Imbalanced Node Classification Beyond Homophilic Assumption

Jie Liu[1] , **Mengting He**[1] , **Guangtao Wang**[2] , **Quoc Viet Hung Nguyen**[3] , **Xuequn Shang**[1†] and **Hongzhi Yin**[4†]
[1]Northwestern Polytechnical University
[2]Bytedance Inc
[3]Griffith University
[4]The University of Queensland
{jayliu,hmt468}@mail.nwpu.edu.cn, xjtuwgt@gmail.com, henry.nguyen@griffith.edu.au,
shang@nwpu.edu.cn, h.yin1@uq.edu.au

Abstract

Imbalanced node classification widely exists in real-world networks where graph neural networks (GNNs) are usually highly inclined to majority classes and suffer from severe performance degradation on classifying minority class nodes. Various imbalanced node classification methods have been proposed recently which construct synthetic nodes and edges w.r.t. minority classes to balance the label and topology distribution. However, they are all based on the homophilic assumption that nodes of the same label tend to connect despite the wide existence of heterophilic edges in real-world graphs. Thus, they uniformly aggregate features from both homophilic and heterophilic neighbors and rely on feature similarity to generate synthetic edges, which cannot be applied to imbalanced graphs in high heterophily. To address this problem, we propose a novel GraphSANN for imbalanced node classification on both homophilic and heterophilic graphs. Firstly, we propose a *unified feature mixer* to generate synthetic nodes with both homophilic and heterophilic interpolation in a unified way. Next, by randomly sampling edges between synthetic nodes and existing nodes as candidate edges, we design an *adaptive subgraph extractor* to adaptively extract the contextual subgraphs of candidate edges with flexible ranges. Finally, we develop a *multi-filter subgraph encoder* which constructs different filter channels to discriminatively aggregate neighbors' information along the homophilic and heterophilic edges. Extensive experiments on eight datasets demonstrate the superiority of our model for imbalanced node classification on both homophilic and heterophilic graphs.

1 Introduction

Graph Neural Networks (GNNs) successfully extend deep learning approaches to graph data and have exhibited pow-

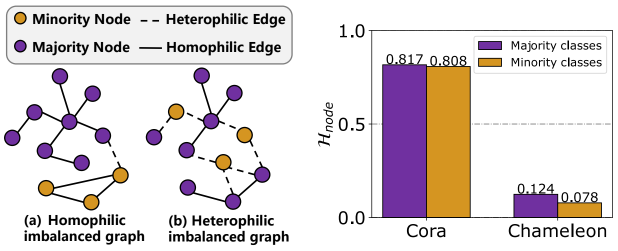

Figure 1: **Left**: Illustration of (a) homophilic and (b) heterophilic imbalanced graphs. Many imbalanced networks exhibit strong heterophily. For example, in transaction networks, fraudsters often disguise themselves by connecting to normal customers. **Right**: Comparison of node homophily $\mathcal{H}_{node}$ of a homophilic imbalanced network (Cora) and a heterophilic imbalanced network (Chameleon).

erful learning ability on node classification task [Kipf and Welling, 2017; Veličković *et al.*, 2018; Liu *et al.*, 2023]. Despite their effectiveness, most existing GNNs neglect the widely existing class-imbalance problem in real-world networks, where certain class(es) have significantly fewer node samples for training than other classes [Sun *et al.*, 2021b]. For example, in online transaction networks, the majority of nodes are normal customers while only a small number are fraudsters; in molecular networks, there are much more low-mass atoms than high-mass atoms. Due to the dominating role of majority class nodes in the training set, classical GNNs are often highly inclined to majority classes, leading to severe performance degradation for minority node classification.

To address the class-imbalance problem in the node classification task, many methods have been proposed recently. They mainly relieve the imbalance problem by generating synthetic nodes for minority classes and further constructing synthetic edges between the generated nodes and the original nodes. For example, GraphSMOTE [Zhao *et al.*, 2021] generates synthetic nodes by interpolating nodes of the same minority class through SMOTE [Chawla *et al.*, 2002] and generates their linkages through a pre-trained edge generator. ImGAGN [Qu *et al.*, 2021] also synthesizes minority class nodes and connects them to real minority class nodes through a generative adversarial network. GraphENS [Park

[†]Corresponding authors.

et al., 2022] further synthesizes the whole ego networks for synthetic minority nodes based on information from both minority and majority classes.

Although having acquired prominent performances on certain imbalanced datasets, these existing methods are based on the homophily assumption that edges tend to connect nodes of the same class label (Figure 1(a)). However, many investigations [Sun *et al.*, 2021a; Yu *et al.*, 2020] show that heterophilic connections which link nodes of different classes also widely exist in imbalanced graphs (Figure 1(b)). Existing imbalanced node classification methods suffer from three severe problems when applied to networks with a large portion of heterophilic connections. **P1**: Most existing methods generate synthetic nodes based on homophilic interpolation, which restricts interpolated node pairs to be the same minority class. This causes synthetic nodes to lack diversity when real minority class nodes are very limited. **P2**: Existing models mainly resort to node feature similarity for synthetic edge construction. This strategy works well for homophilic edges which connect nodes with similar features but fail in constructing heterophilic edges and would thus introduce structure bias (i.e., heterophilic/homophilic edge distribution drift). **P3**: Existing methods conduct uniform message passing for both homophilic and heterophilic edges when aggregating features, and consequently result in much noisy information from dissimilar neighbors derived from heterophilic edges to be aggregated into the target nodes. This would seriously degrade the quality of node embeddings and hurt the following node classification task.

In light of this, we propose a novel **S**ubgraph-aware **A**daptive **Graph N**eural **N**etwork (**GraphSANN**) for imbalanced node classification on both homophilic and heterophilic graphs. GraphSANN consists of three major components, i.e., unified feature mixer, adaptive subgraph extractor, and multi-filter subgraph encoder. Specifically, to tackle **P1**, GraphSANN first applies a unified feature mixer to carry out both homophilic and heterophilic interpolation in a unified way. Next, to tackle **P2**, instead of generating edges based on feature similarity, we propose an adaptive subgraph extractor to extract the surrounding subgraphs of candidate synthetic edges with flexible ranges. In this way, distant but similar nodes can be absorbed into the subgraph whose general structural information will be encoded to predict the existence of the edge. To tackle **P3** and encode the subgraphs consisting of both homophilic and heterophilic connections, we design a multi-filter subgraph encoder to aggregate messages only from similar nodes instead of dissimilar ones by fusing the output messages of three distinct filters. Finally, after generating synthetic nodes/edges and attaching them to the original graph, we apply a multi-filter GNN as node classifier to encode the acquired balanced graph for node classification.

