Anonymity Can Help Minority: A Novel Synthetic Data Over-sampling Strategy on Multi-label Graphs

Yijun Duan[1]✉, Xin Liu[1], Adam Jatowt[2], Hai-tao Yu[3], Steven Lynden[1], Kyoung-Sook Kim[1], and Akiyoshi Matono[1]

[1]AIRC, AIST, Japan
[2]Department of Computer Science, University of Innsbruck, Austria
[3] Faculty of Library, Information and Media Science, University of Tsukuba, Japan
{yijun.duan, xin.liu, steven.lynden, ks.kim, a.matono}@aist.go.jp
jatowt@acm.org
yuhaitao@slis.tsukuba.ac.jp

Abstract. In many real-world networks (e.g., social networks), nodes are associated with multiple labels and node classes are imbalanced, that is, some classes have significantly fewer samples than others. However, the research problem of imbalanced multi-label graph node classification remains unexplored. This non-trivial task challenges existing graph neural networks (GNNs) because the majority class could dominate the loss functions of GNNs and result in overfitting to those majority class features and label correlations. On non-graph data, minority over-sampling methods (such as SMOTE and its variants) have been demonstrated to be effective for the imbalanced data classification problem. This study proposes and validates a new hypothesis with unlabeled data oversampling, which is meaningless for imbalanced non-graph data; however, feature propagation and topological interplay mechanisms between graph nodes can facilitate representation learning of imbalanced graphs. Furthermore, we determine empirically that ensemble data synthesis through the creation of virtual minority samples in the central region of a minority, and the generation of virtual unlabeled samples in the boundary region between a minority and majority is the best practice for the imbalanced multi-label graph node classification task. Our proposed novel data over-sampling framework is evaluated using multiple real-word network datasets, and it outperforms diverse, strong benchmark models by a large margin.

Keywords: Imbalanced learning · graph representation learning · data over-sampling · generative adversarial network

1 Introduction

Graphs are becoming ubiquitous across a large spectrum of real-world applications in the forms of social networks, citation networks, telecommunication

networks, biological networks, etc. [32]. For a considerable number of real-world graph node classification tasks, the training data follows a *long-tail* distribution, and the node classes are *imbalanced*. In other words, a few majority classes have a significant fraction of samples, while most classes only contain a handful of instances. Taking the NCI chemical compound graph as an example, only about 5% of molecules are labeled as active in the anticancer bioassay test [25]. On the other hand, graph nodes are associated with multiple labels in many real-world networked data instead of a single one. Many social media sites, such as Flickr and YouTube, allow users to join diverse groups representing their various interests. A person can join several interest groups on Flickr, such as *Landscape* and *Travel*, and different video genres on YouTube, such as *Cooking* and *Wrestling*.

To date, a large body of work has been focused on the representation learning of graphs with balanced node classes and simplex labels [8, 11, 23, 29]. However, these models do not perform well on the widely-existing imbalanced and multi-label graphs because of the following reasons. (1) *Problem caused by the imbalanced setting*: The imbalanced data makes the classifier overfit the majority class, and the features of the minority class cannot be sufficiently learned [9]. Furthermore, the above problem is aggravated by the presence of the *topological interplay* effect [25] between graph nodes, making the feature propagation dominated by the majority classes. (2) *Problem caused by the multi-label setting*: Multi-label graph architectures typically encode significantly more complex interactions between nodes with shared labels [25], which is challenging to capture. Therefore, it is essential to develop a specific graph learning method for class imbalanced multi-label graph data. However, research in this direction is still in its infancy. Thus, in this study, we propose *imbalanced multi-label graph representation learning* to address this challenge while also contributing to graph learning theory.

Many past studies [2, 34, 35] have demonstrated that for imbalanced data, *minority over-sampling* is an effective measure to improve classification accuracy. This strategy has recently been confirmed to be still effective for graph data [33]. Traditional over-sampling techniques mainly consist of two steps: (1) selecting some minority instances as "seed examples"; (2) generating synthetic data with features and label similar to the seed examples and adding them into the training set. For example, the most popular over-sampling technique SMOTE [2] addresses the problem of minority generation by performing interpolation between randomly-selected minority instances and their nearest neighbors. However, mainstream over-sampling techniques have the following shortcomings when applied to graph data: (1) the selection of seed examples prioritizes global minority nodes while ignoring local minority nodes; (2) each synthetic instance is always assigned a label based on some specific strategy, which may be incorrect. Different from i.i.d. non-graph data, because the relationship between graph nodes are explicitly expressed by the edge connecting them, the representation learning of a node can be heavily dependent on its neighboring *unlabeled* nodes through the feature propagation mechanism on graphs.

Motivated by the observations above, we propose and validate the following assumption. In addition to synthetic minority samples, synthetic *unlabeled* samples can also facilitate the debiasing of GNNs on an imbalanced training set. In particular, for nearby global minority samples which are a local majority, we can "safely" produce virtual samples of the same class and add them into the training sets to balance class distribution. Global minority samples, which are also a local minority, are more likely to be local outliers and thus risky for selection as seed examples for further over-sampling; for nearby global minority samples whose neighbors are class-balanced, it is difficult to determine the labels of virtual samples. Thus, the production of *unlabeled* virtual nodes should be encouraged, which can help minorities by "blocking" the over-aggregation of majority features delivered through edges. This idea is illustrated in Fig. 1. **We argue that the key to over-sampling on an imbalanced multi-label graph is to flexibly combine the synthesis of both labeled and unlabeled instances enriched by label correlations**.

