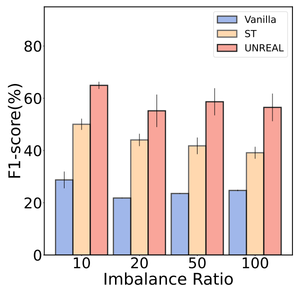
Analysis. More results are presented to confirm the effectiveness of ST in boosting imbalanced learning on Cora, CiteSeer, and PubMed using three GNN architectures in Figure 5. It can be observed that, across different ratios and data sets, ST consistently outperforms the vanilla model by a wide margin, confirming the positive value of unlabeled nodes. More notably, we can discover that ST performs gradually poorly in heavily imbalanced scenarios, especially for Cora and CiteSeer. In this work, we argue that for highly imbalanced data, ST is unlikely to achieve optimal performance because classifiers' biased and untrustworthy predictions may introduce low-quality nodes into the training set early on.



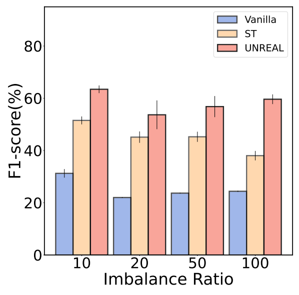
(a) Cora-GAT



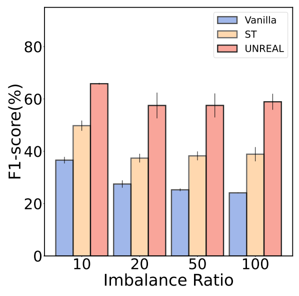
(b) Cora-SAGE



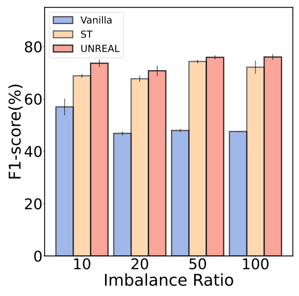
(c) CiteSeer-GCN



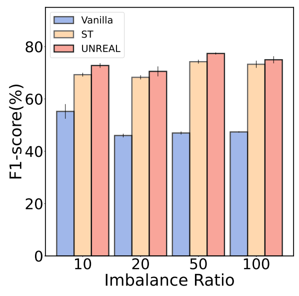
(d) CiteSeer-GAT



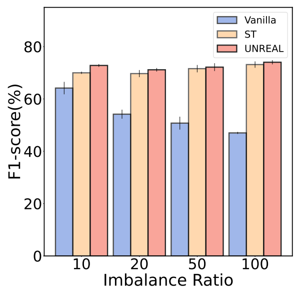
(e) CiteSeer-SAGE



(f) PubMed-GCN

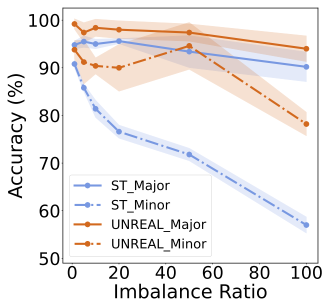


(g) PubMed-GAT

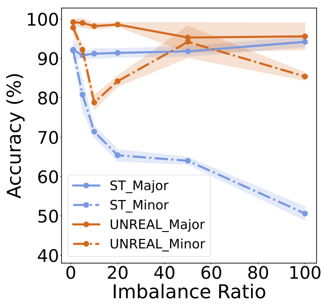


(h) PubMed-SAGE

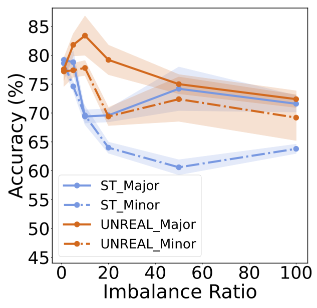
Figure 5. The experimental results on the three citation datasets under different imbalance scenarios (ρ = 10, 20, 50, 100). We report the F1-score (%) with the standard errors of Vanilla, ST, and UNREAL.



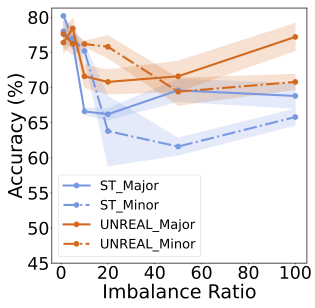
(a) Cora-GAT



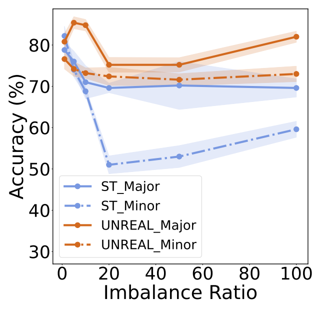
(b) Cora-SAGE



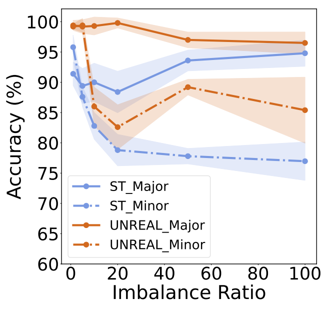
(c) CiteSeer-GCN



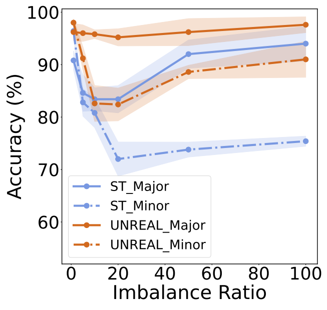
(d) CiteSeer-GAT



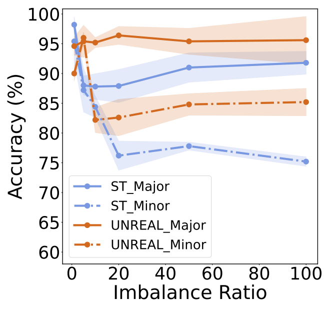
(e) CiteSeer-SAGE



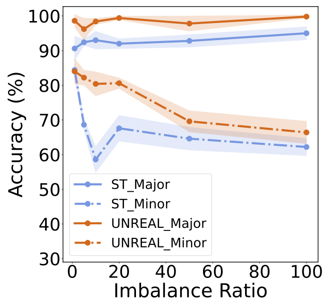
(f) PubMed-GCN



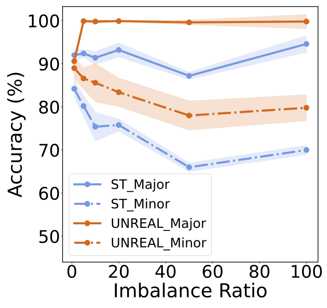
(g) PubMed-GAT



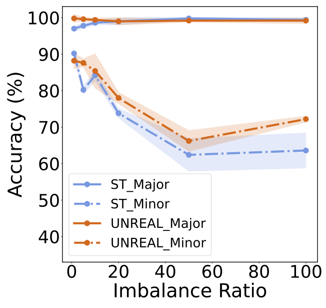
(h) PubMed-SAGE



(i) Computers-GCN



(j) Computers-GAT



(k) Computers-SAGE

Figure 6. Here, we present the experimental results from four benchmark datasets under various imbalance scenarios. We select top 100 unlabeled nodes newly added to the training set via ST & UNREAL, and evaluate the performance of ST & UNREAL based on three GNN architectures by testing the accuracy with the standard errors of these nodes' pseudo labels. Minor means that we only test unlabeled nodes which are selected into the minority classes, and Major means that we only test unlabeled nodes which are selected into the majority classes.

A.2. More Experiments for Pseudo-label Misjudgment Augmentation Problem in Imbalanced Learning (Section 3.2)

Details of Experimental Setup. Since true labels for all benchmark nodes are provided, we first conduct experiments to test the accuracy of pseudo labels for unlabeled nodes on class-imbalanced graphs inventively. We select top 100 unlabeled nodes newly added to the training set through ST & UNREAL, and evaluate the performance of ST & UNREAL by testing the accuracy $(\%)$ with the standard errors of these nodes' pseudo labels. We test unlabeled nodes that are selected into the minority classes and unlabeled nodes that are selected into the majority classes separately. We evaluate the performance of each method on Cora, CiteSeer, PubMed, Amazon-Computers under different imbalance scenarios. We process the datasets with a traditional imbalanced distribution following Zhao et al. (2021); Park et al. (2021); Song et al. (2022). The imbalance ratio ρ between the numbers of the most frequent class and the least frequent class is set as 1 (balanced), 5, 10, 20, 50, 100. We fix architecture as the 2-layer GNN (i.e. GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), GraphSAGE (Hamilton et al., 2017)) having 128 hidden dimensions and train models for 2000 epochs. The validation accuracy is used to select the model. Each experiment is repeated five times, and the average experiment results are reported in Figure 6.

Analysis. As shown in Figure 6, in different imbalanced scenarios for ST, the accuracy of the pseudo labels for the unlabeled nodes selected into the minority classes and the majority classes of the training set are reported. We can see that as ρ increases, the accuracy of pseudo labels for unlabeled nodes selected into minority classes decreases, implying that the influence of the classifier's bias increases. The end results demonstrate a number of intriguing aspects. (1) This means that the classifier's pseudo-labels are not credible in a highly imbalanced scenario. More seriously, we want to add unlabeled nodes whose pseudo-labels are minority classes to the training set, which will cause ST to add excessive noise during the training process. (2) It's noteworthy that to evaluate the performance of ST, we only focus on the top 100 nodes which are pick out based on Confidence Rankings (Section 4.2) from the classifier. So, we believe that even if a node's pseudo-label is correct, the classifier's confidence is skewed, which means that we may include low-quality unlabeled nodes in the training set while ignoring high-quality unlabeled nodes. This, we believe, is the primary factor of the ST's poor performance in imbalanced scenarios. (3) More importantly, regardless of selecting majority class nodes or minority class nodes, UNREAL consistently outperforms ST. Looking back at Figure 5, it is clear that UNREAL consistently outperforms ST by a wide margin, and as the imbalance ratio increases, so does the gap in F1 scores between ST and our method.

B. More Results of Main Experiments (Section 5)

B.1. More Results in Heavily-imbalanced Scenarios

Due to space limitations, we present the complete experimental results here. In Table 3, Table 4, and Table 5, we report the results in heavily imbalanced scenarios (ρ = 20, 50, 100).

Table 3. Experimental results of our method UNREAL and other baselines on four class-imbalanced node classification benchmark datasets with $\rho = 20$. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures.

