GRAPHENS: NEIGHBOR-AWARE EGO NETWORK SYNTHESIS FOR CLASS-IMBALANCED NODE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In many real-world node classification scenarios, nodes are highly class-imbalanced, where graph neural networks (GNNs) can be readily biased to major class instances. Albeit existing class imbalance approaches in other domains can alleviate this issue to some extent, they do not consider the impact of message passing between nodes. In this paper, we hypothesize that overfitting to the neighbor sets of minor class due to message passing is a major challenge for class-imbalanced node classification. To tackle this issue, we propose *GraphENS*, a novel augmentation method that synthesizes the whole ego network for minor class (minor node and its one-hop neighbors) by combining two different ego networks based on their similarity. Additionally, we introduce a saliency-based node mixing method to exploit the abundant class-generic attributes of other nodes while blocking the injection of class-specific features. Our approach consistently outperforms the baselines over multiple node classification benchmark datasets and architectures.

1 INTRODUCTION

Node classification for graphs has attracted significant attention as the importance of large-scale graphs analysis increases in various domains such as bioinformatics and commercial graphs to name a few (Perozzi et al., 2016; Hamilton et al., 2017; Ying et al., 2018; Mohammadrezaei et al., 2018). For example, in retail services, acquiring the qualitative node representations for items or customers is critical for improving the quality of recommendation systems (Perozzi et al., 2016; Ying et al., 2018). Detecting abnormal users in social networks, as another example, is also closely related to classifying the property of each node (Mohammadrezaei et al., 2018). Recently, Graph Neural Networks (GNNs) have demonstrated their effectiveness on learning node representations (Hamilton et al., 2017; Kipf & Welling, 2017; Velickovic et al.). However, the nodes in many real-world graphs are sometimes class-imbalanced (Mohammadrezaei et al., 2018; Wang et al., 2020a), hence GNNs are prone to be biased toward major classes, as in general class-imbalance tasks. This bias forces networks to poorly classify the nodes of minor classes, resulting in destructive impacts and a large cost to their services.

While the peculiar characteristics of imbalanced node classification and specialized solutions suitable for it have hardly been investigated, applying generic imbalance handling methods (Chawla et al., 2002; Cui et al., 2019; Cao et al., 2019; Kang et al., 2020; Menon et al., 2021) directly to the graph domain has several non-trivial challenges. One of the distinct natures of graph data is that adjacent nodes are involved in constructing the representation of each node, which makes the model more confused to learn the unbiased representation of minor class nodes. Here, we hypothesize that it is in fact more serious to overfitting to neighbors of minor nodes than to overfitting to the node feature itself. This 'neighbor memorization' is the critical obstacle to naively adopt the class-imbalance approaches of other domains such as re-weighting and re-sampling used in image classification. Specifically, re-weighting approaches (Cui et al., 2019; Tan et al., 2020; Cao et al., 2019; Menon et al., 2021; Hong et al., 2021), applying penalties according to the number of data, simply assign large weight to minor nodes, hence there is no change in neighbor sets of minor nodes observed during training. Re-sampling methods (Chawla et al., 2002; Kang et al., 2020; Zhang et al., 2021; Wang et al., 2021), sampling data to balance the number of data for each class, are also vulner-

able to overfit on minor nodes with their neighbors. Another challenge of the re-sampling method, especially for oversampling variants, is determining how to connect the newly sampled nodes to the original graph. For example, simply connecting an oversampled node with all neighbors of original node will change the edges of neighbor nodes as well and hence significantly alters the message passing to the neighbors, which might impair their class semantics. To mitigate this issue, GraphSMOTE (Zhao et al., 2021) exploits edge predictor to decide the connectivity with the neighbors of two minor nodes used in oversampling. Nevertheless, GNNs trained with GraphSMOTE still suffer from neighbor memorization when the number of minor nodes is limited.

In this paper, we propose *GraphENS*, a novel augmentation approach that synthesizes the *whole* ego network for minor class (minor node and its one-hop neighbors) by interpolating two different ego networks in the data; to enlarge the limited neighbor views of minor instances in the data, our method combines the ego network of anchoring minor node with that of randomly selected node from all classes, where it interpolates ego networks based on KL-divergence between model predictions of ego networks in order to keep the semantics of minor classes. Synthesized ego networks are attached to the original graph to construct a class-balanced graph, and GNNs are trained with the enlarged graph. *GraphENS* enables the model to learn the minor class nodes with feasible neighbors by generating the virtual ego networks.

To further prevent the synthesis of deleterious ego networks, we introduce a saliency-based node mixing approach to generate the central node of ego network. Our method separates class-generic node features from class-specific node features by using saliency information of each feature, and exploits only class-generic attributes to combine with node feature of anchoring minor node. We validate our method on various benchmark datasets including citation networks (Sen et al., 2008), and Amazon product co-purchasing networks (Shchur et al., 2018) with diverse architectures such as GCN (Kipf & Welling, 2017), GAT (Velickovic et al.), and GraphSAGE (Hamilton et al., 2017), and confirm that our approach consistently outperforms baseline methods over various settings.

In summary, our contribution is threefold:

- We demonstrate that in class-imbalanced node classification, GNNs severely overfit to neighbor sets of minor class nodes, rather than to minor nodes themselves. This 'neighbor memorization' problem becomes severe when the number of minor nodes is extremely small.
- Our method effectively alleviates the neighbor memorization problem in class-imbalanced graphs by synthesizing feasible ego networks based on the similarity between source ego networks. We also block the injection of harmful features in generating the mixed nodes using node feature saliency.
- Through extensive experiments, we show that our approach consistently outperforms baselines on multiple benchmark datasets including real-world imbalanced datasets. Even in highly imbalanced synthetic graphs, our method exhibits superior performance.

2 Related Work and Preliminary

2.1 Class Imbalance Problem

The goal of class-imbalance handling in classification is to construct an unbiased classifier to the label distribution of the training set. There are three main streams: loss modification, post-hoc correction, and re-sampling approaches. Loss modification methods alter the objective function to assign more weights (Japkowicz & Stephen, 2002; Cui et al., 2019) or margins (Tan et al., 2020; Cao et al., 2019; Menon et al., 2021) on minor classes. Post-hoc correction strategies (Kang et al., 2020; Tian et al., 2020; Menon et al., 2021; Hong et al., 2021) remedy logits to compensate minor classes in the inference. Re-sampling approaches augment minor class data by sampling strategies (Kang et al., 2020; Liu et al., 2020; Ren et al., 2020) or generation (Chawla et al., 2002; Kim et al., 2020; Chu et al., 2020; Zhang et al., 2021; Wang et al., 2021). Among minor class generation approaches, SMOTE (Chawla et al., 2002) is a widely used method to mix minor data with the nearest data of the identical class. Synthesizing minor class data from data of other classes (Kim et al., 2020; Chu et al., 2020; Wang et al., 2021) is introduced to exploit the rich information of other classes. Kim et al. (2020) produces new minor class data by taking gradient steps to translate major class data into

minor class data. Wang et al. (2021) synthesizes minor class data by combining features of minor class data with feature displacements of other data. To extend these approaches to the graph domain, structural aspects of graph have to be considered when generating minor instances.

In node classification, imbalance handling works (Zhou et al., 2018; Wang et al., 2020b; Shi et al., 2020; Zhao et al., 2021; Qu et al., 2021) are proposed to exploit structural information in graphs. DR-GCN (Shi et al., 2020) produces the virtual minor nodes generated by additional conditional GAN and regularizes the features of virtual nodes close to adjacent nodes. GraphSMOTE (Zhao et al., 2021) generates synthetic minor nodes by interpolating two minor class nodes and a (pre-trained) edge predictor determines the connectivity of synthetized nodes between synthetized nodes and neighbors of two source minor nodes. ImGAGN (Qu et al., 2021) synthesizes minor nodes by interpolating features among whole minor nodes with the generated weight matrix. Then the synthesized nodes are connected to the original minor nodes if weights in the matrix are larger than a fixed threshold. As GraphSMOTE and ImGAGN only utilize nodes of the identical minor class to generate minor nodes, the sample diversity of synthesized nodes would be significantly constrained. Moreover, ImGAGN mainly targets binary classification and its extension to multi-class classification is non-trivial since an independent generator is required per each class. Compared to these approaches, our GraphENS utilizes whole nodes to synthesize minor nodes, thus our method outperforms baselines when the number of minor classes is low in Section 5.3 (Table 4).