The major contributions of this work are stated as follows:

- To the best of our knowledge, this paper is the first work to tackle the imbalanced node classification problem beyond the homophilic assumption.
- We design a novel imbalanced node classification model GraphSANN which is able to build balanced graph by generating synthetic nodes and constructing both homophilic and heterophilic synthetic edges between generated and original nodes, and aggregates the information from homophilic and heterophilic neighbors discriminatively.
- Extensive experiments on eight benchmark datasets show that GraphSANN acquires superior performance on both imbalanced homophilic and heterophilic graphs.

2 Related Work

2.1 Heterophilic Graph Neural Networks

Since most existing GNNs follow the homophily assumption and thus face significant performance degradation on heterophilic graphs, heterophily-based GNNs have been proposed, which can be roughly categorized into two groups [Zheng *et al.*, 2022]: (1) Neighbor extension methods which aim to expand local neighborhood to absorb features from distant but informative nodes. For example, MixHop [Abu-El-Haija *et al.*, 2019] aggregates messages from multi-hop neighbors respectively and mixes them together through concatenation. UGCN [Jin *et al.*, 2021] further restricts nodes from two-hop neighbors to have at least two different paths to the ego node. (2) Adaptive message aggregation methods which design adaptive aggregation operations to learn discriminative information from homophilic and heterophilic linkages. For example, FAGCN [Bo *et al.*, 2021] adopts a self-gating attention mechanism to uniformly learn low-frequency and high-frequency signals from neighbors. ACM [Luan *et al.*, 2022] further designs a linear combination of low-pass and high-pass filters to adaptively learn information from different filter channels.

2.2 Imbalanced Node Classification

Generally, imbalanced node classification methods can be divided into two groups, generic and network-specific methods. Generic ones directly combine general class-imbalance approaches (e.g. oversampling, re-weight, SMOTE [Chawla *et al.*, 2002], etc.) with GNNs to graph data. For example, *Oversampling* [Buda *et al.*, 2018] replicates existing node embeddings learned from GNNs to produce more minority node representations; *Re-weight* [Yuan and Ma, 2012] assigns larger penalty weights to minority nodes when computing training loss. Network-specific methods usually take account of the sophisticated topology of graphs to generate synthetic nodes and further determine the connections between the generated nodes and original nodes [Chen *et al.*, 2021; Xia *et al.*, 2021]. Among them, DR-GCN [Shi *et al.*, 2020] utilizes class-conditional adversarial training to enhance the separation of different classes. GraphSMOTE [Zhao *et al.*, 2021] interpolates nodes from minority classes as synthetic nodes and generates linkages through a pre-trained edge generator. ImGAGN [Qu *et al.*, 2021] synthesizes minority nodes and connects them to original nodes through a generative adversarial network. GraphENS [Park *et al.*, 2022] demonstrates the overfitting problem of neighbor memorization and proposes to generate the whole ego-networks for synthetic nodes based on information from all classes.

However, these models are all based on the homophily assumption and thus suffer from performance degradation when applied to networks with strong heterophily.

3 Notations and Problem Definition

Definition 1. Graph Homophily and Heterophily. Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ represents the node set, and for each edge $e = (v, \mu), v, \mu \in \mathcal{V}$, if v and μ have the same class label, the edge e is homophilic. Otherwise, e is heterophilic. Most graphs have both homophilic and heterophilic edges at the same time. We define the node homophily $\mathcal{H}_{node}$ and edge homophily $\mathcal{H}_{edge}$ to quantitatively measure the homophily degree of a graph as follows.

$$\mathcal{H}_{node} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\mu \in \mathcal{N}(v) : y_v = y_\mu|}{|\mathcal{N}(v)|}, \tag{1a}$$

$$\mathcal{H}_{edge} = \frac{|\{(v, \mu) \in \mathcal{E} : y_v = y_\mu\}|}{|\mathcal{E}|}. \tag{1b}$$

Based on the definitions of $\mathcal{H}_{node}, \mathcal{H}_{edge} \in [0, 1]$, the node heterophily and edge heterophily can be defined as $1 - \mathcal{H}_{node}$ and $1 - \mathcal{H}_{edge}$, respectively. Graphs with strong homophily have higher $\mathcal{H}_{node}$ and $\mathcal{H}_{edge}$, and vice versa.

Problem Definition. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ denote an attribute graph, where $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is node feature matrix whose i-th row represents a d-dimensional feature vector of the i-th node. For node classification, each node is also associated with a one-hot node label $Y_{i,:} \in \mathbb{R}^C$ where C is the number of node classes. If the class size distribution is imbalanced, we name it as an imbalanced node classification problem. The imbalance ratio is defined as *im_ratio* = $\frac{min_c(|\mathcal{V}_c|)}{max_c(|\mathcal{V}_c|)} \ll 1$, where $|\mathcal{V}_c|$ denotes the node set with class label $c \in \{1, 2, \cdots, C\}$. Then, we give the formal definition of imbalanced node classification as follows.

Given an imbalanced graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ composed of both homophilic and heterophilic edges, our goal is to learn a node classifier $f : f(\mathcal{V}, \mathcal{E}, \mathbf{X}) \to \mathbf{Y}$ that can well classify both majority and minority classes and can be well generalized on graphs with either low or high heterophily.

4 Methodology

In this section, we introduce our novel GraphSANN for imbalanced node classification. As illustrated in Figure 2, GraphSANN consists of three core components, (1) *Unified Feature Mixer* which carries out both homophilic and heterophilic interpolation to generate synthetic nodes (Subsection 4.1); (2) *Adaptive Subgraph Extractor* which adaptively extracts subgraphs around candidate synthetic edges (Subsection 4.2) and (3) *Multi-filter based subgraph encoder* which encodes the subgraphs extracted from component (2) into edge score using multiple passes of filters (Subsection 4.3). Please refer to Algorithm 1 in the Appendix for the forward propagation procedure of GraphSANN . We elaborate on each component in the following subsections.

4.1 Unified Feature Mixer

Most existing class-imbalance models apply homophilic interpolation on existing node pairs $< v_s, v_t >$ to generate synthetic nodes, which restricts v_s and v_t to be the nearest neighbors of the same minority class [Zhao *et al.*, 2021; Qu *et al.*, 2021]. The generated nodes under this strategy suffer from feature diversity problem especially when original minority nodes are very limited [Park *et al.*, 2022]. To address this problem, we propose a unified feature mixer that conducts both homophilic and heterophilic interpolation in a unified way, which consists of two steps: (1) unified node pair sampling and (2) integrated gradient based feature mixup.