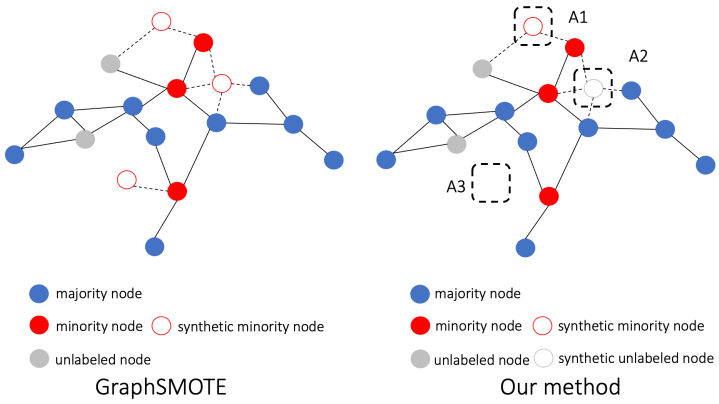

Fig. 1: A comparison between our method and the current state-of-the-art graph over-sampling method GraphSMOTE [33]. The latter's idea is to generate new minority instances near randomly selected minority nodes and create virtual edges (dotted lines in the figure) between those synthetic nodes and real nodes. Instead, we synthesize minority instances in safe areas (i.e., A1), generate *unlabeled* instances in locally balanced areas (i.e., A2), and do not conduct data over-sampling near minority nodes which are outliers (i.e., A3). For the simplicity of illustration, only a single-label scenario is shown.

We extend the existing over-sampling algorithms to a novel framework for the imbalanced multi-label graph node classification task based on the above considerations. We extend the classic global minority-based seed examples selection to the local minority perspective (see Sec. 4.1). Distinct from interpolation that is commonly-used in mainstream over-sampling techniques [18], we use a

generative adversarial network (GAN) [7] to generate new instances. As a representative deep generative model, GAN can capture label correlation information by estimating the probability distribution of seed examples [31]. We propose an ensemble architecture of GAN and cGAN [16] for the flexible generation of both unlabeled and labeled synthetics (see Sec. 4.2). To make use of the graph topology information, we propose to obtain new edges between generated samples and existing data with an edge predictor (see Sec. 4.3). The augmented graph is finally sent to a graph convolutional network (GCN) [11] for representation learning, together with the learned label correlations (see Sec. 4.4). We name our proposed framework as **SORAG**, which is abbreviated from **S**ynthetic data **O**versampling St**RA**tegy on **G**raph.

In summary, our contribution is three-fold:

- We advance the traditional simplex-label graph learning to an imbalanced multi-label graph learning setting, which is more general and common in real-world applications. To the best of our knowledge, this study is the first to focus on this task.
- We propose a novel and general framework which extends a previous oversampling algorithm to adapt to graph data. It flexibly ensembles the synthesis of labeled and unlabeled nodes to support the minority classes and leverage label correlations to generate more natural nodes.
- Extensive experiments on multiple real-world datasets demonstrate the high effectiveness of our approach. Compared with the current state-of-the-art model GraphSMOTE [33], our method has an improvement of 1.5% in terms of Micro-F1 and 3.3% in terms of Macro-F1 on average.

2 Related Works

2.1 Graph Neural Networks

Graph representation learning (GRL) has evolved considerably in recent years. GNN can be broadly regarded as the third (and latest) generation of GRL after traditional graph embedding and modern graph embedding [15]. GNNs can be classified into spatial and spectral types based on their graph filter. Spatial-based graph filters explicitly leverage the graph structure. Representative works in this field include the GraphSAGE filter [8], GAT-filter [29], the ECC-filter [26], GGNN-filter [12], Mo-filter [17], and so on. Spectral-based graph filters use graph spectral theory to design filtering operations in the spectral domain. An early work [1] deals with the eigendecomposition of the Laplacian matrix and the matrix multiplication between dense matrices, thus being computationally expensive. To overcome this problem, the Poly-Filter [3], Cheby-Filter [3], and GCN-Filter [11] have been successively proposed. In particular, our task is semi-supervised, which means we need to learn the representation of all nodes from a small portion of labeled nodes. Some recent works on semi-supervised graph node classification can be found in [15].

2.2 Imbalanced Learning

Learning from imbalanced data has been a long-standing challenge in machine learning. With an imbalanced class distribution, existing methods addressing this issue can be grouped into three categories [9]: (1) pre-processing the training data, (2) post-processing the output, and (3) direct learning methods. Data pre-processing aims to make the classification results on the new training set equivalent to imbalance-aware classification decisions on the original training set, typically like sampling [5] and weighting [35]. Post-processing the output makes the classifier biased toward minority classes by adjusting the classifier decision threshold [4, 24]. Direct learning methods embed class distribution information into the component (e.g., objective function) of the learning algorithm, with typical methods being cost-sensitive decision tree [14], cost-sensitive SVM [19], and so on. Studies on multi-class single-label imbalanced GRL have emerged only recently [25, 30, 33]. However, different from these works, our proposal is the first to utilize synthetic unlabeled nodes to weaken the tendency of GNNs to overfit to majority without introducing contradictory labels. Additionally, our proposed model is also applicable to multi-label datasets.