	Dataset	Cora		CiteSeer		PubMed		Amazon-Computers	
	Imbalance Ratio ($\rho = 20$)	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Vanilla	53.20 ± 0.88	47.81 ± 1.23	35.32 ± 0.15	21.81 ± 0.12	61.13 ± 0.35	46.85 ± 0.76	72.34 ± 2.92	65.42 ± 3.00
	Re-Weight	57.51 ± 1.05	54.63 ± 1.08	36.99 ± 1.79	27.33 ± 2.32	66.52 ± 2.42	58.22 ± 3.65	72.45 ± 2.06	65.85 ± 1.46
	PC Softmax	61.74 ± 1.50	60.55 ± 1.97	42.53 ± 1.53	36.54 ± 1.13	68.26 ± 1.99	66.54 ± 1.87	73.84 ± 2.64	66.32 ± 2.97
	BalancedSoftmax	64.06 ± 0.74	62.88 ± 0.86	47.29 ± 1.29	44.08 ± 1.71	69.71 ± 1.74	68.31 ± 1.71	76.92 ± 2.01	69.86 ± 1.99
	Renode	59.40 ± 1.00	56.88 ± 1.52	38.25 ± 1.60	27.61 ± 2.25	67.45 ± 3.34	60.40 ± 5.74	74.15 ± 1.72	67.27 ± 0.92
	GraphENS	67.30 ± 1.45	66.82 ± 1.40	46.39 ± 3.48	42.38 ± 4.14	71.37 ± 1.77	69.37 ± 1.69	75.41 ± 1.75	69.32 ± 1.58
	BalancedSoftmax+TAM	64.75 ± 0.54	63.46 ± 0.72	48.52 ± 1.62	46.38 ± 1.79	69.95 ± 2.09	68.90 ± 1.86	77.09 ± 2.02	69.86 ± 1.76
	Renode+TAM	59.88 ± 1.16	58.05 ± 1.66	41.11 ± 2.45	31.58 ± 2.62	68.53 ± 3.53	64.82 ± 4.32	73.46 ± 1.77	67.50 ± 1.18
	GraphENS+TAM	66.94 ± 1.38	66.67 ± 1.42	48.80 ± 2.98	45.06 ± 4.16	71.92 ± 1.58	69.35 ± 1.88	75.78 ± 1.57	68.58 ± 1.78
	UNREAL	**77.02 ± 0.75**	**74.15 ± 0.87**	**55.81 ± 6.11**	**55.19 ± 6.23**	**73.06 ± 1.87**	**70.77 ± 1.96**	**85.69 ± 0.11**	**74.81 ± 0.68**
	Δ	**+9.72**	**+7.33**	**+7.01**	**+8.81**	**+1.14**	**+1.40**	**+8.60**	**+4.95**
GAT	Vanilla	51.51 ± 0.53	46.59 ± 0.61	34.74 ± 0.16	22.00 ± 0.15	60.22 ± 0.47	46.03 ± 0.70	68.09 ± 2.96	60.08 ± 2.76
	Re-Weight	58.68 ± 3.44	55.98 ± 3.97	36.78 ± 0.94	26.63 ± 1.61	63.47 ± 1.73	54.63 ± 3.25	71.44 ± 2.42	62.86 ± 1.94
	PC Softmax	59.62 ± 1.41	58.77 ± 1.95	43.38 ± 2.01	37.76 ± 2.12	70.81 ± 1.41	70.25 ± 1.30	71.16 ± 1.15	62.26 ± 0.87
	BalancedSoftmax	62.05 ± 1.62	61.14 ± 1.71	47.89 ± 1.25	44.84 ± 1.35	69.91 ± 1.68	67.43 ± 1.73	72.91 ± 1.93	62.79 ± 0.98
	Renode	59.52 ± 2.28	57.16 ± 2.47	37.21 ± 2.01	27.09 ± 3.17	64.56 ± 1.65	55.87 ± 2.83	69.34 ± 2.35	59.02 ± 1.67
	GraphENS	64.52 ± 2.05	62.52 ± 1.84	43.74 ± 3.81	37.47 ± 4.21	69.00 ± 2.67	65.54 ± 3.54	71.78 ± 2.30	61.83 ± 1.75
	BalancedSoftmax+TAM	63.30 ± 0.99	62.81 ± 1.18	49.34 ± 1.29	46.92 ± 1.39	71.17 ± 2.09	68.85 ± 2.90	65.59 ± 2.86	58.12 ± 1.22
	Renode+TAM	61.32 ± 2.18	59.19 ± 2.64	39.85 ± 2.20	30.63 ± 2.63	66.28 ± 3.24	58.99 ± 3.04	65.81 ± 2.57	56.73 ± 1.62
	GraphENS+TAM	65.78 ± 1.62	63.80 ± 1.79	44.81 ± 2.66	39.47 ± 3.54	70.33 ± 2.33	67.00 ± 3.25	73.55 ± 2.04	64.03 ± 1.32
	UNREAL	**79.10 ± 0.71**	**76.21 ± 0.58**	**55.11 ± 5.00**	**53.67 ± 5.51**	**72.54 ± 1.52**	**70.54 ± 1.91**	**83.19 ± 0.66**	**74.39 ± 0.89**
	Δ	**+13.22**	**+12.41**	**+6.75**	**+8.81**	**+1.37**	**+1.69**	**+9.64**	**+10.36**
SAGE	Vanilla	54.61 ± 1.21	50.95 ± 1.90	37.36 ± 1.03	27.49 ± 1.41	62.04 ± 1.34	54.18 ± 1.73	62.70 ± 2.87	55.39 ± 2.69
	Re-Weight	57.37 ± 0.61	55.30 ± 0.72	37.69 ± 1.20	27.92 ± 2.01	65.01 ± 2.69	58.34 ± 2.19	68.31 ± 2.06	60.45 ± 2.40
	PC Softmax	59.25 ± 0.74	58.55 ± 0.81	42.77 ± 1.82	40.08 ± 1.82	70.55 ± 1.19	67.60 ± 1.59	70.57 ± 2.86	62.73 ± 2.69
	BalancedSoftmax	61.93 ± 1.26	60.89 ± 1.36	43.64 ± 1.33	38.31 ± 1.13	69.89 ± 1.40	68.12 ± 0.78	68.45 ± 2.92	62.12 ± 3.10
	Renode	58.48 ± 0.97	55.39 ± 0.94	40.65 ± 2.36	31.78 ± 3.24	66.50 ± 2.63	58.72 ± 4.16	68.36 ± 1.54	61.60 ± 2.00
	GraphENS	63.54 ± 0.91	62.20 ± 0.87	44.89 ± 2.51	40.48 ± 2.94	71.37 ± 1.77	69.37 ± 1.69	75.47 ± 2.20	67.49 ± 1.65
	BalancedSoftmax+TAM	64.16 ± 0.94	63.63 ± 1.10	44.32 ± 2.36	40.17 ± 2.06	70.06 ± 1.46	69.54 ± 1.35	66.10 ± 2.37	59.22 ± 2.48
	Renode+TAM	59.77 ± 2.20	57.98 ± 2.79	42.50 ± 0.93	35.11 ± 1.84	67.31 ± 2.73	60.63 ± 3.49	66.42 ± 2.32	58.62 ± 1.95
	GraphENS+TAM	63.39 ± 1.36	61.66 ± 1.53	45.92 ± 1.96	41.97 ± 2.50	69.62 ± 2.57	66.85 ± 3.00	75.75 ± 2.30	68.86 ± 1.29
	UNREAL	**73.10 ± 1.60**	**69.92 ± 1.43**	**58.35 ± 4.58**	**57.51 ± 4.92**	**73.67 ± 0.58**	**71.15 ± 0.67**	**78.88 ± 2.16**	**69.00 ± 1.42**
	Δ	**+8.94**	**+5.69**	**+12.43**	**+15.54**	**+2.30**	**+1.61**	**+3.13**	**+0.14**

Analysis. In Table 1, When ρ is 10, UNREAL outperforms other baselines by a wide margin, especially for the datasets Cora and CiteSeer. For the datasets PubMed and Amazon-Computers, UNREAL can still achieve cutting-edge results in the majority of cases. However, the improvement is not substantial. This, we believe, is due to the following factors: (1) For PubMed, because it only has three types of nodes, there is only one minor category in our settings, which means that the imbalanced scenario is not typical. So in such a mild imbalance setting, each method can still achieve good results. Nonetheless, UNREAL can still achieve the most advanced performance. (2) For Amazon-Computers, based on prior experience, we discovered that various models can achieve a good classification effect on this dataset, even if the label setting is sparse, which is related to the nature of the dataset itself.

In Table 3, Table 4, and Table 5, the results demonstrate several intriguing aspects: (1) The performance gap between other methods and UNREAL is growing, which means that current methods perform poorly in heavily imbalanced scenarios, whereas UNREAL can still achieve stable performance. We believe this is because high-quality unlabeled nodes are compensated into the training set. (2) Existing oversampling methods, such as GraphENS (Park et al., 2021), which achieves state-of-the-art experimental performance in a slightly imbalanced setting, perform extremely poorly in heavily imbalanced scenarios. We contend that the paradigm of synthesizing virtual nodes and local topologies always introduces a significant amount of redundancy into model training. (3) It is noteworthy that BalanceSoftmax (Ren et al., 2020) achieves superior results in highly imbalanced scenes. BalancedSoftmax avoids estimation bias caused by label distribution migration in general.

Table 4. Experimental results of our method UNREAL and other baselines on four class-imbalanced node classification benchmark datasets with $\rho = 50$. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures.

		Cora		CiteSeer		PubMed		Amazon-Computers	
	Imbalance Ratio ($\rho = 50$)	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Vanilla	51.81 ± 0.62	43.98 ± 1.00	37.59 ± 0.17	23.54 ± 0.13	61.65 ± 0.34	47.95 ± 0.58	77.36 ± 3.41	69.68 ± 3.12
	Re-Weight	58.54 ± 2.39	54.13 ± 3.20	38.19 ± 1.28	27.43 ± 2.34	65.70 ± 1.59	56.35 ± 4.26	79.10 ± 2.44	71.40 ± 2.86
	PC Softmax	64.87 ± 2.23	62.01 ± 3.14	42.42 ± 2.19	38.83 ± 2.70	69.21 ± 0.59	69.40 ± 0.87	81.90 ± 1.63	74.34 ± 2.13
	BalancedSoftmax	65.94 ± 1.55	64.00 ± 2.05	47.62 ± 1.11	46.55 ± 1.46	70.40 ± 1.00	69.04 ± 0.66	82.97 ± 0.83	73.74 ± 1.27
	Renode	62.22 ± 1.76	61.18 ± 2.24	41.23 ± 1.66	33.66 ± 2.69	68.67 ± 1.21	63.05 ± 1.47	81.71 ± 0.99	72.55 ± 1.61
	GraphENS	63.47 ± 0.98	62.21 ± 1.65	48.17 ± 1.58	41.07 ± 2.34	69.63 ± 2.55	64.30 ± 3.51	81.63 ± 2.35	72.57 ± 2.33
	BalancedSoftmax+TAM	68.57 ± 1.58	67.25 ± 1.27	53.43 ± 2.42	51.74 ± 2.80	77.20 ± 1.45	74.86 ± 0.99	81.74 ± 2.30	73.85 ± 2.68
	Renode+TAM	63.93 ± 1.96	61.64 ± 2.71	48.17 ± 1.58	41.07 ± 2.34	69.63 ± 2.55	64.30 ± 3.51	80.55 ± 1.75	72.33 ± 1.63
	GraphENS+TAM	65.05 ± 1.11	62.11 ± 1.98	45.03 ± 1.34	42.65 ± 1.94	69.74 ± 0.78	70.82 ± 0.63	81.69 ± 2.22	72.09 ± 1.75
	UNREAL	**75.62 ± 2.02**	**72.59 ± 2.13**	**59.97 ± 4.59**	**58.66 ± 5.20**	**78.55 ± 0.84**	**75.91 ± 0.81**	**85.54 ± 0.26**	**75.76 ± 0.13**
	Δ	**+7.05**	**+5.34**	**+6.54**	**+6.92**	**+1.35**	**+1.06**	**+2.57**	**+1.91**
GAT	Vanilla	53.90 ± 0.63	45.53 ± 0.89	36.48 ± 0.08	23.68 ± 0.16	60.16 ± 0.47	46.99 ± 0.58	72.42 ± 2.17	64.41 ± 2.68
	Re-Weight	59.78 ± 1.92	56.69 ± 2.21	38.70 ± 2.23	29.38 ± 3.06	66.27 ± 0.68	57.34 ± 1.41	73.46 ± 3.07	67.00 ± 2.60
	PC Softmax	59.44 ± 2.62	58.06 ± 2.69	43.13 ± 1.56	37.04 ± 2.07	70.86 ± 0.44	70.96 ± 0.54	77.21 ± 2.90	69.17 ± 2.89
	BalancedSoftmax	64.71 ± 2.28	62.55 ± 2.61	51.89 ± 1.15	49.36 ± 1.52	70.94 ± 1.09	70.33 ± 0.99	77.49 ± 1.58	70.44 ± 2.33
	Renode	63.81 ± 1.72	60.63 ± 2.26	41.60 ± 2.30	33.94 ± 4.60	70.35 ± 1.26	67.43 ± 0.01	72.39 ± 2.75	65.23 ± 3.35
	GraphENS	64.52 ± 2.51	61.41 ± 3.15	45.23 ± 2.97	41.12 ± 4.23	69.66 ± 1.01	66.83 ± 0.94	78.36 ± 2.74	70.44 ± 2.51
	BalancedSoftmax+TAM	68.05 ± 1.03	66.07 ± 1.14	54.28 ± 0.79	52.77 ± 0.97	75.65 ± 1.11	74.02 ± 1.44	78.86 ± 1.53	70.71 ± 2.04
	Renode+TAM	64.40 ± 1.83	63.48 ± 2.83	43.54 ± 1.54	35.80 ± 2.43	71.23 ± 2.04	66.61 ± 4.31	76.07 ± 2.70	68.43 ± 2.68
	GraphENS+TAM	65.33 ± 2.67	65.34 ± 2.53	48.00 ± 1.46	48.14 ± 1.43	71.50 ± 1.26	72.58 ± 1.07	80.02 ± 2.32	72.38 ± 2.47
	UNREAL	**77.07 ± 0.83**	**73.44 ± 1.05**	**57.70 ± 4.35**	**56.81 ± 4.67**	**79.41 ± 0.29**	**77.38 ± 0.39**	**86.06 ± 0.45**	**77.55 ± 0.71**
	Δ	**+9.02**	**+7.37**	**+3.42**	**+4.04**	**+3.76**	**+3.36**	**+6.04**	**+5.17**
SAGE	Vanilla	53.02 ± 0.83	45.58 ± 1.30	38.81 ± 0.89	25.28 ± 0.51	61.41 ± 1.01	50.46 ± 2.47	56.53 ± 2.12	48.52 ± 2.75
	Re-Weight	58.03 ± 0.81	54.32 ± 0.99	38.49 ± 1.34	30.41 ± 1.82	62.41 ± 0.90	51.37 ± 2.62	70.36 ± 2.21	61.52 ± 2.73
	PC Softmax	62.33 ± 1.62	59.97 ± 1.98	41.79 ± 1.19	36.90 ± 0.84	69.58 ± 1.09	67.13 ± 0.95	73.53 ± 2.02	66.12 ± 3.19
	BalancedSoftmax	64.57 ± 0.77	62.22 ± 0.82	41.84 ± 1.72	40.09 ± 1.04	70.43 ± 0.38	68.99 ± 0.99	73.27 ± 2.30	68.30 ± 1.97
	Renode	61.35 ± 1.86	58.88 ± 2.53	40.37 ± 2.33	32.57 ± 3.62	67.54 ± 3.05	59.77 ± 5.30	70.46 ± 3.45	62.30 ± 4.40
	GraphENS	63.95 ± 0.96	62.63 ± 2.12	41.99 ± 1.54	37.44 ± 2.43	66.07 ± 1.12	61.63 ± 1.82	76.21 ± 2.84	68.10 ± 2.56
	BalancedSoftmax+TAM	65.97 ± 0.71	65.53 ± 0.88	52.89 ± 1.65	49.92 ± 1.83	71.11 ± 0.75	71.73 ± 0.79	73.12 ± 1.41	66.45 ± 1.04
	Renode+TAM	62.79 ± 0.47	61.05 ± 0.82	43.04 ± 1.30	36.97 ± 1.92	71.79 ± 1.33	67.80 ± 2.45	74.55 ± 2.95	66.06 ± 2.16
	GraphENS+TAM	65.98 ± 1.37	64.84 ± 1.13	49.54 ± 1.79	49.48 ± 1.70	73.24 ± 1.32	73.73 ± 1.14	80.75 ± 1.22	72.31 ± 0.95
	UNREAL	**76.04 ± 1.30**	**72.99 ± 1.25**	**58.70 ± 4.10**	**57.53 ± 4.59**	**75.27 ± 1.26**	**72.16 ± 1.50**	**82.03 ± 0.77**	**72.98 ± 0.52**
	Δ	**+10.06**	**+7.46**	**+5.81**	**+7.61**	**+2.03**	**-1.57**	**+1.28**	**+0.67**