Unified Node Pair Sampling

Before conducting feature mixup to generate synthetic minority nodes, we first sample node pairs $< v_s, v_t >$ from the existing node set for interpolation. Here v_s is sampled from minority classes while v_t is from the entire classes, and thus it could have either the same or different class compared to v_s, formed as $S_{pair} = \{< v_s, v_t >| \ v_s \in \mathcal{V}_{minor}, \ v_t \in \mathcal{V}\}$. Let $\mathcal{C} = \{1, 2, \ldots, C\}$ and $\mathcal{C}_M \subset \mathcal{C}$ be the entire class set and minority class set, the sampling distributions for v_s and v_t are denoted as $v_s \sim p_s(u \mid \mathcal{C}_M)$ and $v_t \sim p_t(u \mid \mathcal{C})$, respectively. To obtain a uniformly sampling v_t from all the classes, we define p_s and p_t as follows:

$$p_s(u \mid \mathcal{C}_M) = \frac{1}{|\mathcal{V}_m|}, \ m \in \mathcal{C}_M, \tag{2a}$$

$$p_t(u \mid \mathcal{C}) = \frac{\log(|\mathcal{V}_c| + 1)}{(|\mathcal{V}_c| + 1) \sum_{c \in \mathcal{C}} \log(|\mathcal{V}_c| + 1)}. \tag{2b}$$

It is straightforward to get that p_t reaches the peak when $|\mathcal{V}_c| = 2$ and gets 0 when $|\mathcal{V}_c| = 0$ or $|\mathcal{V}_c| = \infty$. In this way, even if the sizes of majority classes are significantly larger than minority classes, nodes from the entire classes have roughly equal chances to be sampled as v_t. Here we use over-sampling scale ζ to control the amount of sampled node pairs.

Integrated Gradient based Feature Mixup

Afterward, we interpolate the raw features of each node pair $< v_s, v_t >$ to generate synthetic nodes. Since v_t is sampled beyond the same minority class as v_s, we only preserve generic node attributes which are irrelevant to class prediction to avoid introducing distracting information. As introduced by [Sundararajan *et al.*, 2017], integrating gradient can effectively evaluate the contributions of input features to the model prediction. Compared to directly using gradients to evaluate feature importance [Park *et al.*, 2022], integrated gradient addresses the saturation and thresholding problems [Shrikumar *et al.*, 2017] and can acquire more reliable feature importance. Specifically, the integrated gradient $\mathrm{IG}_i(\mathbf{x})$ along the i-th dimension of input node feature $\mathbf{x} \in \mathbb{R}^d$ is calculated as follows:

$$\mathrm{IG}_i(\mathbf{x}) = \mathbf{x}_i \int_{t=0}^{1} \frac{\partial \mathcal{L}_{cls}(t\mathbf{x}, \mathbf{y})}{\partial \mathbf{x}_i} dt, \tag{3}$$

where $\mathbf{y}$ represents the vector of true class labels and $\mathcal{L}_{cls}$ denotes the node classification loss. Then, we compute the distance ψ_{st} between v_s and v_t by:

$$\psi_{st} = \|\mathbf{W}_p \mathbf{x}_s - \mathbf{W}_p \mathbf{x}_t\|_2, \tag{4}$$

where $\mathbf{W}_p \in \mathbb{R}^{d \times d'}$ represents the projection matrix, and $\mathbf{x}_s$ and $\mathbf{x}_t$ are raw attributes of v_s and v_t, respectively. We define $\hat{\psi}_{st} = \frac{1}{1+\psi_{st}} \in [0, 1]$ as the similarity between v_s and

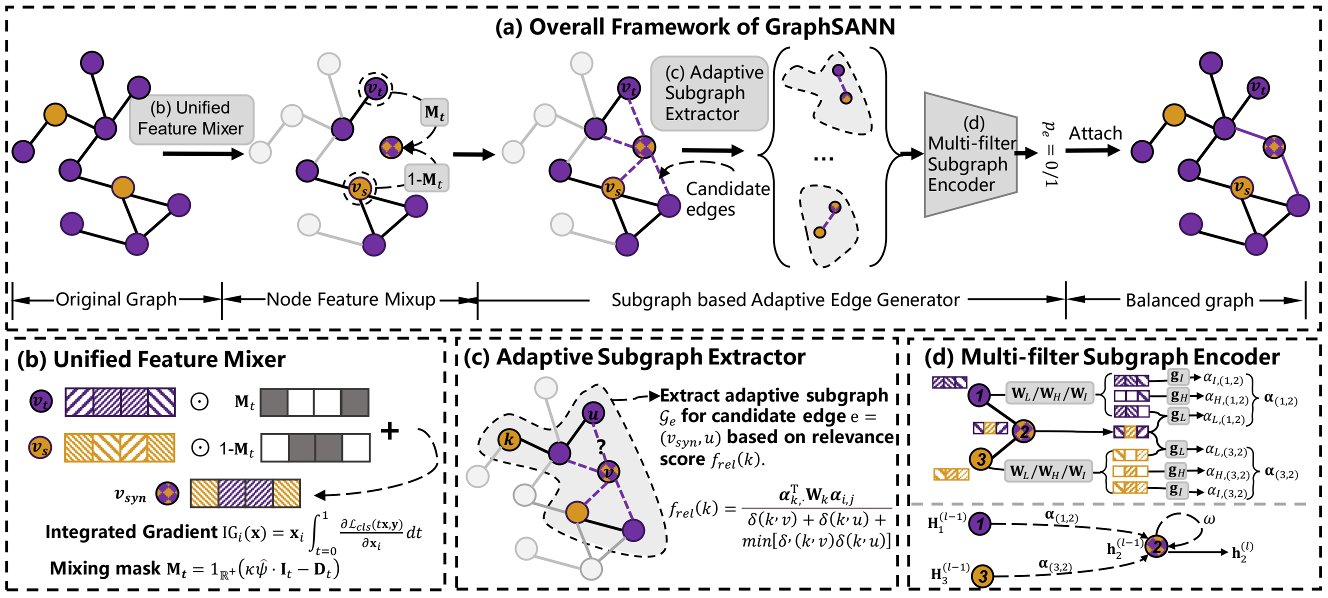
(a) Overall Framework of GraphSANN
(b) Unified Feature Mixer
v_t
$\mathbf{M}_t$
1-$\mathbf{M}_t$
v_s
(c) Adaptive Subgraph Extractor
Candidate edges
...
(d) Multi-filter Subgraph Encoder
$p_e = 0/1$
Attach
Original Graph
Node Feature Mixup
Subgraph based Adaptive Edge Generator
Balanced graph
(b) Unified Feature Mixer
v_{syn}
Integrated Gradient $\mathrm{IG}_i(\mathbf{x}) = \mathbf{x}_i \int_{t=0}^{1} \frac{\partial \mathcal{L}_{cls}(t\mathbf{x},\mathbf{y})}{\partial \mathbf{x}_i} dt$
Mixing mask $\mathbf{M}_t = 1_{\mathbb{R}^+}(\kappa\hat{\psi} \cdot \mathbf{I}_t - \mathbf{D}_t)$
(c) Adaptive Subgraph Extractor
k u v ?
Extract adaptive subgraph $\mathcal{G}_e$ for candidate edge e = (v_{syn}, u) based on relevance score $f_{rel}(k)$.
$f_{rel}(k) = \frac{\boldsymbol{\alpha}_{k,\cdot}^{\mathrm{T}} \mathbf{W}_k \boldsymbol{\alpha}_{i,j}}{\delta(k, v) + \delta(k, u) + min[\delta(k, v)\delta(k, u)]}$
(d) Multi-filter Subgraph Encoder
$\mathbf{W}_L/\mathbf{W}_H/\mathbf{W}_I$
$\mathbf{g}_I$ $\alpha_{I,(1,2)}$
$\mathbf{g}_H$ $\alpha_{H,(1,2)}$
$\mathbf{g}_L$ $\alpha_{L,(1,2)}$
$\boldsymbol{\alpha}_{(1,2)}$
$\mathbf{g}_L$ $\alpha_{L,(3,2)}$
$\mathbf{g}_H$ $\alpha_{H,(3,2)}$
$\mathbf{g}_I$ $\alpha_{I,(3,2)}$
$\boldsymbol{\alpha}_{(3,2)}$
$\mathbf{H}_1^{(l-1)}$
$\mathbf{H}_3^{(l-1)}$
$\mathbf{h}_2^{(l-1)}$
$\mathbf{h}_2^{(l)}$
ω