3 Problem Formulation

Input. The input is a graph $G = \{V, A, X, L, B\}$. $V = \{v_1, v_2, ..., v_n\}$ denotes the set of nodes. $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix. $A_{ij} = 1$ when there is an undirected edge between nodes v_i and v_j; otherwise $A_{ij} = 0$. The self-loops in G have been removed, so $A_{ii} = 0$, $i \in \{1, 2, ..., n\}$. $X \in \mathbb{R}^{n \times k}$ is the feature matrix, where $x_i \in \mathbb{R}^{1 \times k}$ is the feature vector of node v_i. $L = \{c_1, c_2, ..., c_m\}$ is a set of unique labels. B is a $n \times m$ affiliation matrix of labels with $B_{ij} = 1$ if v_i has label c_j; otherwise $B_{ij} = 0$. Our task is in a semi-supervised transductive manner. Only a tiny portion of the nodes is used for training, which we denote as V^{train}.

Output. Our goal is to learn a graph neural network f that maps the input graph G into a dense vector representation $Z \in \mathbb{R}^{n \times d}$, where $z_i \in \mathbb{R}^{1 \times d}$ is the vector of node v_i, and predicts the class labels for the test nodes set V^{test}.

Imbalanced learning. Let $|c_i|$ represent the number of samples associated with the label c_i. The distribution of $\{|c_1|, |c_2|, ..., |c_m|\}$ is imbalanced. That is, a few labels contain most samples, and most labels contain only a few samples. When presented with imbalanced data, existing GNNs tend to bias toward majority groups, leaving minority instances under-trained. We aim to learn a neural network classifier f that can work well for both majority and minority classes.

4 Methodology

An illustration of the proposed framework is shown in Fig. 2. We elaborate on each component as follows.

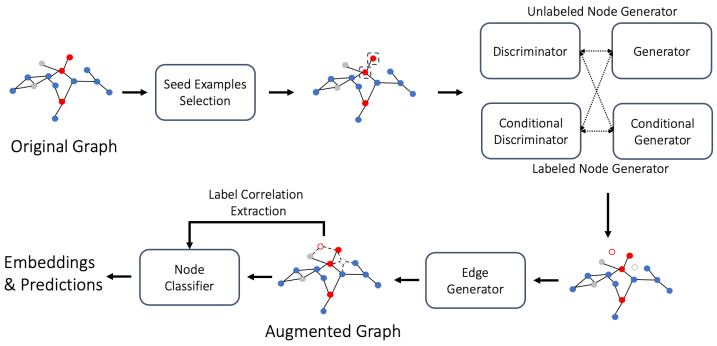

Fig. 2: Overview of the proposed method.

4.1 Imbalance Measurement

In multi-label learning, a commonly used measure that evaluates the global imbalance of a particular label is *IRLbl*. Let $|C_i|$ be the number of instance whose i-th label value is 1; *IRLbl* is then defined as follows.

$$IRLbl_i = \frac{max\left\{|c_1|, |c_2|, ..., |c_m|\right\}}{c_i}. \tag{1}$$

Therefore, the larger the value of *IRLbl* for a label, the more minority class it is. For a node v_i, its GMD is defined as follows.

$$GMD_i = \frac{IRLbl_j \cdot [B_{ij} = 1]}{\sum_{j=1}^{m} [B_{ij} = 1]}, \tag{2}$$

where $[B_{ij} = 1]$ means v_i has the j-th label, and $\sum_{j=1}^{m} [B_{ij} = 1]$ counts the number of labels v_i has.

The local minority degree (LMD) of a node can be measured by the proportion of opposite class values in its local neighborhood. For v_i, let N_i^k denote its K-hop neighbor nodes. Then, for label c_j, the proportion of neighbors having an opposite class to the class of v_i is computed as

$$S_{ij} = \frac{\sum_{v_m \in N_i^k} [B_{ij} \neq B_{mj}]}{|N_i^k|}, \tag{3}$$

where $S \in \mathbb{R}^{n \times m}$ is a matrix defined to store the local imbalance of all nodes for each label. Given S, a straightforward way to compute LMD for v_i is to average its S_{ij} for all labels as follows.

$$LMD_i = \frac{\sum_{j=1}^{m} S_{ij}[B_{ij} = g_j]}{m}, \tag{4}$$

where $g_j \in \{0, 1\}$ denotes the minority class of j-th label. Namely, if $|c_j| \geq 0.5 \cdot n$, $g_j = 1$; else, $g_j = 0$. Here, n is the total number of vertices. Further, we group global minority nodes into different types based on LMD, and each type is identified correctly by the classifier with different difficulties. Following [13, 20], we discretize the range [0,1] of LMD_i to define four types of nodes, namely safe (SF), borderline (BD), rare (RR) and outlier (OT), according to their local imbalance.