Table 5. Experimental results of our method UNREAL and other baselines on four class-imbalanced node classification benchmark datasets with $\rho = 100$. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures.

		Cora		CiteSeer		PubMed		Amazon-Computers	
	Imbalance Ratio ($\rho = 100$)	bAcc.	F1	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Vanilla	51.62 ± 0.20	43.91 ± 0.25	38.83 ± 0.26	24.71 ± 0.25	61.28 ± 0.12	47.55 ± 0.16	76.09 ± 3.79	69.32 ± 3.49
	Re-Weight	59.11 ± 1.06	54.04 ± 1.36	42.67 ± 2.06	33.17 ± 3.40	67.14 ± 2.71	55.24 ± 5.36	81.53 ± 2.20	71.45 ± 2.05
	PC Softmax	63.75 ± 1.02	61.19 ± 1.43	38.34 ± 0.71	33.65 ± 1.42	70.85 ± 0.44	70.26 ± 0.63	82.22 ± 1.99	72.38 ± 2.52
	BalancedSoftmax	63.03 ± 1.57	61.28 ± 1.77	48.49 ± 1.20	46.59 ± 1.34	70.77 ± 1.88	68.88 ± 1.74	83.33 ± 3.35	74.34 ± 2.74
	Renode	60.76 ± 2.53	58.09 ± 3.00	43.41 ± 2.07	33.69 ± 2.76	67.63 ± 2.77	61.70 ± 4.84	82.13 ± 1.73	71.79 ± 1.85
	GraphENS	63.00 ± 1.30	62.33 ± 1.67	45.99 ± 2.06	37.23 ± 3.40	68.65 ± 1.00	62.17 ± 1.60	83.37 ± 2.17	73.96 ± 1.98
	BalancedSoftmax+TAM	69.44 ± 0.59	67.10 ± 0.88	52.60 ± 0.69	51.21 ± 0.84	73.73 ± 1.10	73.72 ± 0.83	83.70 ± 2.17	75.39 ± 1.43
	Renode+TAM	64.19 ± 1.46	60.90 ± 1.56	44.78 ± 1.51	35.90 ± 2.61	70.53 ± 0.75	64.35 ± 1.79	82.32 ± 2.19	73.09 ± 1.75
	GraphENS+TAM	60.40 ± 4.42	57.77 ± 4.02	42.72 ± 2.54	39.40 ± 2.57	70.73 ± 1.96	72.50 ± 1.87	81.29 ± 1.52	71.66 ± 1.75
	UNREAL	$\mathbf{72.82 \pm 3.55}$	$\mathbf{69.12 \pm 3.45}$	$\mathbf{57.66 \pm 1.96}$	$\mathbf{56.50 \pm 1.12}$	$\mathbf{78.73 \pm 0.88}$	$\mathbf{76.03 \pm 1.08}$	$\mathbf{84.30 \pm 0.30}$	$\mathbf{76.06 \pm 0.32}$
	Δ	**+3.38**	**+2.02**	**+5.06**	**+5.29**	**+5.00**	**+2.31**	**+0.60**	**+0.67**
GAT	Vanilla	51.58 ± 0.32	43.37 ± 0.21	37.91 ± 0.28	23.49 ± 0.21	62.07 ± 0.17	47.39 ± 0.20	72.66 ± 2.97	64.87 ± 3.46
	Re-Weight	58.28 ± 1.88	54.47 ± 2.35	38.13 ± 1.55	29.60 ± 3.02	67.41 ± 2.69	58.06 ± 5.07	77.10 ± 3.26	68.35 ± 2.71
	PC Softmax	63.74 ± 2.01	59.76 ± 2.19	45.07 ± 1.13	39.21 ± 2.29	69.68 ± 1.29	69.44 ± 1.29	79.72 ± 1.52	70.78 ± 1.45
	BalancedSoftmax	63.19 ± 1.35	61.03 ± 1.46	46.03 ± 2.11	43.38 ± 2.24	71.45 ± 1.23	69.10 ± 1.20	79.15 ± 2.08	69.68 ± 2.13
	Renode	60.04 ± 2.21	58.04 ± 2.66	42.40 ± 2.97	34.09 ± 0.04	68.54 ± 2.11	65.63 ± 3.15	75.34 ± 1.65	69.99 ± 1.60
	GraphENS	63.93 ± 2.70	61.77 ± 3.38	44.43 ± 1.90	39.26 ± 2.55	68.50 ± 1.81	64.14 ± 3.28	81.63 ± 2.08	71.20 ± 2.75
	BalancedSoftmax+TAM	64.96 ± 3.23	62.91 ± 3.96	52.75 ± 1.29	50.69 ± 1.83	73.38 ± 0.77	72.45 ± 0.88	80.86 ± 2.52	72.93 ± 2.95
	Renode+TAM	63.45 ± 1.41	61.51 ± 1.95	41.55 ± 1.39	36.13 ± 2.87	71.53 ± 2.35	68.11 ± 4.28	78.60 ± 1.90	70.35 ± 2.80
	GraphENS+TAM	62.52 ± 0.95	61.65 ± 1.19	45.79 ± 1.31	44.80 ± 1.14	69.09 ± 1.11	70.64 ± 1.10	83.33 ± 0.83	72.81 ± 1.22
	UNREAL	$\mathbf{75.42 \pm 0.91}$	$\mathbf{71.50 \pm 0.89}$	$\mathbf{60.35 \pm 1.87}$	$\mathbf{59.63 \pm 1.86}$	$\mathbf{77.88 \pm 1.31}$	$\mathbf{74.98 \pm 1.35}$	$\mathbf{85.33 \pm 0.19}$	$\mathbf{75.83 \pm 0.74}$
	Δ	**+10.46**	**+8.59**	**+7.60**	**+8.94**	**+4.50**	**+2.53**	**+2.00**	**+3.02**
SAGE	Vanilla	52.65 ± 0.24	43.79 ± 0.47	36.63 ± 0.09	24.12 ± 0.09	62.29 ± 0.25	47.02 ± 0.38	55.94 ± 2.37	47.21 ± 2.73
	Re-Weight	59.42 ± 2.88	55.26 ± 4.40	36.24 ± 1.30	27.07 ± 2.88	63.33 ± 0.75	55.11 ± 1.62	70.76 ± 3.35	62.09 ± 3.30
	PC Softmax	64.01 ± 1.15	60.74 ± 1.68	44.74 ± 1.41	37.61 ± 1.69	72.62 ± 1.42	70.95 ± 1.70	75.96 ± 2.44	69.12 ± 2.90
	BalancedSoftmax	63.43 ± 2.12	62.30 ± 2.27	49.33 ± 1.12	44.58 ± 1.64	70.68 ± 0.92	69.15 ± 0.84	74.66 ± 0.86	66.28 ± 1.92
	Renode	62.42 ± 0.90	60.08 ± 1.19	39.61 ± 2.66	30.13 ± 3.86	67.11 ± 1.12	61.09 ± 3.50	73.73 ± 2.26	64.47 ± 2.39
	GraphENS	63.09 ± 0.97	61.20 ± 1.74	42.03 ± 1.88	36.71 ± 2.99	69.71 ± 1.87	63.47 ± 3.87	81.33 ± 1.66	72.83 ± 1.76
	BalancedSoftmax+TAM	66.58 ± 1.53	64.56 ± 2.49	53.33 ± 1.06	50.15 ± 1.45	72.59 ± 2.06	72.22 ± 2.08	78.01 ± 1.06	71.02 ± 1.08
	Renode+TAM	62.06 ± 2.08	60.72 ± 3.32	42.08 ± 1.88	33.19 ± 3.45	69.95 ± 1.01	65.99 ± 2.28	74.81 ± 3.29	67.48 ± 3.32
	GraphENS+TAM	65.95 ± 2.25	63.88 ± 1.78	51.03 ± 1.51	50.49 ± 1.88	73.58 ± 2.01	72.44 ± 1.77	81.72 ± 1.08	72.31 ± 1.98
	UNREAL	$\mathbf{73.47 \pm 2.31}$	$\mathbf{68.30 \pm 2.11}$	$\mathbf{59.77 \pm 2.98}$	$\mathbf{58.92 \pm 3.07}$	$\mathbf{77.11 \pm 0.59}$	$\mathbf{74.03 \pm 0.81}$	$\mathbf{82.92 \pm 2.94}$	$\mathbf{73.11 \pm 2.57}$
	Δ	**+6.89**	**+3.74**	**+6.44**	**+8.43**	**+3.53**	**+1.59**	**+1.20**	**+0.28**

B.2. More Results When Unlabeled Data is Imbalanced

In Table 6, we found that existing over-sampling methods use too much memory due to synthetic node generation, and cannot handle Flickr on a 3090 GPU with 24GB memory. This include GraphENS (Park et al., 2021), GraphSMOTE (Zhao et al., 2021) and ReNode (Chen et al., 2021). More importantly, on Flickr, UNREAL consistently outperforms other approaches.