2.2 Graph Neural Networks for Node Classification

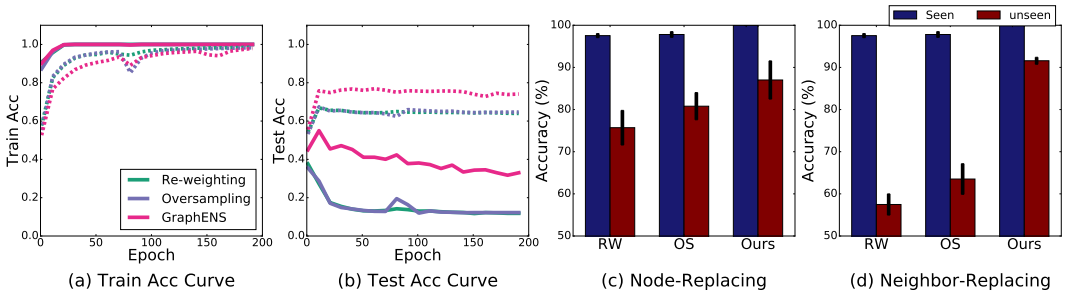
We briefly introduce GNNs for node classification tasks. Let us first define graph $G(V, E)$ where V is the set of nodes and E is the set of edges between two nodes. Let $X \in \mathbb{R}^{|V| \times d}$ be the node features whose the i-th row represents the d-dimensional feature of the i-th node. $\mathcal{N}(v)$ is the set of adjacent nodes that are directly connected to v: $\{u \in V | \{u, v\} \in E\}$. In node classification, each node in the graph corresponds to a class $y \in \{1, \dots, C\}$ where C is the number of classes.

In this paper, we consider several variants of GNNs that consists of three differentiable functions: 1) message function m_l, 2) permutation invariant message aggregation function ϕ_l, and 3) node update function h_l. Let $x_v^{(l)}$ be the latent vector of node v at layer l. To simplify notations for the recursive definition of GNNs, we use $x_v^{(0)}$ to denote the input node feature. At each GNN layer, node features are updated as $x_v^{(l+1)} = h_l(x_v^{(l)}, \phi_l(\{m_l(x_v^{(l)}, x_u^{(l)}, e_{v,u}) | u \in \mathcal{N}(v)\})$ to consider adjacent node embeddings. By passing these aggregated node features to a linear classifier as input, we obtain the prediction $\hat{y} = f(x_v)$ for node v where $\hat{y}_c = P(y = c | x_v)$. As a representative example, a layer of Graph Convolutional Network (GCN) (Kipf & Welling, 2017) is defined as $x_v^{(l+1)} = \Theta \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{e_{v,u}}{\sqrt{\hat{d}_u \hat{d}_v}} x_u^{(l)}$ where $\hat{d}_v = 1 + \sum_{u \in \mathcal{N}(v)} e_{v,u}$ with $e_{v,u}$ is the edge weight of edge $\{u, v\} \in E$ and Θ is a filter parameters. There are several variants depending on how these key components are designed, but they are beyond scope of our work.

3 Neighbor Memorization Problem

In this section, we define and investigate the *neighbor memorization problem* uniquely appearing in the node classification task of a class-imbalanced graph. Conventional imbalance handling approaches such as re-weighting and oversampling enable the model to train in a class-balanced manner. However, these methods are vulnerable to overfit toward minor classes (Zhou et al., 2020) since they solely assign large weights on limited minor instances in the training phase. In the node classification task, the unique characteristics of this task, which have not been considered until now, should be considered: not only overfitting to the minor node feature v_{minor}, but also overfitting to $\mathcal{N}(v_{minor})$ can be an issue. That is, the model is exposed to highly restricted neighboring structures for minor classes. Here, since GNN models are based on message passing that receives information from (a number of) neighboring nodes, we hypothesize that the overfitting and memorizing $\mathcal{N}(v_{minor})$ can be more serious than overfitting to v_{minor}, and we experimentally verify that below.

Experimental setup We first conduct experiments to observe overfitting to minor classes on class-imbalanced graphs. GNNs are trained with two widely used methods - re-weighting and over-sampling on a node classification benchmark dataset, PubMed (Sen et al., 2008). We process the



(a) Train Acc Curve (b) Test Acc Curve (c) Node-Replacing (d) Neighbor-Replacing

Figure 1: The results for overfitting to minor classes, node memorization, and neighbor memorization. Dash lines are learning curves including all classes and solid lines are learning curves for the minor class in (a) and (b). In (c), the blue/red bars indicate the accuracy of seen/unseen nodes with seen neighbors in the node-replacing experiments. In (d), the blue/red bars denote the accuracy of seen nodes with seen/unseen neighbors in the neighbor-replacing experiments. Note that RW and OS represent re-weighting and oversampling.

PubMed dataset to follow a long-tailed distribution by well-known construction rule in the vision domain (Cui et al., 2019). The imbalance ratio between the numbers of the most frequent class and the least frequent class is set as 100. We fix architecture as the 2-layer GraphSAGE having 256 hidden dimension and train models for 2000 epochs. We select the model by the validation accuracy.

Overfitting to minor classes The learning curves of imbalance handling approaches are presented in Figure 1 (a) and (b). The solid lines and the dash lines are the curves of minor class accuracy and overall accuracy, respectively. We observe that the test accuracy of the minor class is much lower than the overall accuracy while the train accuracy of the minor class is extremely high. This result implies that the existing approaches are prone to memorize minor class nodes. Note that we only present the learning curves for the first 200 epochs to show the tendency of the early phase more clearly. We confirm the test accuracy differences between minor classes and all classes are almost maintained until the last epoch for both methods. The learning curve for entire epochs is in Appendix A.1 (Figure 4). In the following paragraph, we investigate more specifically whether this overfitting is mainly due to memorizing *node features* or their *neighbor structures*.

Neighbor memorization problem To investigate which kinds of information GNN is prone to be more overfitting - self node features or aggregated features from neighbors, we design two 'replacing' experiments. In the first replacing experiment, we scrutinize 'node memorization' where GNNs memorize seen minor node features excessively. We intentionally replace a seen minor node with an unseen minor node given a fixed seen neighbors (node-replacing experiment). Toward this, we randomly sample three nodes from the identical minor class: we sample v_{anchor} (recipient), v_{seen} (donator) from the training set and v_{unseen} (donator) from the test set. We then replace the feature of v_{anchor} with that of v_{unseen} in the original graph, and evaluate the classification accuracy of replaced v_{anchor}. That is, the replaced feature of v_{anchor} is not exposed during training, but its neighbor view does. For comparison, we replace the feature of v_{anchor} with of v_{seen} as a baseline; it is the case of seen node features and neighbors but it has the artifact of grafting. We measure the accuracy for all minor test nodes and experiments are repeated 50 times. In Figure 1(c), the blue and red bars represent the results of v_{anchor} with seen node features and v_{anchor} with unseen node features, respectively.

Now, we conduct a similar experiment for 'neighbor memorization'. We would like to estimate how much GNN relies on seen topological structures. To this end, we measure the performance degradation when the seen neighbor set is replaced by an unseen neighbor set given a fixed seen self-node feature (neighbor-replacing experiment). We sample v_{seen} (donator) and v_{anchor} (recipient) from the training set and v_{unseen} (recipient) from the test set as before, but v_{anchor} now has a different role compared to the node-replacing experiment: it serves as providing a seen neighbor set. Specifically, we replace the feature of v_{unseen} with that of v_{seen}, which represents the situation where a seen node is embedded into unseen neighbors. Then, we evaluate the classification accuracy of replaced v_{unseen}. For a baseline, the node feature of v_{anchor} is replaced by that of v_{seen} and measures the accuracy of replaced v_{anchor}; it is the case of seen node feature with a seen neighbor set. In Figure 1(d), the blue bar denotes the performance of seen node feature with seen neighbor set

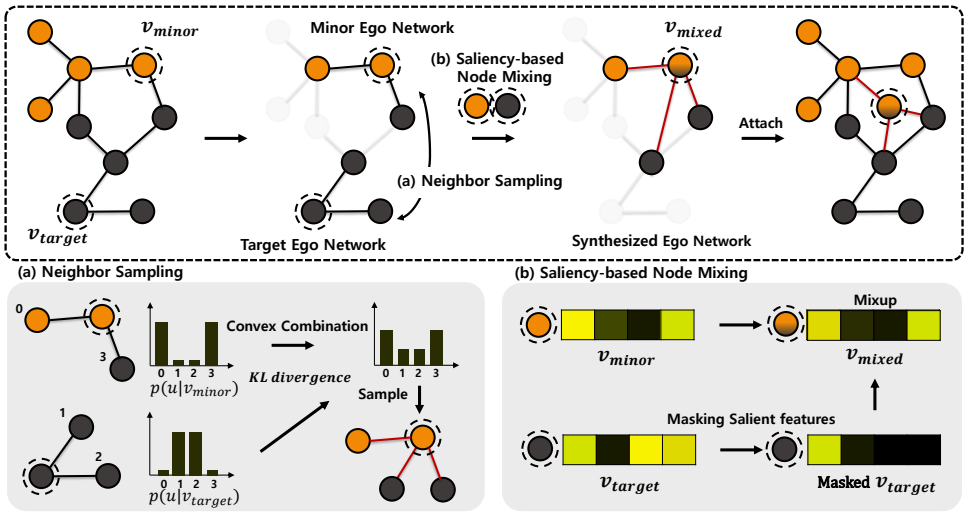


Figure 2: Overall pipeline of *GraphENS*

and the red bar represents the performance of seen node feature with an unseen neighbor set. The overall procedure is described by Figure 6 in Appendix B.