Figure 2: (a) Overall framework of GraphSANN . It is composed of three core components, i.e., (b) unified feature mixer (c) adaptive subgraph extractor, and (d) multi-filter subgraph encoder.

v_t. Finally, we construct the masking vector $\mathbf{M}_t \in \mathbb{R}^d$ as follows:

$$\mathbf{M}_t = 1_{\mathbb{R}^+}(\kappa\hat{\psi_{st}} \cdot \mathbf{I}_t - \mathbf{D}_t)), \tag{5}$$

where $\mathbf{I}_t \in \mathbb{R}^d$ is an all-ones vector, κ is a hyper-parameter, $\mathbf{D}_t = [\mathrm{IG}_1(\mathbf{x}_t), \dots, \mathrm{IG}_d(\mathbf{x}_t)]$ is the feature importance vector for x_t, and $1_{\mathbb{R}^+}(\cdot)$ is an indicator function which returns 1 when the input is positive otherwise 0. Thus, the mixed feature $\mathbf{x}_{syn}$ of synthetic nodes is formulated as follows:

$$\mathbf{x}_{syn} = (1 - \mathbf{M}_t) \odot \mathbf{x}_s + \mathbf{M}_t \odot \mathbf{x}_t. \tag{6}$$

4.2 Adaptive Subgraph Extractor

After generating synthetic nodes $v_{syn} \in \mathcal{V}_{syn}$ for the minority classes, we need to determine their connections to the original graph. The existence of an edge $e = (v_{syn}, u)$ is highly related to the structural information embedded in its surrounding subgraph $\mathcal{G}_e = \{\mathcal{V}_e, \mathcal{E}_e\}$ regardless of its homophily or heterophily [Zhang and Chen, 2018]. Here, we first randomly sample connections between synthetic and original nodes as candidate edges, then we adaptively extract the subgraphs of candidate edges and finally encode their structure information to predict edge existence. Specifically, given the node pair set S_{pair} and synthetic node set $\mathcal{V}_{syn}$, the candidate synthetic edge set $\mathcal{E}_{syn}$ is constructed as follows:

$$\mathcal{E}_{syn} = \{(v, u) \mid v \in \mathcal{V}_{syn}, u \in \mathcal{V}_{nei}\}, \tag{7}$$

$$\mathcal{V}_{nei} = [\mathcal{N}_1(v_s) \cup \mathcal{N}_1(v_t)]_\xi, < v_s, v_t > \in S_{pair}, \tag{8}$$

where $\mathcal{N}_1(v)$ returns 1-hop neighbors of v and v itself, $[\cdot]_\xi$ is random sampling with sampling ratio ξ. After obtaining $\mathcal{E}_{syn}$, we extract the enclosing subgraph $\mathcal{G}_e$ for each $e \in \mathcal{E}_{syn}$. Considering that nodes with high structural and semantic similarities might be distant from each other in heterophilic graphs, instead of fixing the subgraph to h-hop neighbors, we adaptively adjust the range of a subgraph based on a relevance score function $f_{rel}(\cdot)$. For any h-hop neighbor $k \in \mathcal{N}_h(v) \cup \mathcal{N}_h(u)$, $f_{rel}(k)$ is calculated as follows:

$$f_{rel}(k) = \frac{\boldsymbol{\alpha}_{k,\cdot}^{(l),\mathrm{T}} \mathbf{W}_k \boldsymbol{\alpha}_{i,j}^{(l)}}{\delta(k, v) + \delta(k, u) + min[\delta(k, v), \delta(k, u)]}, \tag{9}$$

$$\boldsymbol{\alpha}_{k,\cdot}^{(l)} = \frac{1}{|\mathcal{N}_1(k)|} \sum_{i \in \mathcal{N}_1(k)} \boldsymbol{\alpha}_{k,i}^{(l)}. \tag{10}$$

Where $\delta(k, v)$ is the length of shortest path from k to v, $\boldsymbol{\alpha}_{k,i} \in \mathbb{R}^3$ represents the multi-pass weight vector between node k and i (further illustrated in Subsection 4.3). $\boldsymbol{\alpha}_{k,\cdot}^{(l)}$ is the mean value of all the coefficient vectors between k and its 1-hop neighbors, which reflects the homophily status of node k. $\mathbf{W}_k \in \mathbb{R}^{3\times 3}$ is a weight matrix. Then we select top M nodes from $f_{rel}(k)$ along with central nodes u and v to construct the enclosing subgraph. The M is calculated based on subgraph density: $M = \lceil |\mathcal{V}_e|(1 + \frac{2|\mathcal{E}_e|}{|\mathcal{V}_e|(|\mathcal{V}_e - 1|)}) \rceil$.

4.3 Multi-filter Subgraph Encoder

In this section, we propose a novel subgraph encoder to embed the subgraph surrounding a candidate edge $e \in \mathcal{E}_{syn}$ into a vector, and then predict whether the edge e should be generated or not based on that. Let $\mathcal{G}_e$ be the extracted enclosing subgraph of the candidate edge e. Considering the widely existing heterophilic connections in $\mathcal{G}_e$, we design a multi-filter subgraph encoder that can discriminatively aggregate information from homophilic and heterophilic neighbors. Specifically, let $\mathbf{h}_u^{(l-1)} \in \mathbb{R}^{d_{l-1}\times 1}$ denote the $(l-1)$-th layer feature of node $u \in \mathcal{G}_e$, $\mathbf{h}_u^{(0)} = [\mathbf{x}_u \parallel \mathbf{z}_u]$ where $\mathbf{z}_u$ is one-hot labeling feature acquired by DRNL[Zhang and Chen, 2018]. We now compute the weight coefficients $\alpha_{L,(u,k)}^{(l)}, \alpha_{H,(u,k)}^{(l)}, \alpha_{I,(u,k)}^{(l)}$ which reflect the importance of different frequencies of signals as follows:

$$\alpha_{L,(u,k)}^{(l)} = \sigma\Big(\mathbf{g}_L^{\mathrm{T}}\big[\mathbf{W}_L^{(l)}\mathbf{h}_u^{(l-1)} \parallel \mathbf{W}_L^{(l)}\mathbf{h}_k^{(l-1)}\big]\Big), \tag{11}$$

$$\alpha_{H,(u,k)}^{(l)} = \sigma\Big(\mathbf{g}_H^{\mathrm{T}}\big[-\mathbf{W}_H^{(l)}\mathbf{h}_k^{(l-1)}\big]\Big), \tag{12}$$

$$\alpha_{I,(u,k)}^{(l)} = \sigma\Big(\mathbf{g}_I^{\mathrm{T}}\big[\mathbf{W}_I^{(l)}\mathbf{h}_u^{(l-1)}\big]\Big). \tag{13}$$