Table 6. Experimental results of our method UNREAL and other baselines on Flickr. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures.

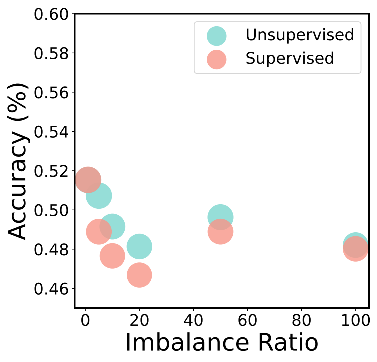
Dataset (Flickr)	GCN		GAT		SAGE	
Imbalance Ratio($\rho \approx \mathbf{10.80}$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
Vanilla	24.62 ± 0.07	24.53 ± 0.11	25.87 ±0.30	25.32 ± 0.44	25.29 ± 0.18	24.16 ± 0.27
Re-Weight	28.31 ± 1.64	24.06 ± 1.16	**30.66** ± **0.76**	27.12 ± 0.34	27.39 ± 1.84	22.62 ± 1.04
PC Softmax	<u>29.21 ± 2.16</u>	<u>25.81 ± 1.75</u>	30.20 ± 0.46	<u>27.24 ± 0.37</u>	25.40 ± 2.49	21.08 ± 1.73
BalancedSoftmax	27.61 ± 0.61	23.70 ± 0.77	26.01 ± 2.81	23.50 ± 3.07	28.24 ± 2.10	24.98 ± 1.59
GraphSMOTE	OOM	OOM	OOM	OOM	OOM	OOM
Renode	OOM	OOM	OOM	OOM	OOM	OOM
GraphENS	OOM	OOM	OOM	OOM	OOM	OOM
BalancedSoftmax+TAM	27.06 ± 1.03	23.97 ± 0.60	28.24 ± 0.99	25.52 ± 0.89	<u>29.79 ± 0.37</u>	<u>27.56 ± 0.25</u>
Renode+TAM	OOM	OOM	OOM	OOM	OOM	OOM
GraphENS+TAM	OOM	OOM	OOM	OOM	OOM	OOM
UNREAL	**30.76** ± **0.27**	**30.60** ± **0.29**	<u>29.45 ± 0.72</u>	**28.21** ± **0.76**	**50.68** ± **0.63**	**51.01** ± **1.34**
Δ	**+1.55**	**+4.79**	**-1.21**	**+0.97**	**+20.89**	**+23.45**

C. More Analysis

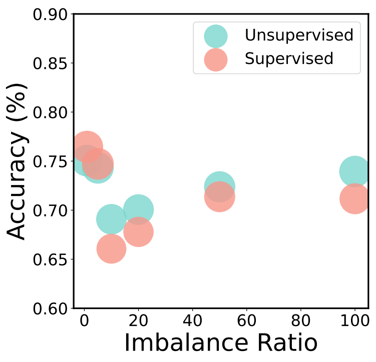
C.1. More Experiments of the Motivating Example in Section 4.1.

We conduct richer experiments on Cora and Amazon-Computers based on three different GNN architectures to verify the motivating example in Section 4.1. We hypothesize that even if the GNN encoder is trained on skewed data, the embeddings it learns are of high quality.

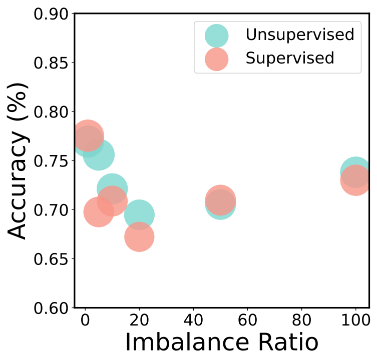
Details of Experimental Setup. As explained in Section 4.1, we can obtain two pseudo-labels for all unlabeled nodes, one from unsupervised algorithms and the other from supervised classifiers. Experiments on more datasets are conducted to compare the accuracy of the two pseudo-labels for all unlabeled nodes. We chose the two benchmark datasets, Cora and Amazon-Computers, to build scenarios with varying degrees of imbalance ($\rho = 1, 5, 10, 20, 50, 100$). To be more specific, half of the classes are designated as minority classes and randomly selected labeled nodes are converted into unlabeled nodes until the training set's imbalance ratio reaches rho. The GNN architecture is fixed as the 2-layer GNN (i.e. GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), GraphSAGE (Hamilton et al., 2017)) having 128 hidden dimensions and train models for 2000 epochs. We set the K-Means algorithm's cluster size k' to 200. Each experiment is repeated five times, and the average experiment results under different imbalance ratios are shown in Figure 7.



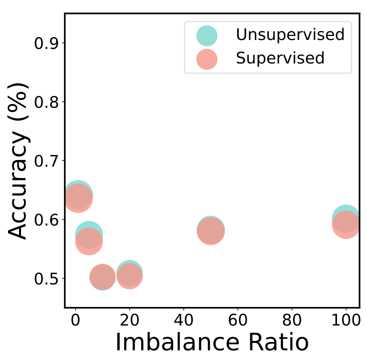
(a) Cora-SAGE



(b) Computers-GCN



(c) Computers-GAT



(d) Computers-SAGE

Figure 7. The partial experimental results on Cora and Amazon-Computers under different imbalance scenarios ($\rho = 1, 5, 10, 20, 50, 100$). We compare the accuracy of the two pseudo-labels (predictions) from unsupervised algorithms and supervised classifiers respectively for all unlabeled nodes.

Analysis. As illustrated by Figure 7, Figure 3(a) and Figure 3(b), the predictions given by unsupervised algorithms still have a high accuracy rate even in imbalanced scenarios. The final results reveal several intriguing aspects: (1) In imbalanced scenarios, the performance of both supervised and unsupervised algorithms degrades, especially in extreme cases ($\rho =$

50,100). (2) The predictions given by the embedding space are superior to the biased classifier, which meaningfully indicates that the classifier is the more under-performed component in the model when trained on an imbalanced training set. (3) A large number of experimental results demonstrate that the predictions from unsupervised algorithms and classifiers are of reference significance, implying that relying on a single component will not result in a good performance. Motivating by this, we propose DPAM (Section 4.1) and Node-Reordering (Section 4.2), the two fundamental components of our algorithm.

C.2. Additional Analysis for DPAM (Section 4.1)

DPAM uses an unsupervised algorithm to obtain pseudo-labels for each unlabeled node in the embedding space, and only unlabeled nodes with aligned pseudo-labels and classifier predictions are put into the candidate pool, effectively circumventing the classifier's bias problem, such as selecting low-quality nodes into the training set based on the skewed confidence rankings. We conduct the novel experiments listed below to see through its essence.

Details of Experimental Setup. We use DPAM to filter the unlabeled nodes of the whole graph, and test the accuracy of pseudo-labels (prediction of the classifier) of the aligned node set $\mathcal{U}_{in}$ and the discarded node set $\mathcal{U}_{out}$ respectively. DPAM based on different GNN structures are trained on two node classification benchmark datasets, Cora, and Amazon-Computers. We process the two datasets with a traditional imbalanced distribution following Zhao et al. (2021); Park et al. (2021); Song et al. (2022). The imbalance ratio ρ between the numbers of the most frequent class and the least frequent class is set as 1, 5, 10, 20, 50, and 100. We fix architecture as the 2-layer GNN (i.e. GCN(Kipf & Welling, 2016), GAT(Veličković et al., 2017), GraphSAGE(Hamilton et al., 2017)) having 128 hidden dimensions and train models for 2000 epochs. We select the model by the validation accuracy. We observe the accuracy of pseudo labels for unlabeled nodes which are filtered out and absorbed into by DPAM respectively. We repeat each experiment five times and present the average experiment results in Table 7 and Table 8.

Analysis. DPAM divides the unlabeled nodes of the whole graph into two parts, $\mathcal{U}_{in}, \mathcal{U}_{out}$. We verify the effect of DPAM by testing the accuracy of pseudo-labels for these two parts of nodes. We can observe that the accuracy of pseudo-labels for $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ differ greatly in different imbalanced scenarios. Usually the pseudo-label accuracy of $\mathcal{U}_{in}$ is high and the pseudo-label accuracy of $\mathcal{U}_{out}$ is lower, which means the effectiveness of DPAM. We can also observe that as ρ increases, the accuracy of both decreases, which also reflects the model bias caused by the imbalanced label distribution.

Table 7. Experimental results of DPAM effectiveness on **Cora** with $\rho = 1, 5, 10, 20, 50, 100$. We observe the accuracy (%) of the pseudo-label (prediction of the classifier) of the aligned node set $\mathcal{U}_{in}$ and the discarded node set $\mathcal{U}_{out}$ respectively. We report averaged results with the standard errors over 5 repetitions on three representative GNN architectures. **All**, **Labeled**, **Unlabeled** represent the size of whole nodes, labeled nodes, and unlabeled nodes on the graph. **Align**, **Out**, **Align-True**, **Out-Ture** represent the size of $\mathcal{U}_{in}, \mathcal{U}_{out}$, nodes with accurate pseudo-labels of $\mathcal{U}_{in}, \mathcal{U}_{out}$ respectively.