We demonstrate that conventional imbalance handling approaches severely suffer from a neighbor memorization problem in Figure 1(c) and (d)[1]. The performance drop of conventional algorithms in the neighbor-replacing experiment is almost four times steeper than in the node-replacing setting. These results imply that the neighbor memorization problem is a critical obstacle in properly handling the class-imbalanced problem in node classification tasks.

From the learning curves and replacing experiments, we find that conventional approaches largely rely on the neighbor information compared to the node features. This neighbor memorization problem results in poor generalization performance on the minor class, so it must be appropriately addressed in class-imbalanced node classification. Recent work, GraphSMOTE (Zhao et al., 2021), connects a mixed node to neighbors of both the source node and target node in the same class, and then removes the unreliable edges by the feature similarity-based edge predictor. However, since GraphSMOTE only utilizes the neighbors of the nodes in the identical class, it still prone to memorize the neighbors of minor classes when the number of minor class nodes is extremely limited.

4 GraphENS

We now describe our novel class-imbalance handling strategy for node classification, *GraphENS* that synthesizes an ego network for generated minor nodes considering the prediction similarity of the ego network instances, in order to alleviate the neighbor memorization described above. *GraphENS* is mainly composed of two core components. Both components exploit the features from two nodes: the minor class node v_{minor} and the target node v_{target} from *entire classes*[2] to synthesize an ego network of minor class. The first component is to augment the neighbor set by combining neighbors of a minor node v_{minor} with those of the target node v_{target}. To prevent the generation of harmful ego networks, *GraphENS* regulates the combination ratio depending on the KL divergence between ego networks centered on v_{minor} and v_{target}, respectively (Section 4.1). The second component of *GraphENS* defines the node feature saliency for GNNs that represents the importance of each node feature in classifying the node property. Equipped with this node feature saliency, *GraphENS* mixes v_{minor} and v_{target} to generate a central node and combine with the ego network synthesized above (Section 4.2). This feature saliency information is utilized to filter out the class-specific node features in the node feature mixing process to preserve the semantics of the minor class. The overall procedure of *GraphENS* is described in Figure 2. Our full algorithm is also provided in the Appendix C (Algorithm 1). We discuss the details of each component in the following subsections.

[1]To reduce the negative effect of oversampling on message-passing, we make the message passing work only on incoming edges to the oversampled nodes.

[2]Detailed explanation and justification for this strategy for target node selection are discussed in Section 5.2.

4.1 NEIGHBOR SAMPLING

Neighbor sampling is a method to determine which nodes to be connected to a new mixed node, v_{mixed} (how to synthesize a mixed node is described in Section 4.2). The goal of neighbor sampling is to expose minor nodes to various environments to mitigate neighbor memorization problem while preventing the generation of harmful connections. To provide diverse neighbor sets, we construct 'adjacent node distribution' for a mixed node from two ego networks of v_{minor} and v_{target} and stochastically sample the neighbors from this distribution. At the same time, regarding the avoidance of detrimental connectivity, we devise a scheme that reflects $\mathcal{N}(v_{target})$ more in constructing the adjacent node distribution when the ego network of v_{target} is more similar to that of v_{minor}. Note that the ego network G_v of node v consists of $\{\mathcal{N}_v \cup v\}$ nodes and edges directly connected to the central node v. The overall scheme of neighbor sampling is in Figure 2 (a).

Constructing adjacent node distribution Now, we build the similarity-aware adjacent node distribution for v_{mixed} by mixing adjacent node distributions of v_{minor} and v_{target}. Here, the adjacency node distribution for a real node v, $p(u|v)$, is defined as $p(u|v) = \frac{1}{|\mathcal{N}(v)|}$ if $u \in V, \{u, v\} \in E$, and $p(u|v) = 0$ otherwise. To construct a 'probable' adjacent node distribution, the mixing ratio is determined based on the similarity between the ego networks of $G_{v_{minor}}$ and $G_{v_{target}}$. Toward this, we utilize logits $o_v = f(G)_v$ for a node v given the full graph G as input of GNN f since GNN aggregates neighbor information with self node feature. It is worth noting that the logits can be obtained without additional cost (we can bring the model confidences at the previous epoch). To further reflect the neighbor information explicitly and avoid the overconfidence problem, we aggregate adjacent logits and utilize these logits to compute the similarity. Specifically, the aggregated logit of node v is computed as $\hat{o}_v = \frac{1}{|\mathcal{N}(v)|+1} \sum_{u \in (\mathcal{N}(v) \cup v)} o_u$. Then, we calculate the KL-divergence between $\hat{o}_{minor}$ and $\hat{o}_{target}$ to estimate the distance between $G_{v_{minor}}$ and $G_{v_{target}}$: $\phi = KL\left(\sigma(\hat{o}_{minor}) \| \sigma(\hat{o}_{target})\right)$ where σ is a softmax function. Now, the normalized distance is used as the mixing ratio for the minor adjacent node distribution, and it is calculated as $\hat{\phi} = \frac{1}{1+e^{-\phi}}$. Finally, the adjacent node distribution for the synthesized v_{mixed} is defined as

$$p(u|v_{mixed}) = \hat{\phi}\, p(u|v_{minor}) + (1 - \hat{\phi})\, p(u|v_{target}). \tag{1}$$

Note here that the mixing ratio of the target node decreases as the distance from the minor node increases and the mixing ratio of anchoring minor node is guaranteed to be at least 0.5.

Sampling neighbors Given the adjacent node distribution $p(u|v_{mixed})$ in Equation 1, we sample neighbors from this distribution without replacement. The number of neighbors is sampled from the degree distribution of a given graph to keep degree statistics with the original graph. Not only that, we make the message passing work only on the incoming edges to v_{mixed} since undirected edges affect message passing in the original graph.

4.2 SALIENCY-BASED NODE MIXING

We now describe how our method synthesizes new central node v_{mixed} of $G_{v_{mixed}}$ by interpolating two nodes v_{minor} and v_{target}. Note that our method does not restrict the target node to be from minor classes to utilize the abundant class-generic information of other classes. To extract only generic attributes independent of class, we first define the node feature saliency to identify the importance of each node feature. Armed with this node feature saliency, *GraphENS* synthesizes new minor node v_{mixed} using minor node features and generic features of target node via convex combination.

Feature saliency Albeit *GraphENS* does not depend on specific ways of calculating the importance of each node attribute, we simply adopt an approach that evaluates the importance of node features based on gradients. In particular, motivated by the works using the gradient of the loss function with respect to input features in image classifications (Simonyan et al., 2013; Shrikumar et al., 2017), we define the feature saliency as follows.

We first compute the gradient of classification loss $\mathcal{L}(G, \mathbf{y})$ with respect to the input node feature matrix $\frac{\partial \mathcal{L}}{\partial X} \in \mathbb{R}^{|V| \times d}$ via backpropagation. Then, the saliency value of i-th feature $s_{(v,i)}$ is obtained by the magnitude of the gradient regarding i-th attribute in node v as $s_{(v,i)} = \left| \left[\frac{\partial \mathcal{L}(G, \mathbf{y})}{\partial X} \right]_{(v,i)} \right|$. The

feature saliency vector $\mathcal{S}_v = [s_{(v,1)}, ..., s_{(v,d)}]^T$ is derived without any modification to the GNN architecture. Our gradient-based feature saliency can be derived without additional backpropagation as we simply exploit the gradient values computed from the previous training iteration.

Node mixup We introduce our simple mixup-like augmentation to oversample the minor nodes without label mixing. Since we select target nodes v_{target} from all classes unlike SMOTE (Chawla et al., 2002), filtering (or masking) out class-specific attributes of the target node is required to prevent the generation of noisy samples. To this end, we explicitly utilize the feature saliency information of the target node.

First, we determine the masking ratio $K\%$ using a normalized distance $\hat{\phi}$ (from Section 4.1). As $\hat{\phi}$ implies the distance between two ego networks centralized by v_{minor} and v_{major}, we adaptively assign $K\%$ proportional to normalized distance $\hat{\phi}$; $K = k\hat{\phi}$ where k is a hyperparameter. The intuition behind here is if target node v_{target} is significantly differ from minor node v_{minor}, we mask more salient features of v_{target}. Specifically, given the feature saliency vector $\mathcal{S}_{v_{target}}$, a binary mask $M_K \in \mathbb{R}^d$, which masks the $K\%$ of node attributes to 0, is sampled from multinomial distribution by applying softmax to $\mathcal{S}_{v_{target}}$. Our final synthesized minor nodes v_{mixed} can be formulated as follows given a random mixing ratio $\lambda \sim Beta(\alpha, \alpha)$:

$$v_{mixed} = (1 - \Lambda_K) \odot v_{minor} + \Lambda_K \odot v_{target}, \text{ where } \Lambda_K = \lambda \cdot M_K. \tag{2}$$

We also evaluate our method but without using saliency on multiple datasets in Section 5.4 (Table 3).