- SF: $0 \leq LMD_i < 0.3$. Safe nodes are basically surrounded by nodes containing similar labels.
- BD: $0.3 \leq LMD_i < 0.7$. Borderline nodes are located in the decision boundary between different classes.
- RR: $0.7 \leq LMD_i < 1.0$. Rare nodes are located in the region overwhelmed by different nodes and distant from the decision boundary.
- OT: $LMD_i = 1.0$. Outliers are totally connected to different nodes.

Furthermore, for v_i, we define two metrics: labeled seed probability (LSP) and unlabeled seed probability (USP) to describe the probability of being selected as a seed example to generate labeled synthetic nodes and unlabeled synthetic nodes, respectively. The LSP and USP are calculated as follows.

$$LSP_i = GMD_i \cdot LMD_i, v_i \in SF \tag{5}$$

$$USP_i = GMD_i \cdot LMD_i, v_i \in BD \tag{6}$$

We compute the LSP and USP scores for all nodes and sort them in descending order. The top-ranked nodes (controlled by the hyper-parameter seed example rate ρ) will be selected as seed examples. A min-max normalization processes all the GMD and LMD scores to improve the computation stability.

4.2 Node Generator

We denote the joint distribution of node feature x and label y in SF region as $P_{SF}(x, y)$, the marginal distribution of y as $P_{SF}(y)$, and the marginal distribution of x in BD region as $P_{BD}(x)$. Generator G_l is expected to generate labeled instances in the SF region, while generator G_u should output unlabeled synthetics in the BD region. Let the data distribution produced by G_l and G_u be denoted as $P_l(x, y)$ and $P_u(x)$, respectively; then, we expect $P_{BD}(x) \approx P_u(x)$ and $P_{SF}(x, y) \approx P_l(x, y)$. Furthermore, a more flexible goal is to have $P_{BD}(x) \approx \alpha \cdot P_u(x) + (1-\alpha) \cdot P_l(x)$, $P_{SF}(x, y) \approx \beta \cdot P_l(x, y) + (1-\beta) \cdot P_u(x, y)$, $\alpha \approx 1, \beta \approx 1$. By adjusting the values of α and β, we can control G_l and G_u to produce various data distributions to fit the original data. Here, $P_u(x, y)$ is the joint distribution of $P_u(x)$ and $P_{SF}(y)$, and $P_l(x)$ is the marginal distribution of $P_l(x, y)$.

To achieve the above goal, we propose a node generator, which is essentially an ensemble of a GAN [7] and a conditional GAN (cGAN) [16]. The GAN is

responsible for generating unlabeled synthetic nodes, whose generator and discriminator are respectively denoted as G_u and D_u. The cGAN is used for generating labeled synthetic instances, where its generator and discriminator are denoted as G_l and D_l, respectively. Our loss function for training the GAN is

$$\min_{G_u} \max_{D_u} \mathcal{L}_{GAN} = \mathbb{E}_{x \sim P_{BD}(x)} log D_u(x) + \alpha \cdot \mathbb{E}_{x \sim P_u(x)} log(1 - D_u(x)) \tag{7}$$

For cGAN, our objective is given as

$$\min_{G_l} \max_{D_l} \mathcal{L}_{cGAN} = \mathbb{E}_{(x,y) \sim P_{SF}(x,y)} log D_l(x,y) + \beta \cdot \mathbb{E}_{(x,y) \sim P_l(x,y)} log(1 - D_l(x,y)) \tag{8}$$

To achieve flexible control over G_l and G_u, we design the following loss function based on the interaction of GAN and cGAN

$$\begin{aligned} \min_{G_u, G_l} \max_{D_u, D_l} \mathcal{L}_{GAN-cGAN} = (1-\alpha) \cdot \mathbb{E}_{x \sim P_l(x)} log(1 - D_u(x)) \\ + (1-\beta) \cdot \mathbb{E}_{(x,y) \sim P_u(x,y)} log(1 - D_l(x,y)) \end{aligned} \tag{9}$$

Putting all these together, our final loss for node generation $\mathcal{L}_{node}$ is

$$\mathcal{L}_{node} = \min_{G_u, G_l} \max_{D_u, D_l} \mathcal{L}_{GAN} + \mathcal{L}_{cGAN} + \mathcal{L}_{GAN-cGAN} \tag{10}$$

For our proposed generator, the following theoretical analysis is performed.

Proposition 1 *For any fixed G_u and G_l, the optimal discriminator D_u and D_l of the game defined by $\mathcal{L}_{node}$ is*

$$D_u^*(x) = \frac{P_{BD}(x)}{P_{BD}(x) + P_\alpha(x)}, D_l^*(x,y) = \frac{P_{SF}(x,y)}{P_{SF}(x,y) + P_\beta(x,y)} \tag{11}$$

where $P_\alpha(x) = \alpha \cdot P_u(x) + (1-\alpha) \cdot P_l(x)$, and $P_\beta(x,y) = \beta \cdot P_l(x,y) + (1-\beta) \cdot P_u(x,y)$.