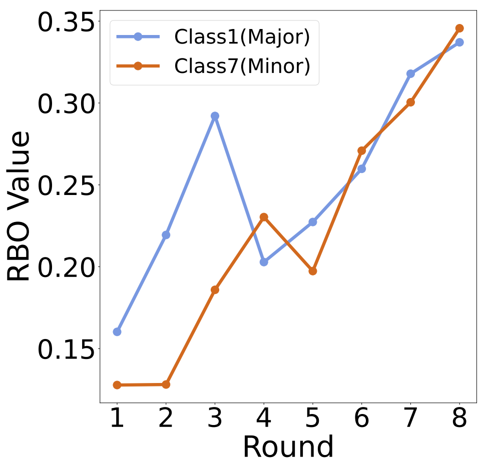
	Dataset	All	Labled	Unlabled	Align	Align-True	Accuracy(%)	Out	Out-True	Accuracy(%)
GCN	$\rho = 1$	2708	140	2568	2072.00 ± 10.29	1391.00 ± 22.56	$\mathbf{67.11 \pm 1.17}$	496.00 ± 10.29	233.80 ± 16.66	$\mathbf{47.17 \pm 3.74}$
	$\rho = 5$	2708	92	2616	2122.80 ± 18.93	1392.00 ± 34.21	$\mathbf{65.58 \pm 1.57}$	493.20 ± 18.73	186.80 ± 13.08	$\mathbf{37.86 \pm 1.75}$
	$\rho = 10$	2708	86	2622	2134.60 ± 23.42	1326.40 ± 24.23	$\mathbf{62.14 \pm 1.67}$	487.40 ± 23.43	181.60 ± 18.24	$\mathbf{37.32 \pm 3.13}$
	$\rho = 20$	2708	83	2625	2149.60 ± 17.67	1310.20 ± 86.72	$\mathbf{60.97 \pm 3.50}$	475.40 ± 17.67	169.80 ± 21.47	$\mathbf{35.64 \pm 3.44}$
	$\rho = 50$	2708	203	2505	1860.80 ± 31.15	1059.40 ± 58.77	$\mathbf{56.90 \pm 2.62}$	644.20 ± 31.14	225.80 ± 10.70	$\mathbf{35.05 \pm 3.79}$
	$\rho = 100$	2708	403	2305	1820.40 ± 12.42	1001.60 ± 21.60	$\mathbf{55.02 \pm 3.99}$	484.60 ± 23.99	151.40 ± 20.74	$\mathbf{31.78 \pm 2.37}$
GAT	$\rho = 1$	2708	140	2568	2072.00 ± 37.18	1412.40 ± 37.31	$\mathbf{68.16 \pm 1.41}$	496.00 ± 20.89	239.40 ± 11.37	$\mathbf{48.29 \pm 2.15}$
	$\rho = 5$	2708	92	2616	2141.40 ± 26.36	1433.00 ± 59.82	$\mathbf{66.90 \pm 2.09}$	474.60 ± 26.36	195.20 ± 24.68	$\mathbf{41.02 \pm 3.27}$
	$\rho = 10$	2708	86	2622	2132.60 ± 29.94	1377.40 ± 49.61	$\mathbf{64.58 \pm 1.60}$	489.40 ± 29.95	185.80 ± 12.28	$\mathbf{37.97 \pm 1.13}$
	$\rho = 20$	2708	83	2625	2150.60 ± 37.35	1344.60 ± 54.17	$\mathbf{62.16 \pm 1.64}$	462.40 ± 33.28	178.00 ± 5.05	$\mathbf{38.60 \pm 2.12}$
	$\rho = 50$	2708	140	2568	1892.40 ± 37.18	1080.80 ± 31.86	$\mathbf{57.52 \pm 1.52}$	612.60 ± 37.17	271.20 ± 6.30	$\mathbf{44.35 \pm 1.86}$
	$\rho = 100$	2708	403	2305	1934.60 ± 19.65	1038.20 ± 21.08	$\mathbf{53.66 \pm 0.83}$	370.40 ± 37.17	147.53 ± 3.20	$\mathbf{39.83 \pm 1.36}$
SAGE	$\rho = 1$	2708	140	2568	1944.00 ± 25.77	973.40 ± 32.26	$\mathbf{51.27 \pm 3.36}$	624.00 ± 25.77	237.00 ± 13.28	$\mathbf{36.11 \pm 4.07}$
	$\rho = 5$	2708	92	2616	2004.40 ± 35.50	1038.20 ± 22.53	$\mathbf{51.80 \pm 3.73}$	611.60 ± 35.50	203.80 ± 7.15	$\mathbf{33.40 \pm 1.85}$
	$\rho = 10$	2708	86	2622	2041.60 ± 32.48	1039.00 ± 41.32	$\mathbf{50.89 \pm 1.88}$	580.40 ± 32.48	189.20 ± 2.35	$\mathbf{32.56 \pm 4.25}$
	$\rho = 20$	2708	83	2625	2040.20 ± 30.94	1002.20 ± 66.97	$\mathbf{48.95 \pm 2.66}$	578.80 ± 30.95	186.60 ± 18.00	$\mathbf{32.18 \pm 1.57}$
	$\rho = 50$	2708	203	2505	1789.40 ± 30.56	870.20 ± 24.33	$\mathbf{48.63 \pm 1.03}$	715.60 ± 30.56	242.40 ± 16.77	$\mathbf{33.87 \pm 1.18}$
	$\rho = 100$	2708	403	2305	1859.00 ± 192.42	914.41 ± 23.65	$\mathbf{49.26 \pm 2.59}$	446.00 ± 21.24	138.87 ± 6.32	$\mathbf{31.15 \pm 2.43}$

Table 8. Experimental results of DPAM effectiveness on **Amazon-Computers** with $\rho = 1, 5, 10, 20, 50, 100$.

	Dataset	All	Labled	Unlabled	Align	Align-True	Accuracy(%)	Out	Out-True	Accuracy(%)
GCN	$\rho = 1$	13752	200	13552	11977.60 ± 108.09	9603.80 ± 93.34	**80.08 ± 3.07**	1554.40 ± 08.23	676.60 ± 141.11	**43.58 ± 2.83**
	$\rho = 5$	13752	120	13632	11593.60 ± 73.16	9172.80 ± 87.32	**79.06 ± 1.17**	2308.40 ± 173.54	544.40 ± 66.26	**30.74 ± 9.09**
	$\rho = 10$	13752	110	13642	11822.40 ± 13.43	8786.60 ± 55.48	**74.24 ± 0.83**	1807.60 ± 109.34	495.00 ± 100.37	**27.24 ± 4.30**
	$\rho = 20$	13752	105	13647	11866.60 ± 17.34	8698.20 ± 188.13	**73.40 ± 1.39**	1780.40 ± 67.36	521.00 ± 60.76	**29.20 ± 2.41**
	$\rho = 50$	13752	255	13497	11843.20 ± 168.20	8994.40 ± 175.24	**75.94 ± 0.75**	1653.80 ± 138.11	474.20 ± 50.72	**28.68 ± 2.16**
	$\rho = 100$	13752	505	13247	9159.00 ± 192.42	7352.90 ± 61.23	**81.41 ± 4.59**	4088.00 ± 93.99	1129.60 ± 75.74	**28.67 ± 4.77**
GAT	$\rho = 1$	13752	200	13552	12008.00 ± 101.93	9984.20 ± 308.03	**83.44 ± 4.13**	1544.80 ± 101.94	580.40 ± 190.49	**43.33 ± 1.32**
	$\rho = 5$	13752	120	13632	11570.80 ± 136.11	8715.00 ± 86.33	**75.33 ± 0.54**	2061.20 ± 136.13	477.00 ± 97.07	**25.39 ± 1.33**
	$\rho = 10$	13752	110	13642	8947.60 ± 13.40	6680.40 ± 177.54	**75.85 ± 6.07**	4694.40 ± 134.74	591.80 ± 13.74	**15.94 ± 2.97**
	$\rho = 20$	13752	105	13647	10245.80 ± 68.00	7300.80 ± 64.89	**71.42 ± 1.80**	3401.20 ± 69.76	370.60 ± 43.87	**18.52 ± 0.09**
	$\rho = 50$	13752	255	13497	10133.60 ± 31.56	7772.00 ± 155.87	**77.17 ± 2.85**	3363.40 ± 10.42	457.20 ± 108.19	**19.28 ± 1.43**
	$\rho = 100$	13752	505	13247	11377.00 ± 63.32	9122.20 ± 96.70	**80.46 ± 1.01**	1910.00 ± 63.32	458.20 ± 41.04	**24.78 ± 2.04**
SAGE	$\rho = 1$	13752	200	13552	10815.20 ± 86.50	7131.40 ± 72.83	**65.94 ± 0.28**	2736.80 ± 86.50	965.40 ± 56.42	**35.26 ± 1.31**
	$\rho = 5$	13752	120	13632	10627.80 ± 78.33	6728.00 ± 53.24	**63.25 ± 0.36**	3004.20 ± 78.03	978.20 ± 59.93	**32.55 ± 1.49**
	$\rho = 10$	13752	110	13642	10475.00 ± 118.41	6015.00 ± 41.14	**57.43 ± 4.01**	3167.00 ± 18.41	1064.40 ± 52.71	**33.59 ± 6.23**
	$\rho = 20$	13752	105	13647	10653.20 ± 87.35	5998.40 ± 69.35	**56.30 ± 4.01**	2993.80 ± 87.35	886.20 ± 73.25	**29.57 ± 1.77**
	$\rho = 50$	13752	255	13497	11044.80 ± 129.14	6760.80 ± 50.26	**61.22 ± 3.42**	2442.20 ± 28.48	879.00 ± 91.45	**35.71 ± 1.78**
	$\rho = 100$	13752	505	13247	9175.20 ± 32.53	6475.60 ± 80.88	**72.07 ± 1.96**	4071.80 ± 32.63	1218.60 ± 14.70	**34.43 ± 1.08**

C.3. More Results and Analysis about Fluctuation of RBO Values (Section 4.2)

In Section 4.2, we argue that as the iteration progresses, the confidence given by the classifier becomes increasingly valuable (the training set is gradually balanced), whereas the geometric rankings are calculated in the embedding space and are unaffected by the classifier. As a result, it is trustworthy that as the credibility of confidence grows in the iterative process, the similarities between the Confidence Rankings and the Geometric Rankings will gradually increase. Notably, the unsupervised algorithm performs worse overall than supervised methods, especially for a balanced training set. As a result, by combining the features of the two rankings, we can greatly improve the performance of our algorithm. Experiments are used to put the aforementioned hypothesis to the test.



(a) Cora-SAGE

Figure 8. Fluctuation of RBO values (ρ = 10) of two rankings as iterations progress.

Details of Experimental Setup. We conduct more experiments on Cora (ρ = 10) to observe the similarities between the Geometric Rankings and Confidence Rankings. The architecture is fixed as the 2-layer GNN (i.e. GCN(Kipf & Welling, 2016), GAT(Veličković et al., 2017), GraphSAGE(Hamilton et al., 2017)) having 128 hidden dimensions and train models for 2000 epochs. The UNREAL model's hyperparameter settings can be found in Appendix D.4. We choose a majority and a minority class at random to compare the similarities of their respective two rankings (our setting is the first class and the last class of Cora), and we limit the number of iterations to eight. Each experiment is repeated five times, and the average experiment results are reported in Figure 3(c), Figure 3(d), and Figure 8.

Analysis. As illustrated by Figure 3(c), Figure 3(d) and Figure 8, it can be observed that the similarities between the Confidence Rankings and the Geometric Rankings will gradually increase as the iteration progresses in the early stages. As a result, our hypothesis is confirmed. It is worth noting that the similarity of the two rankings of the minority class is higher than the majority class when the training set is gradually balanced, which also reflects the compensation benefit of UNREAL for the minority class.

C.4. Additional Analysis for Node-Reordering (Section 4.2)

In this section, we analyze why Node-Reordering works. With DPAM, we filter out a large part of untrustworthy nodes and get a pool of candidate nodes. We try to carefully hunt for a part of high-quality nodes in the pool to add to the training set, which involves a priority issue. As we mentioned before, we have already verified in Section 3.2 that the prediction and confidence given by the classifier are biased, resulting in low accuracy of the pseudo-labels for nodes selected by ST in highly imbalanced scenarios. We can get the geometric ranking according to the distance between the unlabeled nodes and the class centers in the embedding space. Considering the influence of classifier bias on confidence ranking, we believe that geometric ranking is more credible in the early rounds. At the same time, we take into account the suboptimal nature of the unsupervised algorithm. We believe that with the rounds of UNREAL increases, the label distribution of the training set is gradually balanced, and the confidence given by the classifier is more reliable. Node-reordering considers both geometric ranking and confidence ranking, specifically, obtaining the similarity between them to get a weight to reorder the priority of the nodes. To quantify the performance of Node-Reordering, we conduct the novel experiments below.

Details of Experimental Setup. We conduct experiments to test the accuracy of pseudo labels for unlabeled nodes on class-imbalanced graphs. All model combinations based on different GNN structures are trained on two node classification benchmark datasets, Cora, and Amaon-Computers. We process the two datasets with a traditional imbalanced distribution following Zhao et al. (2021); Park et al. (2021); Song et al. (2022). The imbalance ratio ρ between the numbers of the most frequent class and the least frequent class is set as 1, 5, 10, 20, 50, and 100. We fix architecture as the 2-layer GNN (i.e. GCN(Kipf & Welling, 2016), GAT(Veličković et al., 2017), GraphSAGE(Hamilton et al., 2017)) having 128 hidden dimensions and train models for 2000 epochs. We select the model by the validation accuracy. We observe the accuracy of pseudo labels for unlabeled nodes which are newly added to the minority class of the training set. We repeat each experiment five times and present the average experiment results in Table 9 and Table 10.