5 Experiment

5.1 Experiment setup

Datasets We validate *GraphENS* on five benchmark datasets: Cora, CiteSeer, PubMed for citation networks (Sen et al., 2008), AmazonPhoto and AmazonComputers for co-purchase graphs (Shchur et al., 2018). For citation networks, we follow splits in Yang et al. (2016). In supervised learning setting, we construct long-tailed citation networks following Cui et al. (2019) to validate models at a high imbalance ratio, which denotes the ratio between the most frequent class and the least frequent class. Nodes are removed until the class distribution follows a long-tailed distribution with keeping the connection in graphs at most. We sort the classes from major class to minor class and then remove nodes for each class started from major classes. When eliminating nodes, we remove nodes having low degrees and the corresponding edges of those nodes. We also evaluate imbanlance handling methods on natural imbalanced datasets such as AmazonPhoto and AmazonComputers. To make validation/test sets balanced, we sample the same number of nodes from each class for validation/test sets. Then, the remaining nodes are assigned to the training set. We fix the least number of train nodes for all classes as 20. In semi-supervised setting, we set the imbalance ratio as 10. The detailed setup is provided in Appendix D.5.

Baselines We test our methods over three architectures as GCN (Kipf & Welling, 2017), GAT (Velickovic et al.), and GraphSAGE (Hamilton et al., 2017). We adopt re-weight as cost-sensitive loss (Japkowicz & Stephen, 2002) inversely proportional to the number of class data. Oversampling is the approach that samples each class node until the number of each class data reaches the maximum number of class data. For oversampling, we duplicate the edges of the original node when adding an oversampled node to the original graph. cRT (Kang et al., 2020) and PC Softmax (Hong et al., 2021) are recent effective baselines for decoupling and post-hoc correction approaches, respectively. DR-GCN generates virtual minor nodes and forces virtual nodes to be similar to the neighbors of a source node. Although DR-GCN has an additional component to exploit entire unlabeled nodes for semi-supervised learning, we do not consider this component for a fair comparison. GraphSMOTE (Zhao et al., 2021) has two versions, depending on whether the predicted edges are discrete or continuous values. We adopt a discrete version of GraphSMOTE since it exhibits superior performance on multiple benchmark datasets. Implementation details of ours and description of evaluation protocol are deferred to Appendix D.

Table 1: Comparison of our method *GraphENS* with other baselines in extremely class-imbalanced settings (*Imbalance ratio:100*). We report the averaged accuracy, balanced accuracy, and F1-score with the standard errors for 5 repetitions on three benchmark datasets for node classification tasks.

	Method	Cora-*LT*			CiteSeer-*LT*			PubMed-*LT*		
		Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
GCN	Vanilla	73.66 ±0.28	62.72 ±0.39	63.70 ±0.43	53.90 ±0.70	47.32 ±0.61	43.00 ±0.70	70.76 ±0.74	57.56 ±0.59	51.88 ±0.53
	Re-Weight	75.20 ±0.19	68.79 ±0.18	69.27 ±0.26	62.56 ±0.32	55.80 ±0.28	53.74 ±0.28	77.44 ±0.21	72.80 ±0.38	73.66 ±0.27
	Oversampling	77.44 ±0.09	70.73 ±0.10	72.40 ±0.11	62.78 ±0.37	56.01 ±0.35	53.99 ±0.37	76.70 ±0.48	68.49 ±0.28	69.50 ±0.38
	cRT	76.54 ±0.22	69.26 ±0.48	70.95 ±0.50	60.60 ±0.25	54.05 ±0.22	52.36 ±0.22	75.10 ±0.23	67.52 ±0.72	68.08 ±0.85
	PC Softmax	76.42 ±0.34	71.30 ±0.45	71.24 ±0.52	65.70 ±0.42	**61.54** ±0.45	**61.49** ±0.49	76.92 ±0.26	**75.82** ±0.25	74.19 ±0.25
	DR-GCN	73.90 ±0.29	64.30 ±0.39	63.10 ±0.57	56.18 ±1.10	49.57 ±1.08	44.98 ±1.29	72.38 ±0.19	58.86 ±0.15	53.05 ±0.13
	GraphSMOTE	76.76 ±0.31	69.31 ±0.37	70.21 ±0.64	62.58 ±0.30	55.94 ±0.34	54.09 ±0.37	75.98 ±0.22	70.96 ±0.36	71.85 ±0.32
	GraphENS	**77.76** ±0.09	**72.94** ±0.15	**73.13** ±0.11	**66.92** ±0.21	60.19 ±0.21	58.67 ±0.25	**78.12** ±0.06	74.13 ±0.22	**74.58** ±0.13
GAT	Vanilla	73.60 ±0.26	62.75 ±0.37	63.53 ±0.35	56.76 ±0.39	50.15 ±0.34	46.59 ±0.44	71.26 ±0.77	58.86 ±0.82	54.91 ±1.12
	Re-Weight	77.26 ±0.09	70.97 ±0.11	71.37 ±0.33	63.54 ±0.39	56.98 ±0.35	55.30 ±0.42	78.14 ±0.09	**75.80** ±0.14	**76.07** ±0.12
	Oversampling	77.50 ±0.12	71.16 ±0.14	72.58 ±0.19	62.94 ±0.26	56.16 ±0.22	54.29 ±0.27	76.96 ±0.24	70.71 ±0.20	72.29 ±0.20
	cRT	76.74 ±0.30	69.19 ±0.52	70.60 ±0.57	60.80 ±0.53	54.18 ±0.55	52.33 ±0.70	73.76 ±0.23	65.69 ±0.25	66.40 ±0.27
	PC Softmax	76.62 ±0.15	73.14 ±0.19	72.69 ±0.19	62.20 ±0.42	59.49 ±0.40	**59.51** ±0.51	**78.70** ±0.60	75.40 ±0.72	75.96 ±0.62
	DR-GCN	74.42 ±0.55	64.17 ±1.00	64.20 ±0.67	56.74 ±0.74	50.02 ±0.75	45.82 ±1.03	71.52 ±0.25	59.18 ±1.06	54.879 ±2.33
	GraphSMOTE	76.92 ±0.31	70.03 ±0.51	70.47 ±0.55	64.04 ±0.38	57.33 ±0.39	55.43 ±0.49	77.12 ±0.49	73.59 ±1.16	74.40 ±0.72
	GraphENS	**78.10** ±0.13	**73.45** ±0.19	**73.48** ±0.19	**66.90** ±0.29	**60.20** ±0.30	58.70 ±0.26	78.24 ±0.15	74.27 ±0.35	74.68 ±0.30
GraphSAGE	Vanilla	72.08 ±0.53	61.97 ±0.67	61.97 ±0.75	50.76 ±0.46	44.56 ±0.49	40.43 ±0.93	64.54 ±0.35	53.07 ±0.55	48.80 ±1.13
	Re-Weight	74.38 ±0.40	68.24 ±0.35	69.28 ±0.36	59.60 ±0.79	52.73 ±0.91	50.12 ±1.01	69.78 ±0.38	67.06 ±1.79	65.69 ±0.88
	Oversampling	74.88 ±0.64	66.20 ±1.38	66.36 ±1.11	58.64 ±1.17	51.76 ±1.17	49.42 ±1.40	69.36 ±0.12	62.15 ±0.83	62.46 ±0.74
	cRT	73.52 ±0.45	63.65 ±0.77	64.65 ±0.72	51.10 ±0.40	45.04 ±0.39	42.02 ±0.53	67.50 ±0.28	56.76 ±0.65	54.04 ±1.73
	PC Softmax	74.04 ±0.62	65.87 ±0.75	66.96 ±0.72	61.78 ±0.28	56.98 ±0.35	**56.91** ±0.46	74.98 ±0.55	74.28 ±0.31	73.32 ±0.40
	DR-GCN	73.28 ±0.46	63.32 ±0.68	62.95 ±1.12	50.80 ±0.50	44.51 ±0.41	39.02 ±0.65	64.90 ±0.52	52.84 ±0.42	47.56 ±0.43
	GraphSMOTE	74.34 ±0.30	64.76 ±0.49	65.88 ±0.50	58.98 ±0.39	52.11 ±0.38	50.27 ±0.74	70.02 ±0.21	63.04 ±0.67	63.43 ±0.54
	GraphENS	**77.26** ±0.13	**70.07** ±0.28	**70.25** ±0.31	**63.98** ±0.38	**57.33** ±0.42	55.23 ±0.43	**79.60** ±0.19	**74.90** ±0.49	**75.83** ±0.43

Table 2: Results on AmazonPhoto and AmazonComputers in comparison to baselines. The experiment was repeated 5 times and standard errors are reported. Note that the Acc. is equal to bAcc. as the test sets are balanced.

	Method	AmazonPhoto (*Imbalance ratio: 82*)		AmazonComputers (*Imbalance ratio: 244*)	
		Acc.(bAcc.)	F1	Acc.(bAcc.)	F1
GraphSAGE	Vanilla	82.86 ±0.30	78.72 ±0.52	68.47 ±2.19	64.01 ±3.18
	Re-Weight	92.94 ±0.13	92.95 ±0.13	90.04 ±0.29	90.11 ±0.28
	Oversampling	92.46 ±0.47	92.47 ±0.48	89.79 ±0.16	89.85 ±0.17
	cRT	91.24 ±0.28	91.17 ±0.29	86.02 ±0.55	86.00 ±0.56
	PC Softmax	93.32 ±0.25	93.32 ±0.25	86.59 ±0.92	86.62 ±0.91
	GraphSMOTE	92.65 ±0.31	92.61 ±0.32	89.31 ±0.34	89.39 ±0.35
	GraphENS	**93.82** ±0.13	**93.81** ±0.12	**91.94** ±0.17	**91.94** ±0.17

Table 3: Ablation study. SM, NS, and PS denote saliency masking, neighbor sampling, and prediction similarity, respectively.