Where $\mathbf{W}_L^{(l)}$,$\mathbf{W}_H^{(l)}$, $\mathbf{W}_I^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ are the weight matrices that project $\mathbf{h}_u^{(l-1)}$ into low-frequency, high-frequency and identity messages, respectively. $\mathbf{g}_L^{\mathrm{T}} \in \mathbb{R}^{2d_l\times 1}$, $\mathbf{g}_H^{\mathrm{T}}$, $\mathbf{g}_I^{\mathrm{T}} \in \mathbb{R}^{d_l\times 1}$ are the convolutional vectors, $\sigma(\cdot)$ is the Sigmoid function. Then we compute weight vector $\boldsymbol{\alpha}_{(u,k)}^{(l)}$ by normalizing the importance weights of different frequencies:

$$\boldsymbol{\alpha}_{(u,k)}^{(l)} = \big[\widetilde{\alpha}_{L,(u,k)}^{(l)}, \widetilde{\alpha}_{H,(u,k)}^{(l)}, \widetilde{\alpha}_{I,(u,k)}^{(l)}\big], \tag{14}$$

$$\widetilde{\alpha}_{i,(u,k)}^{(l)} = \frac{\exp\big(\alpha_{i,(u,k)}^{(l)}\big)}{\sum_{i\in\{L,H,I\}}\exp\big(\alpha_{i,(u,k)}^{(l)}\big)}. \tag{15}$$

Next, we aggregate multi-frequency messages of neighbor nodes k with weight vector $\boldsymbol{\alpha}_{(u,k)}^{(l)}$ to compute the central node embedding $\mathbf{h}_u^{(l)}$:

$$\mathbf{h}_u^{(l)} = \omega\mathbf{h}_u^{(l-1)} + \sum_{k\in\mathcal{N}_1(u)} \boldsymbol{\alpha}_{(u,k)}^{(l)}\mathbf{H}_k^{(l-1)}, \tag{16}$$

$$\mathbf{H}_k^{(l-1)} = \mathrm{ReLU}\Big(\big[\mathbf{W}_L^{(l)}\mathbf{h}_k^{(l-1)}, \mathbf{W}_H^{(l)}\mathbf{h}_k^{(l-1)}, \mathbf{W}_I^{(l)}\mathbf{h}_k^{(l-1)}\big]^{\mathrm{T}}\Big). \tag{17}$$

Where ω is a hyper-parameter, $\mathbf{h}_u^{(L)}$ denotes the aggregated node embedding of node $u \in \mathcal{G}_e$ after stacking L-layer encoders. To acquire structural information from different orders of neighbors, we concatenate node embeddings from different layers and utilize a mean readout to compute the existence probability p_e of subgraph $\mathcal{G}_e$:

$$\mathbf{h}_\mu = \mathbf{h}_\mu^{(1)} \parallel \mathbf{h}_\mu^{(2)} \parallel \ldots \parallel \mathbf{h}_\mu^{(L)}, \tag{18}$$

$$p_e = \frac{1}{|\mathcal{V}_e|}\sum_{u\in\mathcal{V}_e}\mathbf{W}_{pool}\mathbf{h}_u. \tag{19}$$

Where $\mathbf{W}_{pool} \in \mathbb{R}^{1\times(d_1+\ldots+d_L)}$ projects the latent embeddings into a scalar p_e which reflects the existence probability of synthetic edge e. We remove the edges with low p_e from $\mathcal{E}_{syn}$ based on threshold η and attach the rest of synthetic edges to the original graph to construct the adjacency matrix $\tilde{\mathbf{A}}$ after over-sampling:

$$\tilde{\mathbf{A}}(v_{syn}, u) = \begin{cases} 1, & \text{if } p_e > \eta \\ 0, & \text{otherwise.} \end{cases}$$

4.4 Optimization Objective

In this section, we introduce the optimization objective of our proposed GraphSANN for imbalanced node classification, which consists of two optimization tasks: 1) adjacency matrix reconstruction and 2) node classification.

Datasets	Nodes	Edges	Features	Classes	$\mathcal{H}_{\text{edge}}$	$\mathcal{H}_{\text{node}}$
Cora	2,708	5,429	1,433	7	0.8100	0.8252
Pubmed	19,717	44,338	500	3	0.8024	0.7924
Citeseer	2,277	4,732	3,703	6	0.7362	0.7175
Chameleon	5,201	36,101	2,325	5	0.2795	0.2470
Squirrel	5,201	217,073	2,089	5	0.2416	0.2156
Film	7,600	33,544	931	5	0.2200	0.2400
Amazon-CP	13,381	245,778	767	10	0.7721	0.7853
Amazon-PH	7,487	119,043	745	8	0.8272	0.8365

Table 1: Statistics of Datasets.

Adjacency Matrix Reconstruction. We train our multi-filter subgraph encoder with an adjacency matrix reconstruction task. Let $\mathbf{A}$ denote the adjacency matrix of the original graph, and $\mathbf{A}(u, v) = 1$ indicate the existence of an edge between u and v. Considering the sparsity of positive edges, we also adopt negative sampling [Zhou *et al.*, 2022]. Specifically, for each positive edge $\mathbf{A}(u, v) = 1$, we randomly sample an unlinked edge which makes $\mathbf{A}(u, m) = 0$ as a negative sample and constructs a negative set $\mathcal{M}^-$. The loss function for adjacency matrix reconstruction is formed as follows:

$$\mathcal{L}_{rec} = \sum_{\substack{\mathbf{A}(u,v)>0,\\(v,m)\in\mathcal{M}^-}} \Big[\big\|\hat{\mathbf{A}}(u,v) - \mathbf{A}(u,v)\big\|_F^2 + \big\|\hat{\mathbf{A}}(u,m) - \mathbf{A}(u,m)\big\|_F^2\Big], \tag{20}$$

where $\hat{\mathbf{A}}$ is the predicted adjacency matrix of original graph. **Node classification**. After attaching the synthetic nodes and edges to the original graph, we transform it into a balanced network. Since the balanced graph can also be heterophilic, we adopt the multi-filter graph encoder introduced in 4.3 as node classifier by replacing the readout procedure with a one-layer MLP followed by Softmax:

$$\hat{\mathbf{y}}_u = \mathrm{Softmax}(\mathrm{MLP}(\mathbf{h}_u^{(L)})). \tag{21}$$