Analysis. As shown in Table 9 and Table 10, we verify the effectiveness of each component of UNREAL by testing the accuracy of the nodes' pseudo-labels selected by different model combinations, DPAM+Confidence ranking(with or without DGIN), DPAM+Geometric ranking(with or without DGIN), DPAM+Node-Reordering(with or without DGIN). It can be observed that in different imbalanced scenarios, each component of UNREAL (Node-reordering & DGIN) plays an important role, and the performance outperforms the other model combinations significantly.

C.5. Additional Analysis for GI and DGIN (Section 4.3)

In this section, We elaborate on the issue, Geometric Imbalance and verify the effectiveness of DGIN.

Geometric Imbalance. As the prior work Song et al. (2022); Chen et al. (2021) argued, too many labeled nodes remaining near the boundary between two classes will have a significant impact on GNN classification performance. In our work, we try to select some unlabeled nodes to join the training set, which means that nodes near the center of the classes (far from any class boundaries) should be prioritized. Node-Reordering has made some efforts in this part (Section 4.2). Considering that UNREAL iteratively selects nodes to join the training set (Section 4.4), some suboptimal nodes (not so close to class centers) are gradually considered in later stages, which is referred to as Geometric Imbalance (GI). We generalize the novel Geometric Imbalance to have the following properties: (1) Only unlabeled nodes suffer from GI, and this is the first work to investigate the local topology of unlabeled nodes in the imbalanced node classification problem. (2) Geometrically imbalanced nodes are on class boundaries and are frequently indistinguishable. (3) For unlabeled nodes that encounter GI, DPAM often encounters the conflict of pseudo-labeling. This is because these nodes in the embedding space are roughly the same distance apart from the two class centers.

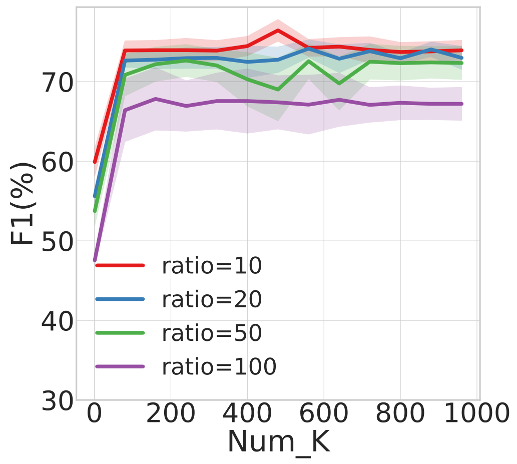
Table 9. Analyzed experimental results of Node-Reordering and DGIN on **Cora** with $\rho = 1, 5, 10, 20, 50, 100$. We select 100 unlabeled nodes newly added to the minority class of training set through different method combinations, and evaluate the validity of Node-Reordering & DGIN by testing the accuracy (%) with the standard errors of the pseudo labels for these nodes. We report averaged results over 5 repetitions on three representative GNN architectures.

	Dataset	Cora					
	Imbalance Ratio (ρ)	$\rho = 1$	$\rho = 5$	$\rho = 10$	$\rho = 20$	$\rho = 50$	$\rho = 100$
GCN	DPAM+Confidence Ranking	61.40 ± 2.73	62.40 ± 2.59	60.20 ± 1.02	58.40 ± 1.05	57.60 ± 1.86	58.40 ± 2.15
GCN	DPAM+Geometric Ranking	64.00 ± 3.67	61.20 ± 2.89	61.20 ± 2.54	63.60 ± 1.31	55.60 ± 2.31	47.80 ± 2.87
GCN	DPAM+Node-Reordering	89.65 ± 3.23	86.98 ± 0.21	88.32 ± 0.83	85.32 ± 2.98	90.87 ± 2.31	71.60 ± 2.91
GCN	DPAM+Confidence Ranking+DGIN	71.00 ± 5.47	75.40 ± 2.15	68.20 ± 1.25	69.40 ± 1.28	67.80 ± 2.75	66.60 ± 0.16
GCN	DPAM+Geometric Ranking+DGIN	69.60 ± 3.78	73.80 ± 0.45	64.80 ± 1.26	64.20 ± 1.91	57.00 ± 1.57	69.00 ± 1.71
GCN	**DPAM+Node-Reordering+DGIN(UNREAL)**	**92.80 ± 1.30**	**96.40 ± 4.27**	**92.20 ± 0.85**	**89.40 ± 1.37**	**93.00 ± 0.82**	**77.80 ± 2.50**
GAT	DPAM+Confidence Ranking	61.60 ± 4.26	64.00 ± 2.07	62.60 ± 3.47	57.80 ± 1.65	58.20 ± 1.07	60.60 ± 0.79
GAT	DPAM+Geometric Ranking	64.00 ± 2.78	67.80 ± 3.76	65.00 ± 4.30	52.00 ± 1.02	65.20 ± 2.58	40.80 ± 2.63
GAT	DPAM+Node-Reordering	91.79 ± 0.23	90.45 ± 5.78	84.32 ± 3.45	88.34 ± 0.23	90.32 ± 0.43	75.34 ± 1.54
GAT	DPAM+Confidence Ranking+DGIN	69.80 ± 2.77	72.80 ± 3.94	72.40 ± 1.13	67.60 ± 1.59	71.60 ± 9.12	64.00 ± 1.74
GAT	DPAM+Geometric Ranking+DGIN	73.60 ± 4.82	74.00 ± 5.47	68.40 ± 1.62	57.20 ± 2.17	68.00 ± 1.17	62.00 ± 1.53
GAT	**DPAM+Node-Reordering+DGIN(UNREAL)**	**93.80 ± 1.92**	**91.20 ± 4.60**	**90.40 ± 1.69**	**90.00 ± 9.92**	**94.60 ± 4.92**	**78.20 ± 2.47**
SAGE	DPAM+Confidence Ranking	54.80 ± 4.96	53.00 ± 2.46	51.80 ± 1.97	43.60 ± 2.57	46.20 ± 0.53	41.60 ± 1.14
SAGE	DPAM+Geometric Ranking	53.60 ± 2.78	45.40 ± 1.75	40.60 ± 0.26	52.60 ± 2.47	47.40 ± 4.27	44.80 ± 2.84
SAGE	DPAM+Node-Reordering	90.69 ± 0.21	86.90 ± 0.56	86.45 ± 3.21	88.34 ± 2.43	75.34 ± 4.20	76.43 ± 1.43
SAGE	DPAM+Confidence Ranking+DGIN	66.20 ± 5.78	59.00 ± 3.04	63.80 ± 1.52	54.60 ± 1.64	60.60 ± 1.37	57.40 ± 2.26
SAGE	DPAM+Geometric Ranking+DGIN	61.60 ± 3.71	61.80 ± 5.21	54.00 ± 7.31	53.60 ± 1.38	63.00 ± 1.23	45.20 ± 1.96
SAGE	**DPAM+Node-Reordering+DGIN(UNREAL)**	**97.80 ± 1.78**	**92.20 ± 1.32**	**90.80 ± 1.82**	**89.20 ± 1.39**	**94.20 ± 8.04**	**85.40 ± 1.02**

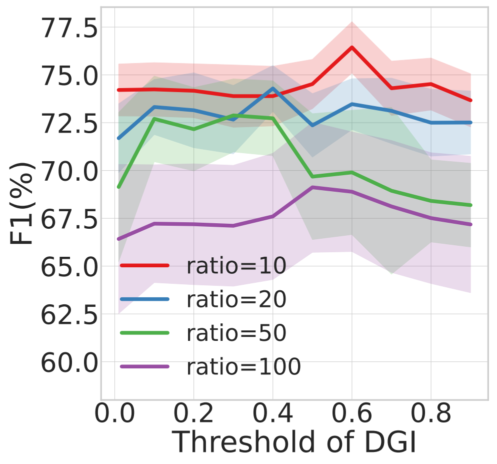
Table 10. Analyzed experimental results of Node-Reordering and DGIN on **Amazon-Computers** with $\rho = 1, 5, 10, 20, 50, 100$. We select 100 unlabeled nodes newly added to the minority class of training set through different method combinations, and evaluate the validity of Node-Reordering & DGIN by testing the accuracy (%) with the standard errors of the pseudo labels for these nodes. We report averaged results over 5 repetitions on three representative GNN architectures.

	Dataset	Amazon-Computers					
	Imbalance Ratio (ρ)	$\rho = 1$	$\rho = 5$	$\rho = 10$	$\rho = 20$	$\rho = 50$	$\rho = 100$
GCN	DPAM+Confidence Ranking	75.40 ± 2.50	70.20 ± 3.03	74.88 ± 3.11	68.20 ± 4.20	63.60 ± 2.30	61.40 ± 1.51
GCN	DPAM+Geometric Ranking	76.00 ± 1.41	74.80 ± 4.71	76.80 ± 2.28	65.80 ± 3.27	64.80 ± 3.70	65.60 ± 3.98
GCN	DPAM+Node-Reordering	82.80 ± 2.38	79.60 ± 3.64	78.20 ± 0.26	74.00 ± 3.28	65.20 ± 1.87	66.00 ± 2.82
GCN	DPAM+Confidence Ranking+DGIN	76.40 ± 2.07	67.20 ± 4.32	75.80 ± 2.38	66.20 ± 3.70	62.80 ± 0.12	59.20 ± 1.30
GCN	DPAM+Geometric Ranking+DGIN	78.20 ± 0.83	80.00 ± 1.22	76.40 ± 1.67	66.00 ± 2.44	64.20 ± 3.83	66.20 ± 2.38
GCN	**DPAM+Node-Reordering+DGIN(UNREAL)**	**84.40 ± 3.60**	**82.20 ± 2.16**	**80.40 ± 3.46**	**80.60 ± 1.51**	**69.60 ± 3.04**	**66.40 ± 3.20**
GAT	DPAM+Confidence Ranking	84.60 ± 2.40	79.20 ± 1.78	73.00 ± 2.12	74.80 ± 2.16	65.00 ± 1.73	68.60 ± 1.40
GAT	DPAM+Geometric Ranking	86.00 ± 3.80	79.80 ± 2.94	74.80 ± 3.42	75.00 ± 2.91	70.80 ± 2.16	69.40 ± 1.10
GAT	DPAM+Node-Reordering	87.40 ± 2.30	80.60 ± 3.04	80.40 ± 2.19	79.00 ± 3.67	75.00 ± 1.22	73.40 ± 2.52
GAT	DPAM+Confidence Ranking+DGIN	84.20 ± 1.64	79.40 ± 2.07	76.40 ± 6.50	76.00 ± 2.34	66.00 ± 0.12	72.00 ± 1.84
GAT	DPAM+Geometric Ranking+DGIN	83.80 ± 1.09	80.20 ± 1.09	76.20 ± 2.28	77.80 ± 2.58	71.60 ± 0.89	69.00 ± 1.16
GAT	**DPAM+Node-Reordering+DGIN(UNREAL)**	**89.00 ± 2.54**	**86.60 ± 2.50**	**85.60 ± 4.44**	**83.40 ± 3.31**	**78.00 ± 3.39**	**79.80 ± 3.03**
SAGE	DPAM+Confidence Ranking	85.20 ± 3.38	80.20 ± 6.26	84.8 ± 0.83	77.60 ± 0.89	61.00 ± 0.70	65.40 ± 2.65
SAGE	DPAM+Geometric Ranking	86.00 ± 0.70	81.20 ± 2.16	83.40 ± 1.14	78.00 ± 1.22	61.40 ± 0.54	65.00 ± 1.72
SAGE	DPAM+Node-Reordering	86.00 ± 1.58	83.20 ± 3.27	84.60 ± 0.54	79.20 ± 1.92	61.80 ± 0.44	67.80 ± 1.03
SAGE	DPAM+Confidence Ranking+DGIN	86.40 ± 2.07	81.60 ± 3.20	83.40 ± 1.14	**79.20 ± 0.44**	61.20 ± 0.44	70.40 ± 3.59
SAGE	DPAM+Geometric Ranking+DGIN	87.00 ± 2.12	80.80 ± 2.48	84.20 ± 1.30	78.20 ± 1.48	61.20 ± 0.47	68.20 ± 1.72
SAGE	**DPAM+Node-Reordering+DGIN(UNREAL)**	**88.20 ± 2.16**	**87.60 ± 1.14**	**85.40 ± 4.72**	78.00 ± 1.55	**66.20 ± 2.86**	**72.20 ± 0.83**

Motivating Example For GI. Here, we conduct a novel experiment on Cora (ρ = 10) to verify the necessity of the GI issue. We choose two adjacent classes (our setting is the first and last classes of Cora) at random in the embedding space and examine only the nodes with pseudo-labels that belong to these two classes (these nodes were previously filtered by DPAM), and we divide the category boundaries of the two classes into five regions, S1, S2, S3, S4, S5. We independently validate the pseudo-label accuracy of unlabeled nodes in each region. The architecture is fixed as the 2-layer GCN(Kipf & Welling, 2016) having 128 hidden dimensions and train models for 2000 epochs. Each experiment is repeated five times, and the average experiment results are reported in Figure 10(b). The diagram of the motivating example is Figure 10(a).