	Method	CiteSeer-*semi* (*Imbalance ratio: 10*)			AmazonPhoto (*Imbalance ratio: 82*)		
		Acc	bAcc.	F1	Acc.	bAcc.	F1
GraphSAGE	GraphENS (*w/o SM,NS*)	41.82	39.31	32.15	93.17	93.17	93.15
	GraphENS (*w/o SM*)	49.24	48.77	45.85	93.54	93.54	93.53
	GraphENS (*w/o PS*)	49.66	47.96	45.93	93.28	93.28	93.26
	GraphENS	**51.12**	**48.91**	**46.78**	**93.82**	**93.82**	**93.81**

5.2 Target Node Selection

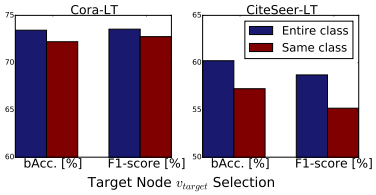


Figure 3: Target node experiments (GAT).

Our method utilizes the target node v_{target} from all classes, not only from the identical minor class of v_{minor}. The intention of constructing the target node pool from entire classes is to guarantee sufficient diversity of synthesized ego networks. If the target nodes are restricted to a minor class at highly imbalanced scenarios, identical neighbor nodes would be redundantly duplicated. This issue makes it hard to mitigate the neighbor memorization problem. To demonstrate this hypothesis, we conduct an experiment and compare our design choice (all classes) to selecting a target from the identical class as the minor node v_{minor}. In the Figure 3, we confirm that exploiting entire classes for the target node achieves superior performances. From this design choice of the target node, our method can utilize the manifold assumption that 'similar predictions of neural networks indicate the close proximity in the manifold', which are commonly utilized in semi-supervised learning (Van Engelen & Hoos, 2020). We aim to enlarge and smooth the decision boundary of the minor class by interpolating the minor nodes and the target node of the entire classes (but excluding target-specific attributes using saliency) as investigated in Verma et al. (2019). Moreover, our method considers the prediction of the ego network (not a single node) to reflect the structural aspects of graph.

To select a target node, we sample a target class from the multinomial distribution with $(logN_1, logN_2, \ldots, logN_C)$ where N_i is the number of the i-th class data and C is the number of classes. Then, we randomly select a target node from the training nodes of the sampled class.

Table 4: Comparison of our method *GraphENS* with other baselines in semi-supervised learning settings. We report the averaged accuracy, balanced accuracy, and F1-score with the standard errors for 5 repetitions.

	Method	Cora-*Semi*			CiteSeer-*Semi*			PubMed-*Semi*		
		Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
GCN	Vanilla	63.98 ±1.32	58.28 ±1.59	55.97 ±2.33	40.76 ±2.68	40.54 ±2.22	35.11 ±3.33	57.48 ±2.67	62.35 ±1.41	55.27 ±3.58
	Re-Weight	67.56 ±1.32	63.05 ±1.80	61.43 ±2.59	42.90 ±2.97	41.42 ±3.06	35.22 ±4.15	66.98 ±2.36	69.54 ±1.83	66.35 ±2.32
	Oversampling	61.86 ±1.04	56.52 ±0.86	53.43 ±1.21	36.34 ±1.70	35.93 ±1.57	27.75 ±3.19	63.76 ±1.99	68.45 ±1.64	62.37 ±2.35
	cRT	63.94 ±1.14	57.16 ±1.49	54.45 ±2.23	41.16 ±2.73	41.78 ±2.36	36.08 ±3.24	57.34 ±2.93	62.21 ±2.12	55.39 ±3.63
	PC Softmax	69.20 ±1.02	64.68 ±1.76	63.80 ±2.05	45.24 ±2.85	44.34 ±2.33	40.38 ±2.86	64.78 ±0.75	66.89 ±1.37	64.82 ±0.74
	DR-GCN	64.22 ±1.34	58.28 ±1.80	55.93 ±2.64	35.80 ±1.94	36.51 ±2.03	27.62 ±3.16	61.78 ±3.27	66.13 ±2.40	59.67 ±4.64
	GraphSMOTE	64.70 ±1.76	58.71 ±2.20	56.31 ±3.29	39.74 ±2.62	38.27 ±2.04	31.45 ±3.39	64.00 ±1.41	67.90 ±1.12	63.44 ±1.51
	GraphENS	**69.78** ±1.01	**65.01** ±1.67	**64.69** ±1.60	**52.42** ±1.09	**49.55** ±1.18	**46.27** ±1.54	**68.54** ±1.15	**70.68** ±1.22	**68.19** ±0.93

5.3 Main Results

Neighbor sampling mitigates neighbor memorization We investigate neighbor memorization over conventional algorithms in Section 3. We validate ours with respect to overfitting to minor classes, node memorization, and neighbor memorization. Although our training accuracy of the minor class is similar to re-weighting and oversampling, the test accuracy of our method is much higher than baselines, which indicates that our approach mitigates overfitting to minor classes in Figure 1 (a) and (b). Our method significantly decreases the performance degradation when seen neighbor set is replaced by an unseen neighbor set, implying that our approach can generalize well for unseen neighbor views compared to baselines (Figure 1 (d)). Neighbor sampling thus effectively alleviates the neighbor memorization by exposing minor nodes to diverse circumstances.

Supervised Learning We evaluate *GraphENS* on highly class-imbalanced situations as most real-world graphs. First, we test our method on three long-tailed citation networks: Cora-*LT*, CiteSeer-*LT*, and PubMed-*LT*. In Table 1, our method brings better imbalance handling performance in most cases compared to other baselines. To verify the effectiveness of our method on real-world imbalanced graphs, we also test GNNs trained with *GraphENS* on naturally class-imbalanced benchmark datasets without any modifications in graphs. Our method outperforms other contenders by significant margins in AmazonComputers (Table 2). Note that our method consistently exhibits higher accuracy and F1 score in other architectures (Table 5 in Appendix A.2).

Semi-supervised learning To validate our approaches when labeled nodes are extremely restricted, we conduct experiments in a semi-supervised setting on citation networks (Table 4). Our approach significantly outperforms baselines for all datasets. Ours also exhibits consistent superior performances in other GNN architectures (Table 6). Note that since *GraphENS* exploits entire classes to synthesize minor nodes, our method can avoid the generation of redundant nodes when the number of minor nodes is extremely low. Hence, our method significantly improves the performance against existing oversampling based approaches- GraphSMOTE, Oversampling. Ours brings further performance gain by utilizing unlabeled nodes to generate ego networks (Appendix A.3).

5.4 Ablation Study

We verify two key components of our method: 'prediction similarity' on neighbor sampling and 'saliency masking' on node mixing. To this end, we introduce baselines: **1)** '*GraphENS* w/o PS' determines prediction similarity uniformly random, **2)** 'w/o SM' mixes nodes without saliency masking ($M_K = 1$), and **3)** 'w/o SM,NS' generates mixed minor nodes without masking and connecting to neighbors of the target . We find that removing each module drops performance (Table 3). Thus, we believe each module effectively contributes to restrain the generation of harmful ego networks.

6 Conclusion

We investigated that existing imbalance handling algorithms suffer from neighbor memorization problem in class-imbalanced node classification. Thus, we proposed a novel augmentation method, *GraphENS*, to synthesize an ego network for minor classes with neighbor sampling and saliency-based node mixing. We verified that our method effectively mitigates neighbor memorization by synthesizing diverse but probable minor ego networks. *GraphENS* demonstrated its effectiveness in that it outperforms baselines over multiple benchmark datasets with various GNN architectures.

REFERENCES

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Aréchiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 1565–1576, 2019.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIX*, volume 12374 of *Lecture Notes in Computer Science*, pp. 694–710. Springer, 2020.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie. Class-balanced loss based on effective number of samples. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 9268–9277. Computer Vision Foundation / IEEE, 2019.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 1024–1034, 2017.

Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang. Disentangling label distribution for long-tailed visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 6626–6636. Computer Vision Foundation / IEEE, 2021.

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, 2002.

Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Jaehyung Kim, Jongheon Jeong, and Jinwoo Shin. M2m: Imbalanced classification via major-to-minor translation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 13893–13902. IEEE, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Zhining Liu, Pengfei Wei, Jing Jiang, Wei Cao, Jiang Bian, and Yi Chang. MESA: boost ensemble imbalanced learning with meta-sampler. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Mohammadreza Mohammadrezaei, Mohammad Ebrahim Shiri, and Amir Masoud Rahmani. Identifying fake accounts on social networks based on graph analysis and classification algorithms. *Secur. Commun. Networks*, 2018:5923156:1–5923156:8, 2018.