The output dimension of MLP$(\cdot)$ is equal to class number C. The loss function of node classification is defined as follows:

$$\mathcal{L}_{cls} = -\frac{1}{|\mathcal{V}|}\sum_{v=1}^{|\mathcal{V}|}\sum_{c=1}^{C}\log(\hat{\mathbf{y}}_v[c]\cdot\mathbf{y}_v[c]). \tag{22}$$

The overall objective function is then formed as follows with $\lambda \in (0, 1]$:

$$\mathbf{min}\mathcal{L} = (1-\lambda)\mathcal{L}_{rec} + \lambda\mathcal{L}_{cls}. \tag{23}$$

5 Experiment

In this section, we conduct extensive experiments on eight public datasets to evaluate the effectiveness of GraphSANN, which aim to answer five research questions: **RQ1**: How does GraphSANN perform compared to other baselines in imbalanced node classification on both homophilic and heterophilic graphs? **RQ2**: How effective is GraphSANN under different imbalance ratios? **RQ3**: How does each core component of GraphSANN contribute to the performance gain? **RQ4**: How do different hyper-parameter values affect the performance of GraphSANN ? **RQ5**: Can GraphSANN learn effective node representation to separate different classes of nodes in the embedding space?

Method $\mathcal{H}_{edge}$	**Cora** 0.8100			**Pubmed** 0.8024			**Citeseer** 0.7362			**Amazon-Computers** 0.7721		
Metrics(%)	ACC	F1	AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1	AUC
GCN	53.68±1.61	45.63±0.35	81.30±0.62	53.69±0.49	51.66±2.53	71.86±0.45	44.59±0.60	28.38±1.37	74.35±0.63	55.32±0.35	44.16±0.36	89.80±1.79
ACM	55.28±0.75	47.95±0.48	85.23±0.48	49.72±0.73	50.43±1.28	68.25±1.02	48.32±0.58	30.47±1.03	78.56±0.47	56.32±0.43	46.13±0.47	91.26±1.78
OverSamp	62.79±0.79	52.06±0.51	89.48±0.78	61.15±0.37	60.33±0.36	78.67±1.38	51.05±0.43	32.86±1.26	82.99±0.69	55.39±0.31	44.31±1.07	88.03 ± 1.59
Re-weight	63.16±1.53	52.39±0.32	90.16±0.51	62.21±0.44	61.12±0.54	79.02±2.41	50.91±0.35	32.79±0.65	82.86±1.38	56.78±0.69	48.12±1.25	91.12 ± 1.84
DR-GCN	67.77±1.09	67.67±0.74	87.23±0.28	55.33±0.23	46.56±0.43	67.45±1.01	46.84±1.42	34.54±1.33	72.48±0.83	24.86±1.27	30.93±1.76	64.53 ± 2.11
ImGAGN	63.60±0.55	62.89±0.60	91.87±0.53	63.21±1.25	62.13±0.87	78.32±2.34	48.04±0.78	36.14±1.01	80.61±0.21	60.69±1.25	42.55±1.91	91.25 ± 0.39
Gsmote	66.76±0.80	65.86±0.81	93.75±0.23	64.98±1.70	64.05±2.12	81.62±2.75	48.20±0.81	34.65±0.51	77.72±0.43	70.02±0.98	62.01±0.85	96.26 ± 0.04
GraphENS	72.68 ±0.76	67.94±0.94	94.32±0.54	69.98 ±2.41	69.53 ±2.31	87.46± 1.58	53.18 ±2.90	49.48 ±3.28	83.52±2.14	83.20±0.27	80.59±0.37	98.13 ± 0.06
Ours	**77.73**±0.75	**74.94**±0.29	**95.59**±0.33	**75.54**±1.12	**74.81**±0.65	**90.68**±0.40	**66.39** ±0.15	**61.97** ±0.14	**85.62** ±0.72	**85.68**±0.43	**84.21**±0.65	**99.69**±0.33
Dataset $\mathcal{H}_{edge}$	**Chameleon** 0.2795			**Film** 0.2516			**Squirrel** 0.2416			**Amazon-Photo** 0.8272		
Metrics(%)	ACC	F1	AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1	AUC
GCN	36.40±2.14	26.47±1.91	61.75±2.33	23.39±1.12	17.09 ±1.33	55.39±1.07	22.69±1.06	17.43±0.87	49.51±1.11	67.23±1.98	54.53±1.99	88.70 ± 1.77
ACM	38.16±0.86	28.33±0.82	62.43±1.45	24.56±1.08	18.56 ±0.76	57.41±0.47	24.22±1.28	18.96±0.92	51.26±1.28	68.66±1.73	56.28±0.63	90.65 ± 0.84
OverSamp	37.28±2.19	28.05±1.71	61.32±2.85	23.78±1.17	16.68± 1.05	56.02 ± 1.37	22.11±1.81	17.15±1.45	50.39±1.21	66.00±2.02	55.52±1.79	89.14 ± 1.53
Re-weight	36.40±1.36	27.59±1.25	59.47±1.61	27.98±1.34	20.95± 1.19	58.52±1.38	21.34±1.85	16.08±1.70	51.52±1.58	65.69±1.35	55.65±1.06	89.39 ±1.53
DR-GCN	37.36±2.85	28.78±2.44	60.34±1.63	19.03±0.75	15.23±0.46	47.43±0.49	15.57±1.34	11.62±1.34	47.28±0.82	65.92±1.64	60.90±1.25	84.35 ±4.49
ImGAGN	44.05±0.75	33.21±0.60	69.62±0.16	21.23±0.45	13.86±0.46	51.81±0.36	18.86 ±0.72	13.82±0.67	54.16±0.24	79.97±1.42	63.83±1.06	95.59 ±0.44
Gsmote	36.92±0.59	27.43±0.54	61.13±0.29	23.75±0.41	17.26 ±0.38	53.37 ±0.21	21.54±1.73	16.11±1.71	50.39±0.28	82.81±0.59	72.44±1.29	96.49 ±0.18
GraphENS	31.43±0.56	26.06±0.52	64.37±0.15	26.72±0.27	18.96±0.92	51.87±0.08	26.80±0.43	24.63±0.55	55.95±0.10	89.68±0.25	87.22±0.28	98.90 ±0.04
Ours	**49.01**±1.24	**48.29**±0.25	**77.07**±0.87	**30.20**±1.02	**26.53**±0.14	**61.41**±0.97	**27.89**±0.56	**26.07**±0.28	**57.64**±1.04	**91.56**±0.72	**90.43**±0.41	**99.43**±0.25

Table 2: Comparision of GraphSANN with other baselines in semi-supervised setting (*im_ratio*=0.1).We report the averaged accuracy, F1-score and AUC-ROC with the standard errors for 5 repetitions on six imitative imbalanced datasets for node classification. Here we use *OverSamp* and *Gsmote* as the abbreviations of *Oversampling* and *GraphSMOTE*, respectively.