(a) Sensitivity performance of k' of K-Means

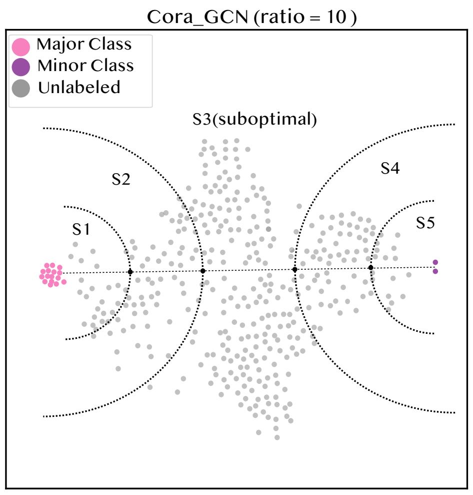


(b) Sensitivity performance of the threshold γ of DGI

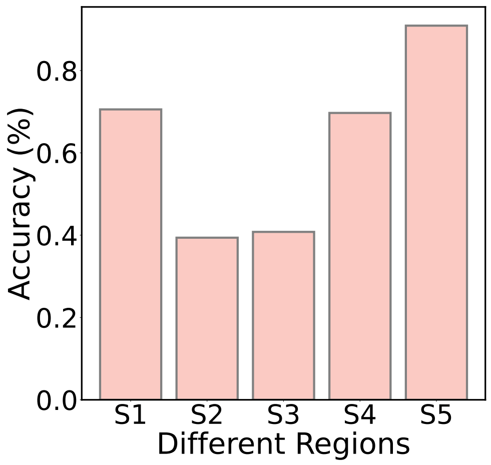
Figure 9. Sensitivity analysis on Cora based on GCN. The two images show the performance change as clusters' size k' of K-Means and the threshold γ of DGI increases respectively.

Table 11. Ablation analysis on different components

Modules	Confidence ranking	Geometric ranking	Node-reordering	DGI	F1
Cora+GCN ($\rho = 10$)	✔	✕	✕	✕	73.93 ± 0.95
	✔	✕	✕	✔	72.74 ± 0.63
	✕	✔	✕	✕	75.85 ± 0.82
	✕	✔	✕	✔	75.34 ± 0.63
	✕	✕	✔	✕	75.00 ± 0.97
	✕	✕	✔	✔	**76.44 ± 1.06**
CiteSeer+SAGE ($\rho = 20$)	✔	✕	✕	✕	46.09 ± 4.08
	✔	✕	✕	✔	47.76 ± 1.06
	✕	✔	✕	✕	50.32 ± 3.75
	✕	✔	✕	✔	53.32 ± 3.75
	✕	✕	✔	✕	**58.71± 3.21**
	✕	✕	✔	✔	57.51 ± 4.92
PubMed+GAT ($\rho = 50$)	✔	✕	✕	✕	76.34 ± 0.39
	✔	✕	✕	✔	75.42 ± 0.39
	✕	✔	✕	✕	77.32 ± 0.21
	✕	✔	✕	✔	76.89 ± 1.43
	✕	✕	✔	✕	76.12 ± 2.63
	✕	✕	✔	✔	**77.38 ± 0.39**
Computers+GAT ($\rho = 100$)	✔	✕	✕	✕	70.86 ± 1.73
	✔	✕	✕	✔	68.86 ± 1.42
	✕	✔	✕	✕	72.32 ± 2.43
	✕	✔	✕	✔	73.65 ± 0.67
	✕	✕	✔	✕	74.03 ± 2.53
	✕	✕	✔	✔	**75.83 ± 0.74**



(a) The diagram of the Motivating Example



(b) Accuracy of pseudo-labels for unlabeled nodes per region.

Figure 10. (a) We visualize the motivating example's intent. The setting is to divide the distance between the class centers into five equal parts and then evenly divide the class boundaries. (b) Accuracy of pseudo-labels for unlabeled nodes per region..

Analysis. In Figure 10(b), the result reveals several intriguing aspects: (1) Unlabeled nodes near the classification boundary are vulnerable to GI issues, and the specific performance is that the correctness of pseudo-labels is extremely low. (2) Geometric Ranking has a high reference significance for selecting high-quality nodes. The closer the unlabeled node is to the center of the class, the higher its pseudo-label accuracy. (3) Including these geometrically imbalanced nodes in the training set will introduce a lot of noise into the model training, so a simple and effective method to deal with this issue is desperately needed. Based on this, we propose DGIN. Looking back at the rich analysis experiments, Table 9 and Table 10 in Appendix C.4, it can be observed that DGIN plays an important role in various imbalanced scenarios, and its performance significantly outperforms the other model combinations, which validates DGIN's effectiveness in dealing with GI issues and contributes to the model's overall performance.

C.6. Hyperparameter Sensitivity Analysis of UNREAL

We investigate the sensitivity of performance to clusters' size k' of the K-Means algorithm and the threshold γ of DGIN in Figure 9. We observe the performance gradually stabilizes when k' has extremely high values, on the other hand, when k' is extremely low values, the performance of UNREAL drops largely. We believe that when k' is too small, the pseudo-labels given by unsupervised algorithms will have more errors. Also, we observe the performance gradually stabilizes when γ has extremely low values. We believe this is because the DGIN screening is too strict, which will lead to the loss of some high-quality nodes. On the other hand, extremely large γ will introduce much noise into the training set.

C.7. Ablation Analysis

In this section, we conduct ablation studies to analyze the benefit of each component in our method. From the results in Section 3.2, the necessity of unsupervised learning in the embedding space has been verified. Thus, in this section, DPAM is applied in all comparing methods. Here, we test the performance of three different ranking methods, namely confidence ranking, geometric ranking, and Node-reordering (which combines the former two rankings with information retrieval techniques). Moreover, we test the effect of DGIN, which aims to eliminate geometrically imbalanced nodes. As shown in Table 11, each component of our method can bring performance improvements. In particular, in three out of four settings in the table, Node-Reordering+DGIN achieves the best F1 scores. In all cases, geometric ranking outperforms confidence ranking, proving our hypothesis that prediction confidence scores may contain bias and be less reliable.

D. Details of the experimental setup

Here, we introduce the method of imbalanced datasets construction, evaluation protocol, and the details of our algorithm and baseline methods.

D.1. Imbalanced datasets construction

Table 12. Summary of the datasets used in our experiments.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Amazon-Computers	13,752	491,722	767	10
Flickr	89,250	899,756	500	7

The detailed descriptions of the datasets are shown in Table 12. For each citation dataset, for $\rho = 10, 20$, we follow the "public" split, and randomly convert minority class nodes to unlabeled nodes until the dataset reaches an imbalanced ratio ρ. For $\rho = 50, 100$, since there are not enough nodes per class in the public split training set, we choose randomly selected nodes as training samples, and for validation and test sets we still follow the public split. For the co-purchased networks Amazon-Computers, we randomly select nodes as training set in each replicated experiment, construct a random validation set with 30 nodes in each class and treat the remaining nodes as the testing set. For Flickr, we follow the dataset split from Zeng et al. (2019). For Computers-Random, we build a training set of equal proportions based on the label distribution of the entire graph (Amazon-Computers). The label distribution in the training set for Computers-Random is summarized in Table 13. The details of label distribution in the training set of the five imbalanced benchmark datasets are in Table 13, and the label distribution of the full graph is provided in Table 14.

D.2. Details of GNNs

We evaluate our method with three classic GNN architectures, namely GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), and GraphSAGE (Hamilton et al., 2017). GNN consists of $L = 1, 2, 3$ layers, and each GNN layer is followed by a BatchNorm layer (momentum=0.99) and a PRelu activation (He et al., 2015). For GAT, we adopt multi-head attention with 8 heads. We search for the best model on the validation set. The choices of the hidden unit size for each layer are 64, 128, and 256.

D.3. Evaluation Protocol

We adopt Adam (Kingma & Ba, 2014) optimizer with an initial learning rate of 0.01 or 0.005. We follow (Song et al., 2022) to devise a scheduler, which cuts the learning rate by half if there is no decrease in validation loss for 100 consecutive epochs. All learnable parameters in the model adopt weight decay with a rate of 0.0005. For the first training iteration, we train the model for 200 epochs using the original training set for Cora, CiteSeer, PubMed, or Amazon-Computers. For Flickr, we train for 2000 epochs in the first iteration. We train models for 2000 epochs in the rest of the iteration with the above optimizer and scheduler. The best models are selected based on validation accuracy. Early stopping is used with patience set to 300.

D.4. Implementation details

In UNREAL, we employ the vanilla K-means algorithm as the unsupervised clustering method. The number of clusters K is chosen from $\{100, 300, 500, 700, 900\}$ for Cora, CiteSeer, PubMed, and Amaon-Computers. For Flickr, K is selected among $\{1000, 2000, 3000, 5000\}$. For Cora, CiteSeer, PubMed, and Amazon-Computers, the number of training round T are tuned among $\{40, 60, 80, 100\}$. For Flickr, T is tuned among $\{40, 50, 60, 70\}$. We also introduce a hyperparameter α, which is the upper bound on the number of nodes being added per class per round. The tuning range of α is $\{4, 6, 8, 10\}$ for Cora, CiteSeer, Amazon-Computers and $\{64, 72, 80\}$ for PubMed. For Flickr the value of α is selected among $\{30, 40, 50, 60\}$. The weight parameters p in RBO is selected among $\{0.5, 0.75, 0.98\}$, and the threshold in DGIN is tuned among $\{0.25, 0.5, 0.75, 1.00\}$. For Flickr, we only add minority nodes to the training set in all iterations, which means that we set α = 0 for majority classes in Flickr.