Bryan Perozzi, Michael Schueppert, Jack Saalweachter, and Mayur Thakur. When recommendation goes wrong: Anomalous link discovery in recommendation networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pp. 569–578. ACM, 2016.

Liang Qu, Huaisheng Zhu, Ruiqi Zheng, Yuhui Shi, and Hongzhi Yin. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pp. 1390–1398. ACM, 2021.

Jiawei Ren, Cunjun Yu, Shunan Sheng, Xiao Ma, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Balanced meta-softmax for long-tailed visual recognition. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29(3):93–106, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018. URL http://arxiv.org/abs/1811.05868.

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

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International Conference on Machine Learning*, pp. 3145–3153. PMLR, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, 2014.

Jingru Tan, Changbao Wang, Buyu Li, Quanquan Li, Wanli Ouyang, Changqing Yin, and Junjie Yan. Equalization loss for long-tailed object recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 11659–11668. IEEE, 2020.

Junjiao Tian, Yen-Cheng Liu, Nathaniel Glaser, Yen-Chang Hsu, and Zsolt Kira. Posterior recalibration for imbalanced datasets. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pp. 6438–6447. PMLR, 2019.

Jianfeng Wang, Thomas Lukasiewicz, Xiaolin Hu, Jianfei Cai, and Zhenghua Xu. RSG: A simple but effective module for learning imbalanced datasets. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 3784–3793. Computer Vision Foundation / IEEE, 2021.

Zheng Wang, Qiao Wang, Tanjie Zhu, and Xiaojun Ye. Extending LINE for network embedding with completely imbalanced labels. *Int. J. Data Warehous. Min.*, 16(3):20–36, 2020a.

Zheng Wang, Xiaojun Ye, Chaokun Wang, Jian Cui, and Philip Yu. Network embedding with completely-imbalanced labels. *IEEE Transactions on Knowledge and Data Engineering*, 2020b.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 40–48. JMLR.org, 2016.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 974–983. ACM, 2018.

Yongshun Zhang, Xiu-Shen Wei, Boyan Zhou, and Jianxin Wu. Bag of tricks for long-tailed visual recognition with deep convolutional neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(4):3447–3455, May 2021.

Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, March 8-12, 2021*, pp. 833–841. ACM, 2021.

Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. BBN: bilateral-branch network with cumulative learning for long-tailed visual recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 9716–9725. Computer Vision Foundation / IEEE, 2020.

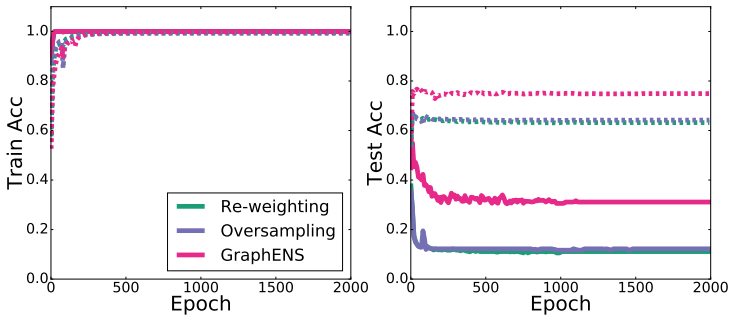
Dawei Zhou, Jingrui He, Hongxia Yang, and Wei Fan. SPARC: self-paced network representation for few-shot rare category characterization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pp. 2807–2816. ACM, 2018.

A Additional Experimental Results

In this section, we present experimental results which are not reported in the main paper due to space constraints.

A.1 Learning Curve of the Training Phase

To show the early trend of the training process, we only show the learning curve for the first 200 epochs in our main paper. We observe that the gap between training and test accuracy is maintained over entire epochs in Table 4.



(a) Train Acc Curve (b) Test Acc Curve

Figure 4: The learning curves over the training phase.

A.2 The Results of AmazonPhoto and AmazonComputers

Due to space constraints, we only present the experiment table with GraphSAGE in the main paper, which exhibits the best vanilla performances, on two co-purchase graph datasets - AmazonPhoto, AmazonComputers. We also test our method with two different GNN architectures - GCN, GAT. As shown in Table 5, *GraphENS* shows the best performances over baselines in most cases. Detailed experimental settings are described in Section D.

Interestingly, we observe that cRT and PC Softmax show inferior performance than other algorithms on AmazonComputers (Table 5). We conjecture that these approaches fail to learn discriminative representations of minor classes due to message-passing in a highly imbalanced graph. Decoupling methods such as CRT and PC Softmax are proposed based on the observation that retraining a classifier with class-balanced sampling after training on imbalanced datasets show comparable performance with baselines (Kang et al., 2020). In other words, learning distinctive representation is not significantly affected by class imbalance. However, oversmoothing caused by message passing impedes learning distinctive representation of minor classes in highly imbalanced datasets. It is consistent with the observation that decoupling methods fail at the extremely high imbalance ratio (Ren et al., 2020).

A.3 The Experiments in Semi-Supervised Learning

In Section 5.3, we report the results of GCN on three citation networks - Cora, CiteSeer, PubMed. We additionally put the result table of two different GNN architectures - GAT, GraphSAGE. emph-GraphENS outperforms baselines in most cases (Table 6).

To further improve the performance, we extend *GraphENS* to utilize unlabeled nodes and validate the extended model. First, we regard unlabeled data as the $C+1$-th class (C is the number of classes) and this class could be sampled when selecting a target node. Since the number of unlabeled data is large, the labeled class could be sampled less frequently if we utilize the number of unlabeled data in the multinomial distribution in Section 5.2. Thus, we treat the unlabeled class as the class having the same number of data with the least frequent class and sample a target class from the distribution with $(logN_1, logN_2, \ldots, logN_C, logN_{C+1})$ where N_i is the number of the i-th class data. Note that we utilize model prediction on unlabeled nodes for neighbor sampling and saliency maps of

Table 5: Accuracy/F1-score on AmazonPhoto and AmazonComputers benchmark datasets in comparison to other baselines. The experiment was performed five times and the averaged accuracy and F1-score with standard error are reported. Note that the accuracy is equal to balance accuracy since the test sets are balanced.

	Method	AmazonPhoto (*Imbalance ratio: 82*)		AmazonComputers (*Imbalance ratio: 244*)	
		Acc.(bAcc.)	F1	Acc.(bAcc.)	F1
GCN	Vanilla	81.91 ±0.25	77.50 ±0.28	58.42 ±0.29	51.21 ±0.39
GCN	Re-Weight	92.73 ±0.12	92.72 ±0.13	90.65 ±0.25	90.70 ±0.23
GCN	Oversampling	92.62 ±0.23	92.61 ±0.23	90.60 ±0.26	90.67 ±0.26
GCN	cRT	90.43 ±0.16	90.20 ±0.21	78.06 ±1.30	77.60 ±1.35
GCN	PC Softmax	91.05 ±0.18	91.09 ±0.18	75.89 ±3.92	74.87 ±4.56
GCN	GraphSMOTE	92.58 ±0.11	92.58 ±0.11	91.00 ±0.21	91.05 ±0.21
GCN	**GraphENS**	**93.67** ±0.18	**93.67** ±0.17	**91.28** ±0.18	**91.28** ±0.17
GAT	Vanilla	81.41 ±0.43	76.76 ±0.48	60.19 ±4.25	54.60 ±5.15
GAT	Re-Weight	92.80 ±0.07	92.80 ±0.07	90.80 ±0.19	90.85 ±0.20
GAT	Oversampling	92.69 ±0.18	92.68 ±0.19	90.54 ±0.20	90.56 ±0.22
GAT	cRT	89.92 ±0.21	89.69 ±0.25	81.56 ±2.73	81.57 ±2.96
GAT	PC Softmax	90.48 ±0.11	90.49 ±0.12	79.41 ±3.91	78.80 ±4.28
GAT	GraphSMOTE	92.63 ±0.11	92.63 ±0.11	90.77 ±0.17	90.80 ±0.19
GAT	**GraphENS**	**93.10** ±0.19	**93.10** ±0.20	**91.11** ±0.28	**91.07** ±0.28

source nodes for saliency masking. To keep class-specific information of source nodes, exploitation of saliency maps of source nodes is required.

In Table 6, the adoption of unlabeled data in *GraphENS* improves the performance significantly in many cases. We speculate that our approach could exploit rich information even in unlabeled nodes to produce safe and diverse ego networks.

Table 6: Comparison of our method *GraphENS* with other baselines in semi-supervised learning settings. We report the averaged accuracy, balanced accuracy, and F1-score with the standard errors for 5 repetitions on three benchmark datasets for node classification tasks. † denotes the results when utilizing unlabeled nodes.