5.1 Experimental Setup

Datasets. To thoroughly evaluate the performance of GraphSANN , we conduct experiments on eight benchmark datasets including six artificial imbalanced datasets and two genuine ones. Among the artificial datasets, Cora, Citeseer and Pubmed are three citation networks with high homophily, while Chameleon, Squirrel and Film are three Wikipedia networks with high heterophily. 3, 3, 2, 2, 2, and 2 classes are randomly selected as minority classes for these six datasets by down-sampling. Following [Zhao *et al.*, 2021], all majority classes have 20 nodes while minority classes only have 20 $\times$ *im_ratio* nodes in the training set. For two Amazon product networks whose class distributions are genuinely imbalanced, we use their original class ratios. The detailed statistical information of the six datasets is summarized in Table 1.

Baselines. We compare GraphSANN with eight state-of-the-art baselines for imbalanced node classification problem, including two vanilla models: *GCN* [Kipf and Welling, 2017] and *ACM* [Luan *et al.*, 2022]; two generic class-imbalance methods: *Oversampling* and *Reweight*; and four network-specific methods: *DR-GCN* [Shi *et al.*, 2020], *ImGAGN* [Qu *et al.*, 2021], *GraphSMOTE* [Zhao *et al.*, 2021] and *GraphENS* [Park *et al.*, 2022]. Please refer to Appendix 7.5 for detailed descriptions of each baseline.

Evaluation Metrics. Following existing works [Zhao *et al.*, 2021] in evaluating imbalanced classification, three evaluation metrics are adopted in this paper: Accuracy, AUC-ROC, and Macro-F1, where both AUC-ROC and Macro-F1 are reported by averaging the metrics over each class.

Parameter settings. The following hyper-parameters are set for our model in all the datasets. Layer number $L = 2$ with hidden dimensions 64 and 32 for both edge generation and node classification. Adam optimizer with learning rate $lr = 0.001$ for homophilic graphs and 0.01 for heterophilic graphs. Dropout rate $\gamma = 0.7$. *Epochs* $= 2000$ with early stop strategy. *Weight_decay*= 5e$-$4. Hyper-parameters $\kappa = 1.05$, $\omega = 0.3$. Initial hop $h = 2$, threshold $\eta = 0.5$ and loss weight $\lambda = 1\mathrm{e}{-6}$. Sampling ratio of candidate edges $\xi = 0.3$. Over-sampling scale $\zeta = 1.0$.

5.2 Imbalanced Node Classification (RQ1)

To answer RQ1, we compare the node classification performance of GraphSANN with other baselines across all eight datasets and report the average performances along with standard deviations of each metric. Table 2 shows the node classification results for six imitative datasets and two genuine datasets. From the table, we can observe that: (1) GraphSANN outperforms all the other baselines by all metrics on all eight datasets. This indicates our proposed model consistently acquires better performance on either homophilic or heterophilic networks. (2) On three heterophilic datasets, most class-imbalance baselines only acquire slightly better, or even worse performance (e.g. *DR-GCN* on Film and *GraphSMOTE* on Squirrel) than vanilla models i.e., *GCN* or *ACM*. This is because these baselines rely on homophilic assumption and generate synthetic edges based on feature similarity. Thus they perform poorly on heterophilic graphs whose edges link nodes with dissimilar features. GraphSANN, however, thanks to the adaptive subgraph extractor and multi-filter encoder blocks, can discriminatively aggregate similar node features to generate heterophilic edges, and thus achieve significant performance gains over the baselines. (3) On genuine datasets, compared to the most competitive baseline *GraphENS*, GraphSANN still acquires 4.5%

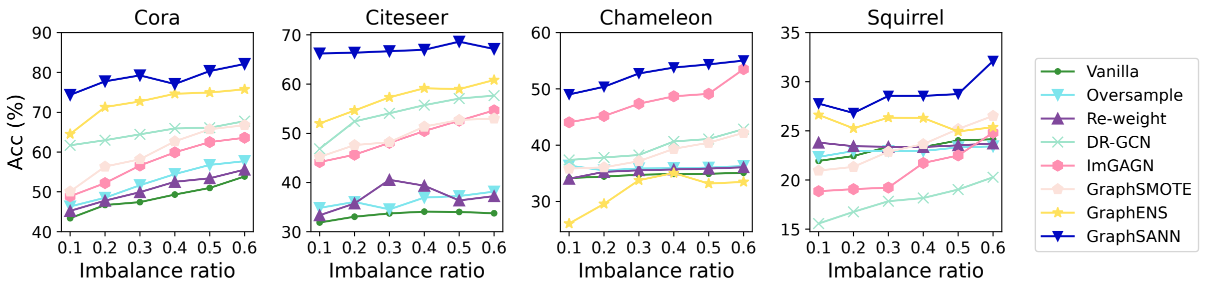

Figure 3: Node classification results under different imbalance ratios.

Method	Cora			Chameleon		
	ACC	F1	AUC	ACC	F1	AUC
w/o UFM	68.32	66.45	93.87	43.25	42.68	69.61
w/o ASE	75.14	73.21	94.21	45.65	43.95	72.25
w/o MSE	76.20	73.52	94.33	42.33	40.75	70.39
GraphSANN	**77.73**	**74.94**	**95.59**	**49.01**	**48.29**	**77.07**

Table 3: Ablation study results.

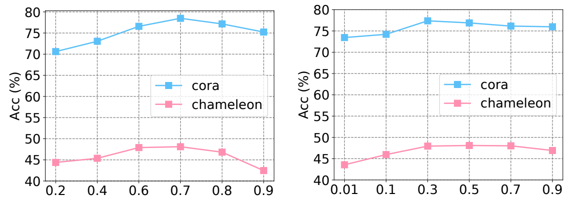

Figure 4: Hyper-parameter sensitivity analysis of dropout rate γ and candidate edge sampling ratio ξ.

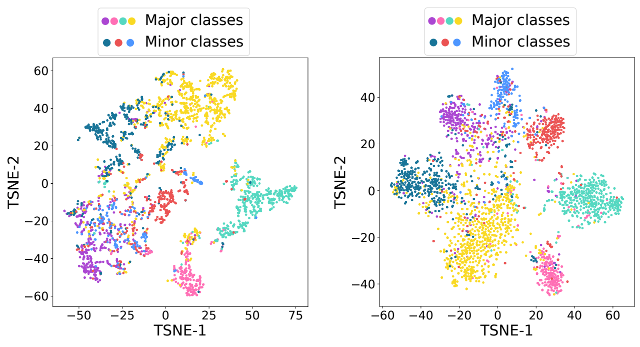

Figure 5: Visualization of GraphSANN and GraphENS.

and 3.6% performance gains w.r.t. F1-score on Amazon-Computers and Amazon-Photo, respectively.