Proof. We have

$$\begin{aligned} \mathcal{L}_{node} = & \int_x P_{BD}(x) log D_u(x) dx + \int_{x,y} P_{SF}(x,y) log D_l(x,y) dxdy \\ & + \alpha \cdot \int_x P_u(x) log(1 - D_u(x)) dx + \beta \cdot \int_{x,y} P_l(x,y) log(1 - D_l(x,y)) dxdy \\ & + (1-\alpha) \cdot \int_x P_l(x) log(1 - D_u(x)) dx + (1-\beta) \cdot \int_{x,y} P_u(x,y) log(1 - D_l(x,y)) dxdy \\ = & \int_x P_{BD}(x) log D_u(x) + P_\alpha(x) \cdot log(1 - D_u(x)) dx \\ & + \int_{x,y} P_{SF}(x,y) log D_l(x,y) + P_\beta(x,y) \cdot log(1 - D_l(x,y)) dxdy \end{aligned} \tag{12}$$

For any $(a, b) \in \mathbb{R}^2 \backslash \{0, 0\}$, the function $f(y) = alogy + blog(1-y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. This concludes the proof.

Proposition 2 *The equilibrium of $\mathcal{L}_{node}$ is achieved if and only if $P_{BD}(x) = P_\alpha(x)$ and $P_{SF}(x, y) = P_\beta(x, y)$ with $D_u^*(x) = D_l^*(x, y) = \frac{1}{2}$, and the optimal value of $\mathcal{L}_{node}$ is -4log2.*

Proof. When $D_u(x) = D_u^*(x), D_l(x, y) = D_l^*(x, y)$, we have

$$\begin{aligned}
\mathcal{L}_{node} &= \int_x P_{BD}(x) log \frac{P_{BD}(x)}{P_{BD}(x) + P_\alpha(x)} dx + \int_{x,y} P_{SF}(x, y) log \frac{P_{SF}(x, y)}{P_{SF}(x, y) + P_\beta(x, y)} dxdy \\
&+ \int_x P_\alpha(x) log \frac{P_\alpha(x)}{P_{BD}(x) + P_\alpha(x)} dx + \int_{x,y} P_\beta(x, y) log \frac{P_\beta(x, y)}{P_{SF}(x, y) + P_\beta(x, y)} dxdy \\
&= -4log2 + 2 \cdot JSD(P_{BD}(x)||P_\alpha(x)) + 2 \cdot JSD(P_{SF}(x, y)||P_\beta(x, y)) \\
&\geq -4log2
\end{aligned} \tag{13}$$

where the optimal value is achieved when the two Jensen-Shannon divergences are equal to 0, namely, $P_{BD}(x) = P_\alpha(x)$, and $P_{SF}(x, y) = P_\beta(x, y)$. When $\alpha = \beta = 1$, we have $P_{BD}(x) = P_u(x), P_{SF}(x, y) = P_l(x, y)$.

In the implementation, both G_u and G_l are designed as a 3-layer feed-forward neural network. In contrast, D_u and D_l are designed with a relatively weaker structure: a 1-layer feed-forward neural network for facilitating the training.

4.3 Edge Generator

The edge generator described in this section is responsible for estimating the relation between virtual nodes and real nodes, which facilitates feature propagation, feature extraction, and node classification. Such edge generators will be trained on real nodes and existing edges. Following a previous work [33], the inter-node relation is embodied in the weighted inner product of node features. Specifically, for two nodes v_i and v_j, let E_{ij} denote the probability of the existence of an edge between them, which is computed as

$$E_{ij} = \sigma(x_i \cdot W^{edge} \cdot x_j^T) \tag{14}$$

where x_i and x_j are the feature vectors of v_i and v_j, respectively. $W^{edge} \in \mathbb{R}^{k \times k}$ is the weight parameter matrix to be learned, and $\sigma = Sigmoid()$. Then, the extended adjacency matrix A' is defined as follows

$$A'_{ij} = \begin{cases} A_{ij}, & \text{if } v_i \text{ and } v_j \text{ are real nodes} \\ E_{ij}, & \text{if } v_i \text{ or } v_j \text{ is synthetic node} \end{cases} \tag{15}$$

Compared to A, A' contains new information about virtual nodes and edges, which will be sent to the node classifier in Sec. 4.4. As the edge generator is

expected to be partially trained based on the final node classifier (see Sec. 4.5), predicted edges should be set as continuous so that the gradient can be calculated and propagated from the node classifier. Thus, E_{ij} is not discretized to some value in {0,1}. The edge generator should be capable of predicting real edges accurately to generate realistic virtual nodes. Then, the pre-trained loss function for training the edge generator is

$$\mathcal{L}_{edge} = \|E - A\|^2 \tag{16}$$

where E refers to predicted edges between real nodes.

4.4 Node Classifier

We now obtain an augmented balanced graph $G' = \{V', A', X', B'\}$, where V' consists of both real nodes and synthetic labeled and unlabeled nodes; further, A', X', and B' denote the edge, feature, and label information of the enlarged vertex set, respectively. A classic two-layer GCN structure [11] is adopted for node classification, given its high accuracy and efficiency. Its first and second layers are denoted as L^1 and L^2, respectively, and their corresponding outputs $\{O^1, O^2\}$ are

$$O^1 = ReLU(\tilde{D}^{-\frac{1}{2}}\tilde{A}'\tilde{D}^{-\frac{1}{2}}X'W^1) \tag{17}$$

$$O^2 = \sigma(F\tilde{D}^{-\frac{1}{2}}\tilde{A}'\tilde{D}^{-\frac{1}{2}}O^1W^2) \tag{18}$$