	Method	Cora-*Semi*			CiteSeer-*Semi*			PubMed-*Semi*		
		Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
GCN	Vanilla	63.98 ±1.32	58.28 ±1.59	55.97 ±2.33	40.76 ±2.68	40.54 ±2.22	35.11 ±3.33	57.48 ±2.67	62.35 ±1.41	55.27 ±3.58
GCN	Re-Weight	67.56 ±1.32	63.05 ±1.80	61.43 ±2.59	42.90 ±2.97	41.42 ±3.06	35.22 ±4.15	66.98 ±2.36	69.54 ±1.83	66.35 ±2.32
GCN	Oversampling	61.86 ±1.04	56.52 ±0.86	53.43 ±1.21	36.34 ±1.70	35.93 ±1.57	27.75 ±3.19	63.76 ±1.99	68.45 ±1.64	62.37 ±2.35
GCN	cRT	63.94 ±1.14	57.16 ±1.49	54.45 ±2.23	41.16 ±2.73	41.78 ±2.36	36.08 ±3.24	57.34 ±2.93	62.21 ±2.12	55.39 ±3.63
GCN	PC Softmax	69.20 ±1.02	64.68 ±1.76	63.80 ±2.05	45.24 ±2.85	44.34 ±2.33	40.38 ±2.86	64.78 ±0.75	66.89 ±1.37	64.82 ±0.74
GCN	DR-GCN	64.22 ±1.34	58.28 ±1.80	55.93 ±2.64	35.80 ±1.94	36.51 ±2.03	27.62 ±3.16	61.78 ±3.27	66.13 ±2.40	59.67 ±4.64
GCN	GraphSMOTE	64.70 ±1.76	58.71 ±2.20	56.31 ±3.29	39.74 ±2.62	38.27 ±2.04	31.45 ±3.39	64.00 ±1.41	67.90 ±1.12	63.44 ±1.51
GCN	**GraphENS**	69.78 ±1.01	65.01 ±1.67	64.69 ±1.60	52.42 ±1.09	49.55 ±1.18	46.27 ±1.54	68.54 ±1.15	**70.68** ±1.22	68.19 ±0.93
GCN	**GraphENS**†	**70.76** ±0.86	**66.78** ±1.31	**66.20** ±1.54	**53.58** ±1.26	**51.69** ±1.03	**49.42** ±1.71	**69.54** ±2.32	70.38 ±1.97	**69.00** ±2.07
GAT	Vanilla	67.80 ±1.30	64.08 ±1.44	63.02 ±1.66	43.42 ±1.69	42.39 ±2.02	38.30 ±2.13	66.16 ±1.75	70.37 ±1.16	65.67 ±1.69
GAT	Re-Weight	69.40 ±1.42	65.14 ±1.92	63.90 ±2.43	50.56 ±1.72	47.83 ±1.67	45.28 ±2.45	68.94 ±2.04	71.57 ±1.33	68.78 ±1.84
GAT	Oversampling	65.36 ±1.09	59.72 ±1.28	57.54 ±1.43	40.58 ±3.80	38.23 ±3.25	31.47 ±5.16	64.72 ±2.33	67.10 ±2.35	63.18 ±2.47
GAT	cRT	67.56 ±1.65	62.10 ±2.27	61.15 ±2.61	45.58 ±1.89	43.45 ±2.26	40.15 ±2.36	65.96 ±2.94	70.10 ±2.17	65.27 ±3.02
GAT	PC Softmax	69.30 ±1.17	**66.05** ±1.47	**65.06** ±1.61	50.08 ±2.13	48.61 ±2.15	47.19 ±2.09	68.42 ±1.16	**71.83** ±0.86	68.28 ±1.06
GAT	DR-GCN	66.76 ±1.77	61.47 ±2.43	59.95 ±3.07	41.28 ±3.14	40.71 ±2.37	33.96 ±3.66	66.96 ±1.80	69.02 ±1.52	66.11 ±2.10
GAT	GraphSMOTE	68.23 ±1.31	63.67 ±1.67	62.50 ±2.15	45.26 ±2.71	43.42 ±3.06	39.46 ±3.65	65.16 ±1.43	69.32 ±0.71	64.39 ±1.58
GAT	**GraphENS**	**69.62** ±0.91	65.27 ±1.55	64.16 ±1.87	53.90 ±1.25	51.74 ±0.82	49.47 ±1.57	**70.24** ±1.45	71.00 ±1.13	**69.93** ±1.21
GAT	**GraphENS**†	69.02 ±1.29	64.66 ±1.75	63.74 ±2.13	**56.32** ±2.00	**53.69** ±1.71	**51.76** ±2.24	70.18 ±2.14	70.52 ±1.70	69.42 ±1.84
GraphSAGE	Vanilla	62.30 ±0.99	56.24 ±1.65	52.84 ±2.47	40.16 ±0.92	39.59 ±1.10	34.12 ±0.53	62.30 ±3.61	66.57 ±3.15	61.02 ±4.14
GraphSAGE	Re-Weight	62.92 ±0.40	55.94 ±0.72	54.06 ±0.66	41.10 ±3.39	39.86 ±3.09	33.80 ±4.16	65.90 ±2.19	**69.27** ±1.40	65.59 ±2.21
GraphSAGE	Oversampling	61.40 ±0.83	52.99 ±1.23	49.57 ±1.63	39.04 ±2.00	38.89 ±1.99	32.17 ±2.88	59.26 ±2.26	63.59 ±2.55	58.06 ±2.78
GraphSAGE	cRT	63.20 ±1.27	57.21 ±1.69	54.71 ±2.57	40.34 ±0.56	40.59 ±0.84	36.18 ±0.76	62.90 ±2.25	67.13 ±1.78	61.52 ±2.80
GraphSAGE	PC Softmax	65.32 ±1.16	60.52 ±1.52	58.91 ±1.94	44.02 ±2.49	44.34 ±1.78	38.99 ±3.25	63.18 ±3.00	66.94 ±2.51	61.19 ±4.44
GraphSAGE	DR-GCN	61.82 ±0.62	55.14 ±0.79	51.93 ±1.01	42.86 ±2.03	41.72 ±1.38	37.49 ±2.64	62.50 ±1.98	65.65 ±2.25	61.73 ±1.91
GraphSAGE	GraphSMOTE	60.84 ±1.56	53.38 ±1.87	49.31 ±3.27	39.28 ±2.04	39.27 ±2.16	32.31 ±2.98	61.34 ±2.73	66.07 ±1.93	58.71 ±4.10
GraphSAGE	**GraphENS**	65.84 ±0.74	62.29 ±1.69	61.52 ±1.79	**51.12** ±1.93	48.91 ±1.94	46.78 ±2.02	**67.62** ±1.77	68.27 ±1.26	67.37 ±1.60
GraphSAGE	**GraphENS**†	**67.72** ±1.04	**64.18** ±1.31	**63.14** ±1.47	50.92 ±2.13	**49.94** ±1.60	**47.06** ±2.23	67.48 ±1.19	68.04 ±0.76	**67.64** ±0.99

A.4 Oversampling Ratio

We investigate the performance tendency according to oversampling scale, which determines the number of generated minor nodes. Specifically, the number of oversampled nodes is computed as oversampling ratio * (the number of nodes in the largest class - the number of nodes in the minor

class)). In Figure 5, we observe that the performance decreases as the scale is (1) under 0.6 or (2) over 1.0. We conjecture that the induced bias of major classes is not sufficiently corrected at low oversampling scales (1). In the regime of (2), a high oversampling scale impedes the learning of major class representations, resulting in performance degradation. Thus, the oversampling scale between 0.8 and 1.0 is a preferable option and we adopted 1.0 as the scale in our main experiments, following conventional oversampling methods.

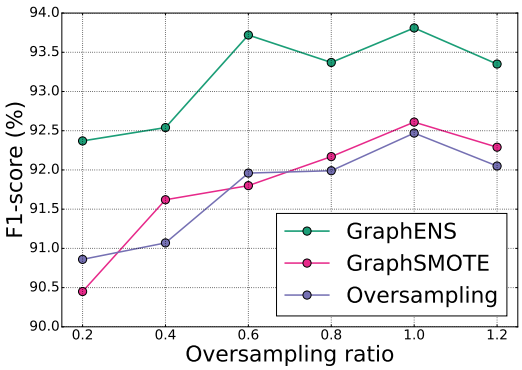


Figure 5: Performance analysis according to the oversampling ratio over AmazonPhoto dataset with GraphSAGE.

B ILLUSTRATIONS OF REPLACING EXPERIMENTS

To facilitate understanding of node/neighbor replacing experiments, we describe the experiments with intuitive illustrations. In Figure 6, to observe node memorization, we replace the node feature of v_{anchor} with v_{seen} in (a) and v_{anchor} with v_{unseen} in (b). The performance gap between v_{anchor} in (a) and (b) is computed. Similar to node-replacing experiments, to see neighbor memorization, we replace the node feature of v_{seen} with v_{anchor} in (c) and v_{seen} with v_{unseen} in (d). The accuracy difference between v_{seen} in (c) and v_{unseen} in (d) is calculated. The results of replacing experiments are in Figure 1.