Table 13. Label distributions in the training sets

Dataset	$\mathcal{C}_0$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_9$
Cora (ρ = 10)	**20** (23.26%)	**20** (23.26%)	**20** (23.26%)	**20** (23.26%)	**2** (23.26%)	**2** (23.26%)	**2** (23.26%)	-	-	-
Cora (ρ = 20)	**20** (24.10%)	**20** (24.10%)	**20** (24.10%)	**20** (24.10%)	**1** (1.19%)	**1** (1.19%)	**1** (1.19%)	-	-	-
Cora (ρ = 50)	**50** (24.63%)	**50** (24.63%)	**50** (24.63%)	**50** (24.63%)	**1** (0.49%)	**1** (0.49%)	**1** (0.49%)	-	-	-
Cora (ρ = 100)	**100** (24.81%)	**100** (24.81%)	**100** (24.81%)	**100** (24.81%)	**1** (0.25%)	**1** (0.25%)	**1** (0.25%)	-	-	-
CiteSeer (ρ = 10)	**20** (30.30%)	**20** (30.30%)	**20** (30.30%)	**2** (30.30%)	**2** (3.03%)	**2** (3.03%)	-	-	-	-
CiteSeer (ρ = 20)	**20** (31.75%)	**20** (31.75%)	**20** (31.75%)	**1** (1.59%)	**1** (1.59%)	**1** (1.59%)	-	-	-	-
CiteSeer (ρ = 50)	**50** (32.68%)	**50** (32.68%)	**50** (32.68%)	**1** (0.65%)	**1** (0.65%)	**1** (0.65%)	-	-	-	-
CiteSeer (ρ = 100)	**100** (33.00%)	**100** (33.00%)	**100** (33.00%)	**1** (0.33%)	**1** (0.33%)	**1** (0.33%)	-	-	-	-
PubMed (ρ = 10)	**20** (47.62%)	**20** (47.62%)	**2** (4.76%)	-	-	-	-	-	-	-
PubMed (ρ = 20)	**20** (48.78%)	**20** (48.78%)	**1** (2.44%)	-	-	-	-	-	-	-
PubMed (ρ = 50)	**50** (49.50%)	**50** (49.50%)	**1** (0.99%)	-	-	-	-	-	-	-
PubMed (ρ = 100)	**100** (49.75%)	**100** (49.75%)	**1** (0.50%)	-	-	-	-	-	-	-
Amazon-Computers (ρ = 10)	**20** (18.18%)	**20** (18.18%)	**20** (18.18%)	**20** (18.18%)	**20** (18.18%)	**2** (1.82%)	**2** (1.82%)	**2** (1.82%)	**2** (1.82%)	**2** (1.82%)
Amzon-Computers (ρ = 20)	**20** (19.05%)	**20** (19.05%)	**20** (19.05%)	**20** (19.05%)	**20** (19.05%)	**1** (0.95%)	**1** (0.95%)	**1** (0.95%)	**1** (0.95%)	**1** (0.95%)
Amzon-Computers (ρ = 50)	**50** (19.61%)	**50** (19.61%)	**50** (19.61%)	**50** (19.61%)	**50** (19.61%)	**1** (0.39%)	**1** (0.39%)	**1** (0.39%)	**1** (0.39%)	**1** (0.39%)
Amzon-Computers (ρ = 100)	**100** (19.80%)	**100** (19.80%)	**100** (19.80%)	**100** (19.80%)	**100** (19.80%)	**1** (0.20%)	**1** (0.20%)	**1** (0.20%)	**1** (0.20%)	**1** (0.20%)
Computers-Random (ρ = 25.50)	**4** (3.01%)	**21** (15.79%)	**14** (10.53%)	**5** (3.76%)	**51** (38.35%)	**3** (2.26%)	**4** (3.01%)	**8** (6.02%)	**21** (15.79%)	**2** (1.50%)
Flickr ($\rho \approx$ 10.80)	**2628** (5.89%)	**4321** (9.68%)	**3164** (7.09%)	**2431** (5.45%)	**11525** (25.83%)	**1742** (3.90%)	**18814** (42.16%)	-	-	-

Table 14. Label distributions on the whole graphs

Dataset	$\mathcal{C}_0$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_9$
Cora ($\rho \approx 4.54$)	**351**	**217**	**418**	**818**	**426**	**298**	**180**	-	-	-
CiteSeer ($\rho \approx 2.66$)	**264**	**590**	**668**	**701**	**696**	**508**	-	-	-	-
PubMed ($\rho \approx 1.91$)	**4103**	**7739**	**7835**	-	-	-	-	-	-	-
Amazon-Computers ($\rho \approx 17.73$)	**436**	**2142**	**1414**	**542**	**5158**	**308**	**487**	**818**	**2156**	**291**
Flickr ($\rho \approx 10.84$)	**5264**	**8506**	**6413**	**4903**	**22966**	**3479**	**37719**	-	-	-

D.5. Baselines

For GraphSMOTE (Zhao et al., 2021), we use the branched algorithms whose edge predictions are discrete-valued, which have achieved superior performance over other variants in most experiments. For the ReNode method (Chen et al., 2021), we search hyperparameters among lower bound of cosine annealing $w_{\min} \in \{0.25, 0.5, 0.75\}$ and upper bound of the cosine annealing $w_{\max} \in \{1.25, 1.5, 1.75\}$ following Chen et al. (2021). PageRank teleport probability is fixed as $\alpha = 0.15$, which is the default setting in the released codes. For TAM (Song et al., 2022), we search the best hyperparameters among the coefficient of ACM term $\alpha \in \{1.25, 1.5, 1.75\}$, the coefficient of ADM term $\beta \in \{0.125, 0.25, 0.5\}$, and the minimum temperature of class-wise temperature $\phi \in \{0.8, 1.2\}$ following Song et al. (2022). The sensitivity to imbalance ratio of class-wise temperature δ is fixed as 0.4 for all main experiments. Following (Song et al., 2022), we adopt a warmup for 5 iterations since we utilize model prediction for unlabeled nodes.

D.6. Configuration

All the algorithms and models are implemented in Python and PyTorch Geometric. Experiments are conducted on a server with an NVIDIA 3090 GPU (24 GB memory) and an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz.

E. Algorithm

Algorithm 1 *UNREAL*

```
Input: Imbalanced dataset ($\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}_0), y$), feature matrix $\mathcal{X}$, adjacency matrix $\mathcal{A}$, unlabeled set $\mathcal{U} = \mathcal{V} - \mathcal{L}_0$, rounds $T$ to select nodes, the size threshold $\alpha$ of nodes being added in each class per round, weight hyperparameter $p$ of RBO, threshold $\gamma$ of DGIN, learning rate $\eta$, the size $k'$ of the clusters, GNN model $f_{\text{g}}$, clustering algorithm $f_{\text{cluster}}$, and the mean function $M(\cdot)$.
 1: for $i = 0$ to round $T$ do
 2:     Train $f_{\text{g}}$ based on the current training set $\mathcal{L}_i := \{\mathcal{C}_1, \cdots, \mathcal{C}_k\}$.
 3:     Obtain node embedding matrix of the labeled node set and unlabeled node set $H^L \in \mathbb{R}^{|\mathcal{L}| \times d}$,$H^U \in \mathbb{R}^{|\mathcal{U}| \times d}$, prediction $\hat{y}$ and confidence $r$ from the classifier.
 4:     % Step 1: Dual Pseudo-tag Alignment Mechanism(DPAM)
 5:     $f_{\text{cluster}}(H^U) \Longrightarrow \{\mathcal{K}_1, c_1, \mathcal{K}_2, c_2, \cdots, \mathcal{K}_{k'}, c_{k'}\}$
 6:     $c_i^{\text{train}} = M(\{h_u^L \mid y_u \in \mathcal{C}_i\})$
 7:     Assign a label $\tilde{y}_m$ to each cluster $\mathcal{K}_m$: $\tilde{y}_m = \arg\min_j \mathsf{distance}(c_j^{\text{train}}, c_m)$.
 8:     Combine clusters with the same pseudo-label $m$ as $\tilde{\mathcal{U}}_m$, and $\mathcal{U} = \bigcup_{m=1}^{k} \tilde{\mathcal{U}}_m$.
 9:     Put unlabeled nodes whose prediction in $\hat{y}$ is $m$ into the set $\mathcal{U}_m$, and $\mathcal{U} = \bigcup_{m=1}^{k} \mathcal{U}_m$.
10:     % Step 2: Node-Reordering
11:     For each $u \in \tilde{\mathcal{U}}_m \cap \mathcal{U}_m$: $\delta_u = \mathsf{distance}\ (h_u^L, c_m^{\text{train}})$.
12:     Obtain geometric rankings $\{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_k\}$ based on $\delta$; and confidence rankings $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_k\}$ based on $r$.
13:     For each $m$, $\mathcal{N}_m^{New} = \max\{r_m, 1 - r_m\} \cdot \mathcal{S}_m + \min\{r_m, 1 - r_m\} \cdot \mathcal{T}_m$.
14:     Select nodes based on the rank of their values in $\mathcal{N}_m^{New}$.
15:     % Step 3:Discarding Geometrically Imbalanced Nodes (DGIN)
16:     Obtain the distance between the embedding of $u$ and the second closest center to $u$ as $\beta_u$, compute GI index of node $u$ as $\frac{\beta_u - \delta_u}{\delta_u}$.
17:     if $\frac{\beta_u - \delta_u}{\delta_u} < \gamma$ then
18:         Discard node $u$.
19:     else
20:         Select it to the training set.
21:     end if
22:     Update the training set $\mathcal{L}_i$.
23: end for
```

F. Notation

Table 15. Elaborated notation table of this paper.

Indices	
n	The number of nodes,$\lvert\mathcal{V}\rvert$.
f	The node feature dimension.
k	The number of different classes.
k'	The number of cluster centers in the embedding space.
d	The dimension of the embedding space, or the dimension of the last layer of GNNs.
T	Rounds to select nodes
Parameters	
$\mathcal{G}$	An undirected and unweighted graph.
$\mathcal{V}$	The node set of $\mathcal{G}$.
$\mathcal{E}$	The edge set of $\mathcal{G}$.
$\mathcal{X}$	The feature matrix of $\mathcal{G}$, $\mathcal{X} \in \mathbb{R}^{n\times f}$.
$\mathcal{L}$	The set of labeled nodes of $\mathcal{G}$.
A	The adjacency matrix of $\mathcal{G}$, $A \in \{0,1\}^{n\times n}$.
$\mathcal{N}(v)$	The set of 1-hop neighbors for node v.
$\mathcal{U}$	The set of unlabeled nodes, $\mathcal{U} = \mathcal{V} - \mathcal{L}$.
$\mathcal{C}_i$	The i class of the labeled sets.
ρ	Imbalance ratio of a dataset, $\rho := \frac{\max_i(\lvert\mathcal{C}_i\rvert)}{\min_i(\lvert\mathcal{C}_i\rvert)}$.
h_v^l	The feature of node v in the l-th layer.
$e_{v,u}$	The edge weight between v and u.
Φ^l	The parameter matrix of the l-th layer.
H^L	The embedding matrix of labeled nodes, $H^L \in \mathbb{R}^{\lvert\mathcal{L}\rvert\times d}$.
H^U	The embedding matrix of unlabeled nodes, $H^U \in \mathbb{R}^{\lvert\mathcal{U}\rvert\times d}$.
h_u^L	The embedding of a node u, if $u \in \mathcal{L}$.
h_u^U	The embedding of a node u, if $u \in \mathcal{U}$.
$\mathcal{K}_i$	The i-th cluster.
c_i	The i-th cluster center,the center of cluster i-th .
$\tilde{y}_i$	The pseudo-label of the cluster $\mathcal{K}_i$.
$\tilde{\mathcal{U}}_m$	The combination of clusters with the same pseudo-label m.
$\hat{y}_u$	The prediction of node u in $\mathcal{U}$ given by GNN model.
$\mathcal{U}_m$	The combination of unlabeled nodes whose prediction given by the GNN model is m.
$\mathcal{Z}$	The pool of candidate nodes after DPAM, $\mathcal{Z} = \bigcup_{i=m}^{k} (\tilde{\mathcal{U}}_m \cap \mathcal{U}_m)$.
c_m^{train}	The class center of class m in the embedding space.
$\mathcal{S}_i$	The sorted lists of geometric rankings.
$\mathcal{T}_i$	The sorted lists of confidence rankings.
r_m	The similarity between two rankings, $r_m = \text{RBO}(\mathcal{S}_m, \mathcal{T}_m)$.
δ_u	The distance between the embedding of u and the closest class center to u.
β_u	The distance between the embedding of u and the second closest class center to u.
γ	Threshold of DGI.
p	Weight hyperparameter of RBO.
α	The size threshold of nodes being added in each class per round.
η	Learning rate of GNN model.
Functions.	
m_l	The message function of MPNNs.
θ_l	The information aggregation function.
ψ_l	The node feature update function.
f_{cluster}	An unsupervised clustering algorithm for the embedding space.
$M(\cdot)$	The mean function.
f_{g}	GNN model.