where $\tilde{A}' = A' + I$, I is an identity matrix of the same size as A'. $\tilde{D}$ is a diagonal matrix and $\tilde{D}_{ii} = \sum_j \tilde{A}'_{ij}$. $\tilde{D}^{-\frac{1}{2}}\tilde{A}'\tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix. Further, W^1 and W^2 are the learnable parameters in the first and second layers, respectively. $ReLU$ and σ are the respective activation functions of the first and the second layer, where $ReLU(Z)_i = max(0, Z_i)$, $\sigma(Z)_i = Sigmoid(Z)_i = \frac{1}{1+exp(-Z_i)}$. O^2 is the posterior probability of the class to which the node belongs. F is the label correlation matrix that is computed in the same way as in [25], which provides helpful extra-label correlation and interaction information. Eventually, given the training labels B^{train}, we minimize the following cross-entropy error to learn the classifier, where p is the number of training samples, m is the size of the label set, and nc stands for node classifier.

$$\mathcal{L}_{nc} = -\sum_{i=1}^{p}\sum_{j=1}^{m} B_{ij}^{train} ln O_{ij}^2 \tag{19}$$

4.5 Optimization Objective

Based on the above content, the final objective function of our framework is given as

$$\min_{\Theta,\Phi,\Psi} \mathcal{L}_{nc} + \lambda \cdot \mathcal{L}_{node} + \mu \cdot \mathcal{L}_{edge} \tag{20}$$

where Θ, Φ, and Ψ are the sets of parameters for the synthetic node generator (Sec. 4.2), edge generator (Sec. 4.3) and node classifier (Sec. 4.4), respectively. λ and μ are weight parameters. The best training strategy in our experiments is to pre-train the node generator and the edge generator first, and then minimize Eq. (20) to train the node classifier and fine-tune the node generator and edge generator at the same time. Our entire framework is easy to implement, general, and flexible. Different structural choices can be adopted for each component, and different regularization terms can be enforced to provide prior knowledge.

4.6 Training Algorithm

The 1 algorithm illustrates the proposed framework. **SORAG** is trained through the following components: (1) the selection of seed examples based on node LSP and USP scores; (2) the pre-training of the node generator (i.e., the ensemble of GAN and cGAN) for synthetic data generation; (3) the pre-training of the edge generator to produce new relation information; and finally, (4) the training of the node classifier on top of the over-sampled graph and the fine-tuning of node generator and edge generator.

5 Experimental Settings

5.1 Datasets

We use three multi-label networks: BlogCatalog3, Flickr, and YouTube as benchmark datasets. In Table 1, we list the statistical information of all datasets used, including the number of nodes, the number of edges, the number of node classes, and the tuned optimal value of key parameters of $\mathbf{SORAG}_F$: {learning rate, weight decay, dropout rate, k (Sec. 4.1), ρ (Sec. 4.1), α (Sec. 4.2), β (Sec. 4.2), λ (Sec. 4.5), μ (Sec. 4.5)}. For each dataset, we assume that a majority class is one with more samples than the average class size, while a minority class is one with less samples. Below is a brief description of each dataset used.

- BlogCatalog3 [27] is a network of social relationships provided by blogger authors. The labels represent the topic categories provided by the authors, such as *Education*, *Food*, and *Health*. This network contains 10,312 nodes, 333,983 edges, and 39 labels.
- Flickr [27] is a network of contacts between users of the photo-sharing website. The labels represent the interest groups of the users, such as *black and white photos*. This network contains 80,513 nodes, 5,899,882 edges, and 195 labels.
- YouTube [28] is a social network between users of the popular video sharing website. The labels represent groups of viewers that enjoy common video genres such as *anime* and *wrestling*. This network contains 1,138,499 nodes, 2,990,443 edges and 47 labels.

Algorithm 1 Full Training Algorithm

Inputs: Graph data: $G = \{V, A, X, L, B\}$
Outputs: Network parameters, node representations, and predicted node class

Initialize the node generator, edge generator, and node classifier
Compute the node LSP and USP scores based on Eq. (5) and Eq. (6), respectively
Select the fraction of nodes with the highest LSP and USP scores as seed examples for D_l and D_u, respectively
while Not Converged **do** ▷ Pre-train the node generator
    Update D_l by ascending along its gradient based on $\mathcal{L}_{node}$ (Eq. (10))
    Update G_l by descending along its gradient based on $\mathcal{L}_{node}$
    Update D_u by ascending along its gradient based on $\mathcal{L}_{node}$
    Update G_u by descending along its gradient based on $\mathcal{L}_{node}$
end while
while Not Converged **do** ▷ Pre-train the edge generator
    Update the edge generator by descending along its gradient based on $\mathcal{L}_{edge}$ (Eq. (16))
end while
Construct label-occurrence network and extract label correlations [25]
while Not Converged **do**▷ Train the node classifier and pre-train the other components
    Generate new unlabeled nodes using G_u
    Generate new labeled nodes using G_l
    Generate the new adjacency matrix A' using the edge generator
    Update the full model based on $\mathcal{L}_{nc} + \lambda \cdot \mathcal{L}_{node} + \mu \cdot \mathcal{L}_{edge}$ (Eq. (20))
end while
Predict the test set labels with the trained model

For all datasets, we attribute each node with a 64-dim embedding vector obtained by performing dimensionality reduction on the adjacency matrix using PCA [6], similar to [25, 33]. All of the above datasets are available at `http://zhang18f.myweb.cs.uwindsor.ca/datasets/`.