5.3 Influence of Imbalance Ratio (RQ2)

To answer RQ2, in this subsection, we further compare the performance of GraphSANN with other baselines under different im_ratios. The imbalance ratio varies from 0.1 to 0.6. Each experiment is repeated 5 times and the average results are reported in Figure 3. From Figure 3, we can observe that (1) GraphSANN consistently outperforms other baselines across all the imbalance ratios on all the datasets, this demonstrates the generalization and robustness of our model under different imbalanced scenarios. (2) Generally, GraphSANN has more significant performance improvement over other baselines under more extreme imbalance ratios. As imbalance ratio increases, the datasets become more balanced, which offsets the effects brought by node/edge augmentation.

5.4 Ablation Study (RQ3)

To further investigate the contribution of each component of GraphSANN, we perform an ablation study and report the results in Table 3. *w/o UFM* replaces the unified feature mixer component with simple SMOTE strategy [Chawla *et al.*, 2002]; *w/o ASE* replaces the adaptive subgraph extractor with fixed 2-hop neighbors of target edge to form subgraphs; and *w/o MSE* replaces multi-filter subgraph encoder component with a raw GCN. GraphSANN represents the full model with all the components available. From this table, we can observe that: (1) All three components contribute to the performance improvement of GraphSANN ; (2) Adaptive subgraph extractor and multi-filter encoder exhibit crucial effects on heterophilic networks in view of sharp performance drops between *w/o ASE* and GraphSANN and between *w/o MSE* and GraphSANN on Chameleon.

5.5 Parameter Sensitivity Study (RQ4)

In this subsection, we investigate the impact of two crucial hyper-parameters, i.e., dropout rate γ of adaptive classifier and sampling ratio ξ of candidate synthetic edges on model performance. We vary γ from 0.1 to 0.9 with step size 0.1 and vary ξ from 0.01 to 0.9. The experiments are conducted on both Cora and Chameleon and test accuracy curves are shown in Figure 4. From Figure 4, we can observe that: (1) As γ increases, model performance gradually rises and reaches peak values when γ reaches 0.7 on both datasets. Then, as dropout rate keeps increasing, the performance gradually drops. (2) The performance decreases as the sampling ratio is under 0.3 or over 0.5. Our explanation is that the original edge distribution is not sufficiently simulated at low ξ, while a high ξ impedes feature aggregations by introducing too many noisy edges, resulting in performance degradation.

5.6 Visualization (RQ5)

In this subsection, we project the latent node embeddings of GraphSANN and the most competitive baseline *GraphENS* on Cora into two-dimensional space using t-SNE [Van der Maaten and Hinton, 2008] and color the nodes based on their class labels. As shown in Figure 5, we can observe that the minority class representations of *GraphENS* (e.g. blue and dark cyan dots) are hard to be distinguished and have large mixed areas with majority clusters, while node representations of GraphSANN are clustered tightly together with clear boundaries for both majority and minority classes. This proves the superiority of GraphSANN in terms of embedding quality for separating different classes despite the class-imbalance problem.

6 Conclusion

In this paper, we design a novel GraphSANN for imbalanced node classification on both homophilic and heterophilic graphs. The elaborately designed three components within it can unifiedly interpolate synthetic nodes, adaptively extracts surrounding subgraphs of candidate synthetic edges, and discriminatively encode them to predict the existence of synthetic edges. Extensive experiments on eight benchmark datasets have demonstrated the superiority of GraphSANN.

Acknowledgments

This work is partially supported by The National Key Research and Development Program of China (Grant No. 2020AAA0108504), Australian Research Council Future Fellowship (Grant No. FT210100624), Discovery Project (Grant No. DP190101985), and Discovery Early Career Research Award (Grant No.DE200101465).

References

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

[Bo *et al.*, 2021] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, 2021.

[Buda *et al.*, 2018] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural networks*, 106:249–259, 2018.

[Chawla *et al.*, 2002] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[Chen *et al.*, 2021] Deli Chen, Yankai Lin, Guangxiang Zhao, Xuancheng Ren, Peng Li, Jie Zhou, and Xu Sun. Topology-imbalance learning for semi-supervised node classification. *Advances in Neural Information Processing Systems*, 34:29885–29897, 2021.

[Jin *et al.*, 2021] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, 34:10654–10664, 2021.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[Liu *et al.*, 2023] Jie Liu, Lingyun Song, Guangtao Wang, and Xuequn Shang. Meta-hgt: Metapath-aware hypergraph transformer for heterogeneous information network embedding. *Neural Networks*, 157:65–76, 2023.

[Luan *et al.*, 2022] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. Revisiting heterophily for graph neural networks. *arXiv preprint arXiv:2210.07606*, 2022.

[Park *et al.*, 2022] Joonhyung Park, Jaeyun Song, and Eunho Yang. Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *International Conference on Learning Representations*, 2022.

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

[Shi *et al.*, 2020] 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.

[Shrikumar *et al.*, 2017] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[Sun *et al.*, 2021a] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Jiuxin Cao, Yingxia Shao, and Nguyen Quoc Viet Hung. Heterogeneous hypergraph embedding for graph classification. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 725–733, 2021.

[Sun *et al.*, 2021b] Xiangguo Sun, Hongzhi Yin, Bo Liu, Hongxu Chen, Qing Meng, Wang Han, and Jiuxin Cao. Multi-level hyperedge distillation for social linking prediction on sparsely observed networks. In *Proceedings of the Web Conference 2021*, pages 2934–2945, 2021.

[Sundararajan *et al.*, 2017] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[Xia *et al.*, 2021] Xin Xia, Hongzhi Yin, Junliang Yu, Yingxia Shao, and Lizhen Cui. Self-supervised graph co-training for session-based recommendation. In *Proceedings of the 30th ACM International conference on information & knowledge management*, pages 2180–2190, 2021.

[Yu *et al.*, 2020] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. Enhancing social recommendation with adversarial graph convolutional networks. *IEEE Transactions on knowledge and data engineering*, 34(8):3727–3739, 2020.

[Yuan and Ma, 2012] Bo Yuan and Xiaoli Ma. Sampling+ reweighting: Boosting the performance of adaboost on imbalanced datasets. In *The 2012 international joint conference on neural networks (IJCNN)*, pages 1–6. IEEE, 2012.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

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

[Zheng *et al.*, 2022] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082*, 2022.

[Zhou *et al.*, 2022] Shijie Zhou, Zhimeng Guo, Charu Aggarwal, Xiang Zhang, and Suhang Wang. Link prediction on heterophilic graphs via disentangled representation learning. *arXiv preprint arXiv:2208.01820*, 2022.