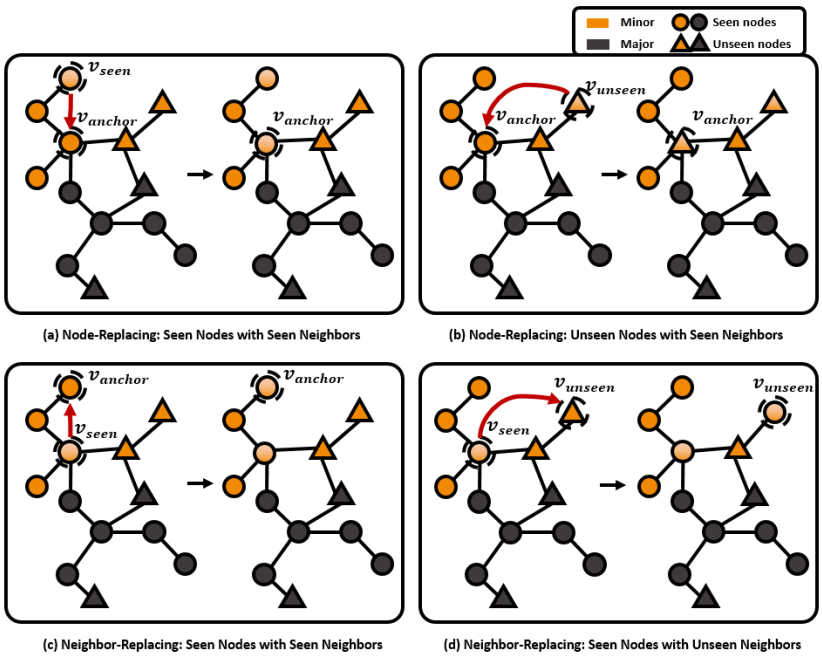


Figure 6: Overall pipeline of node-replacing and neighbor-replacing experiments.

C MAIN ALGORITHM

Algorithm 1 *GraphENS*

Input: Dataset $(G(V,E),\mathbf{y})$, model f_θ, class distribution $p_{class}(\mathcal{C})$, class-conditional node distribution $p_{node}(u|c)$, degree distribution $p_{degree}(\mathcal{D})$, number of classes C, number of samples to be oversampled N_c^{os} , feature masking hyperparameter k, learning rate η, temperature τ
Initialize: Model f_θ.
for $t = 1, 2, \ldots, T$ **do**
  $V_{new}, E_{new} \leftarrow V, E$
  for $c = 1, 2, \cdots, C$ **do**
    for $i = 1, 2, \cdots, N_c^{os}$ **do**
      $c_{target} \sim p_{class}(\mathcal{C})$ ▷ Sampling target class
      $v_{target} \sim p_{node}(u|c_{target})$
      $v_{minor} \sim p_{node}(u|c)$
      $\lambda \sim Beta(2,2)$
      if $t \leq 10$ **then** ▷ Warmup
        $v_{mixed} \leftarrow (1-\lambda)v_{minor} + \lambda v_{target}$
        $v_{n_1}, v_{n_2}, \cdots, v_{n_{|\mathcal{N}(v_{minor})|}} \leftarrow$ Neighbors of v_{minor}
        $\hat{E}_{new} \leftarrow \{\{v_{n_1}, v_{mixed}\}, \{v_{n_2}, v_{mixed}\}, \cdots, \{v_{n_{|\mathcal{N}(v_{minor})|}}, v_{mixed}\}\}$
      else
        $\phi \leftarrow KL(\hat{o}^{t-1}_{v_{minor}} \| \hat{o}^{t-1}_{v_{target}})$
        $\hat{\phi} \leftarrow \frac{1}{1+e^{-\phi}}$

        ▷ Saliency-based node mixing
        $K \leftarrow k\hat{\phi}$
        Compute M_K from `multinomial distribution` using $\mathcal{S}^{t-1}, K$
        $\Lambda_K \leftarrow \lambda \cdot M_K$
        $v_{mixed} \leftarrow (1-\Lambda_K) \odot v_{minor} + \Lambda_K \odot v_{target}$

        ▷ Neighbor sampling
        $p(u|v_{mixed}) \leftarrow \hat{\phi}p(u|v_{minor}) + (1-\hat{\phi})p(u|v_{target})$
        $r \sim p_{degree}(\mathcal{D})$
        $\{v_{n_1}, v_{n_2}, \cdots, v_{n_r}\} \sim p(u|v_{mixed})$
        $\hat{E}_{new} \leftarrow \{\{v_{n_1}, v_{mixed}\}, \{v_{n_2}, v_{mixed}\}, \cdots, \{v_{n_r}, v_{mixed}\}\}$
      end if
      $V_{new} \leftarrow V_{new} \cup \{v_{mixed}\}$
      $E_{new} \leftarrow E_{new} \cup \hat{E}_{new}$
    end for
  end for
  $\mathbf{o} \leftarrow f_\theta(V_{new}, E_{new})$
  $\mathcal{L}_{new} \leftarrow CrossEntropy(\mathbf{o}, \mathbf{y})$
  $\theta \leftarrow \theta - \eta(\nabla_\theta \mathcal{L}_{new})$
  for $v \in V$ **do**
    $\hat{o}^t_v \leftarrow \frac{1}{|\mathcal{N}(v)|+1} \sum_{u \in (\mathcal{N}(v) \cup v)} Softmax(\mathbf{o}_u/\tau)$ ▷ Confidence aggregation
  end for
  $\mathcal{S}^t \leftarrow$ Computing feature saliency with the gradient of $\mathcal{L}_{new}$ (Equation. (2))
end for
Output: f_θ

D DETAILED EXPERIMENT SETTINGS

In this section, we address all the details of experiments we conducted.

D.1 DATA STATISTICS

Table 7: Data summary - Label distribution of training datasets [%]

Dataset	**Domain**	$\mathbf{L}_0$	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$	$\mathbf{L}_5$	$\mathbf{L}_6$	$\mathbf{L}_7$	$\mathbf{L}_8$	$\mathbf{L}_9$
Cora-*Semi*	Citation Network	23.26	23.26	23.26	23.26	2.33	2.33	2.33	-	-	-
Cora-*LT*		54.04	25.04	11.57	5.39	2.38	1.11	0.48	-	-	-
CiteSeer-*Semi*	Citation Network	30.30	30.30	30.30	3.03	3.03	3.03	-	-	-	-
CiteSeer-*LT*		60.72	24.06	9.49	3.76	1.47	0.49	-	-	-	-
PubMed-*Semi*	Citation Network	83.33	8.33	8.33	-	-	-	-	-	-	-
PubMed-*LT*		90.10	9.01	0.89	-	-	-	-	-	-	-
AmazonPhoto	Co-Purchase Graph	31.58	26.64	11.70	11.06	9.92	7.59	1.12	0.39	-	-
AmazonComputers	Co-Purchase Graph	44.26	17.07	16.94	10.35	4.95	2.45	1.96	1.49	0.34	0.18

D.2 ARCHITECTURE

We evaluate our method with three representative GNN architectures - GCN (Kipf & Welling, 2017), GAT (Velickovic et al.), and GraphSAGE (Hamilton et al., 2017). In this section, we describe the detailed architecture of each GNN. For GCN, our model consists of l GCN layers with ReLU activation, followed by dropout Srivastava et al. (2014) with 0.5 dropping rate and a linear classifier. For GAT, our model comprises l GAT layers, which involves an ELU activation Clevert et al. (2016), followed by dropout with 0.5 dropping rate and a linear classifier. We also adopt multi-head attention with 8 heads and apply dropout with 0.6 dropping rate to its attention. GraphSAGE is composed of l SAGE layers with ReLU activation, followed by dropout with 0.5 dropping rate and a linear classifier. We search the architecture to maximize the validation accuracy for all algorithms. The search spaces of reported results are the number of layers $l \in \{1, 2, 3\}$ and hidden dimension $d \in \{64, 128, 256\}$.

D.3 EVALUATION PROTOCOL

We adopt Adam (Kingma & Ba, 2015) optimizer with the initial learning rate as 0.01 and train GNNs for 2000 epochs. The best models are selected with validation accuracy. We design a scheduler as a learning rate is halved if there is no improvement on validation loss for 100 iterations. We choose weight decay to all convolutional layers except for a linear classifier with 0.0005. We select the best model using validation accuracy over the training phase.

D.4 IMPLEMENTATION DETAILS

In this subsection, we explain the hyperparameters of our experiments. For all datasets, we use $Beta(2, 2)$ distribution to sample λ. Feature masking hyperparameter k and temperature τ are tuned among $\{1, 5, 10\}$ and $\{1, 2\}$, respectively. We employ warmup since our approach utilizes model confidence, so models need to be trained to some extent. The number of epochs for warmup is tuned among $\{1, 5\}$.

D.5 THE DETAILED SETUP FOR SEMI-SUPERVISED LEARNING

We validate imbalance handling approaches on the citation networks such as Cora, CiteSeer, and PubMed. We follow the split in Yang et al. (2016) and set the imbalance ratio as 10. Specifically, for Cora and CiteSeer, three classes have only two labeled nodes while other classes have 20 labeled nodes since the number of labeled nodes in each class is 20 in Yang et al. (2016). For PubMed, only two classes have two labeled nodes as PubMed consists of three classes.