5.2 Analyzed Methods and Metrics

To validate the performance of our approach, we compare it against a number of state-of-the-art and representative methods for multi-label graph learning and imbalanced graph learning, which include **GCN** [11], **ML-GCN** [25], **SMOTE** [2], **GraphSMOTE** [33], and **RECT** [30]. Additionally, three variants of our proposed method are implemented, which are **SORAG**$_F$ (the full model), **SORAG**$_L$ (only labeled nodes are generated), and **SORAG**$_U$ (only unlabeled nodes are generated).

It is necessary to mention that all the baselines above except **ML-GCN** (which is intrinsically designed as a multi-label classifier) are manually set to

Table 1: Dataset statistics.

Dataset	BlogCatalog3	Flickr	YouTube
# Nodes	10,312	333,983	39
# Edges	80,513	5,899,882	195
# Classes	1,138,499	2,990,443	47
learning rate	0.05	0.01	0.1
weight decay	5e-4	1e-4	1e-3
dropout rate	0.5	0.5	0.9
k	2	2	2
ρ	0.5	0.5	0.5
α	0.9	0.5	0.8
β	0.8	0.9	0.8
λ	0.1	1	1
μ	1	1	1

conduct the multi-label node classification by modifying the last layer of their network structure. The implementation of the baseline approaches relies on publicly released code from relevant sources[1][2][3][4]. We adopt Micro-F1 and Macro-F1 to evaluate the model performance, which are commonly used in imbalanced data classification.

5.3 Training Configurations

Following the semi-supervised learning setting, we randomly sample a portion of the labeled nodes (i.e., sampling ratio) of each dataset and use them for evaluation. Then, we randomly split the sampled nodes into 60% / 20% / 20% for training, validation, and testing, respectively. Similar to [22], the sampling ratios for the BlogCatalog3 network, the Flickr network, and the YouTube network are set as 10%, 1%, and 1%, respectively. To make the class size balanced, we experiment with different over-sampling rates, and finally they are set as those in Tab. 2. All the analyzed models are trained using Adam optimizer [10] in PyTorch (2020.2.1, community edition) [21]. Each result is presented as a mean based on 10 replicated experiments. All models are trained until they converge, with a typical number of training epochs as 200.

[1] **SMOTE**: `https://github.com/analyticalmindsltd/smote_variants`
[2] **GraphSmote**: `https://github.com/TianxiangZhao/GraphSmote`
[3] **RECT**: `https://github.com/zhengwang100/RECT`
[4] **GCN**: `https://github.com/tkipf/pygcn`

Table 2: The optimal over-sampling rates for the synthetic unlabeled nodes (denoted as Rate_U) and the synthetic labeled nodes (denoted as Rate_L) on each dataset. N/A abbreviates for "not applicable".

	BlogCatalog3		Flickr		YouTube	
	Rate_U	Rate_L	Rate_U	Rate_L	Rate_U	Rate_L
SORAG$_U$	0.9	N/A	0.6	N/A	0.7	N/A
SORAG$_L$	N/A	0.1	N/A	0.3	N/A	0.6
SORAG$_F$	0.1	0.9	0.2	0.9	0.2	0.4

6 Experimental Results

6.1 Imbalanced Multi-label Classification Performance

Table 3 shows the performance of all methods in terms of Micro-F1 and Macro-F1. The results are presented as a mean based on 10 repeated experiments. Based on the results, we reach the following conclusions.

Table 3: Imbalanced multi-label classification comparison. The 1^{st} and 2^{nd} best results are boldfaced and underscored, respectively.

Metrics	Micro-F1 (%)			Macro-F1 (%)		
Methods \Datasets	BlogCatalog3	Flickr	YouTube	BlogCatalog3	Flickr	YouTube
GCN	37.36	34.03	36.19	30.27	21.17	26.53
ML-GCN	37.51	38.91	37.64	30.39	21.56	27.52
SMOTE	40.24	39.30	39.01	30.65	23.08	28.53
GraphSMOTE	42.82	40.01	**43.70**	35.58	24.25	33.81
RECT	41.72	41.23	42.66	<u>38.66</u>	24.47	33.94
SORAG$_L$	<u>44.58</u>	<u>41.61</u>	41.98	38.45	<u>26.48</u>	<u>35.01</u>
SORAG$_U$	43.21	37.92	40.83	37.28	26.15	32.53
SORAG$_F$	**44.89**	**43.13**	<u>42.86</u>	**40.01**	**26.85**	**36.57**

- When compared with the GCN and ML-GCN methods, which do not consider class distribution, the three variants of **SORAG** show significant improvements. For example, compared with ML-GCN, the improvement brought by **SORAG**$_F$ is 7.4%, 4.2%, and 5.2% in terms of Micro-F1 and 9.6%, 5.3%, and 9.1% in terms of Macro-F1, respectively. This demonstrates that our proposed data over-sampling strategy effectively enhances the classification performance of GNNs on imbalanced multi-label graph data.
- **SORAG** provides much more benefits than when applying the previous imbalanced graph node classifier (SMOTE, GraphSMOTE, RECT). On average, it outperforms earlier methods by 3.3%, 3.0%, and 1.1% in terms of Micro-F1 and 2.5%, 2.9%, and 4.5% in terms of Macro-F1, respectively. This result validates the advantage of **SORAG** over previous over-sampling techniques in combining the generation of minority and unlabeled samples.

– Both minority over-sampling and unlabeled data over-sampling can improve classification performance. In particular, the former is more effective. A combination of the two strategies works the best. As a supporting evidence, $\mathbf{SORAG}_F$ is the best performer in 5/6 tasks and the second-best performer in the remaining task.

7 Conclusions

This study investigated a new research problem: imbalanced multilabel graph node classification. In contrast to existing oversampling algorithms, which only generate new minority instances to balance the class distribution, we proposed a novel data generation strategy named **SORAG** which ensembles the synthesis of labeled instances in minority class centers and unlabeled instances in minority class borders. The new supervision information brought about by labeled synthetics and the blocking of over-propagated majority features by unlabeled synthetics facilitates balanced learning between different classes, taking advantage of the strong topological interdependence between nodes on a graph.

We conducted extensive comparative studies to evaluate the proposed framework on diverse naturally imbalanced multilabel networks. The experimental results demonstrated the high effectiveness and robustness of **SORAG** in handling imbalanced data. In the future, we will work on developing graph neural network models that are more adapted to the nature of real-world networks (e.g., scale-free and small-world features, etc.).

8 Acknowledgements

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO). We would also like to acknowledge partial support from JSPS Grant-in-Aid for Scientific Research (Grant Number 21K12042).

References

1. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203 (2013)
2. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority over-sampling technique. JAIR **16**, 321–357 (2002)
3. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems **29**, 3844–3852 (2016)
4. Domingos, P.: Metacost: A general method for making classifiers cost-sensitive. In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 155–164 (1999)
5. Elkan, C.: The foundations of cost-sensitive learning. In: International joint conference on artificial intelligence. vol. 17, pp. 973–978. Lawrence Erlbaum Associates Ltd (2001)

6. F.R.S., K.P.: Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **2**(11), 559–572 (1901). https://doi.org/10.1080/14786440109462720
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
8. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. pp. 1025–1035 (2017)
9. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Transactions on knowledge and data engineering **21**(9), 1263–1284 (2009)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
12. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015)
13. Liu, B., Blekas, K., Tsoumakas, G.: Multi-label sampling based on local label imbalance. Pattern Recognition **122**, 108294 (2022)
14. Lomax, S., Vadera, S.: A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys (CSUR) **45**(2), 1–35 (2013)
15. MA, Y.T.: DEEP LEARNING ON GRAPHS. Cambridge University Press (2021)
16. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
17. Monti, F., Bronstein, M.M., Bresson, X.: Geometric matrix completion with recurrent multi-graph neural networks. arXiv preprint arXiv:1704.06803 (2017)
18. More, A.: Survey of resampling techniques for improving classification performance in unbalanced datasets. arXiv preprint arXiv:1608.06048 (2016)
19. Morik, K., Brockhausen, P., Joachims, T.: Combining statistical learning with a knowledge-based approach: a case study in intensive care monitoring. Tech. rep., Technical Report (1999)
20. Napierala, K., Stefanowski, J.: Types of minority class examples and their influence on learning classifiers from imbalanced data. Journal of Intelligent Information Systems **46**(3), 563–597 (2016)
21. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32**, 8026–8037 (2019)
22. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)
23. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE transactions on neural networks **20**(1), 61–80 (2008)
24. Sheng, V.S., Ling, C.X.: Thresholding for making classifiers cost-sensitive. In: AAAI. vol. 6, pp. 476–481 (2006)
25. Shi, M., Tang, Y., Zhu, X., Liu, J.: Multi-label graph convolutional network representation learning. IEEE Transactions on Big Data (2020)
26. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3693–3702 (2017)

27. Tang, L., Liu, H.: Relational learning via latent social dimensions. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 817–826 (2009)
28. Tang, L., Liu, H.: Scalable learning of collective behavior based on sparse social dimensions. In: Proceedings of the 18th ACM conference on Information and knowledge management. pp. 1107–1116 (2009)
29. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
30. Wang, Z., Ye, X., Wang, C., Cui, J., Yu, P.: Network embedding with completely-imbalanced labels. IEEE Transactions on Knowledge and Data Engineering (2020)
31. Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling tabular data using conditional gan. In: Advances in NIPS (2019)
32. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Network representation learning: A survey. IEEE transactions on Big Data **6**(1), 3–28 (2018)
33. Zhao, T., Zhang, X., Wang, S.: Graphsmote: Imbalanced node classification on graphs with graph neural networks. In: Proceedings of the 14th ACM international conference on web search and data mining. pp. 833–841 (2021)
34. Zhou, Z.H., Liu, X.Y.: Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Transactions on knowledge and data engineering **18**(1), 63–77 (2005)
35. Zhou, Z.H., Liu, X.Y.: On multi-class cost-sensitive learning. Computational Intelligence **26**(3), 232–257 (2010)