Adaptive Graph Enhancement for Imbalanced Multi-relation Graph Learning

Yiyue Qian[†,‡]
Amazon
Seattle, Washington, USA
yyqian5@gmail.com

Tianyi Ma[‡]
University of Notre Dame
Notre Dame, Indiana, USA
tma2@nd.edu

Chuxu Zhang
University of Connecticut
Storrs, Connecticut, USA
chuxu.zhang@uconn.edu

Yanfang Ye*
University of Notre Dame
Notre Dame, Indiana, USA
yye7@nd.edu

ABSTRACT

Graph Neural Networks (GNNs), as the mainstream graph representation learning method, has demonstrated its effectiveness in learning graph embeddings over benchmark datasets. However, existing GNNs still have limitations in handling real-world graphs in the following aspects: (i) nodes in most real-world graphs are inherently class-imbalanced; (ii) node degrees vary considerably in real-world graphs; (iii) most existing works study these two issues separately but ignore the co-occurrence between class imbalance and topology imbalance in graphs. They overlook the fact that topology imbalance varies significantly across different relation types. Hence, we propose a novel model called **AD-GSMOTE** (**Ad**aptive **G**raph **SMOTE**) to tackle the class and topology issues simultaneously in multi-relation graphs. Specifically, we first design an adaptive topology-aware node generator and an efficient triadic edge generator to enhance the graph structure under each relation type by generating synthetic nodes for all tail nodes in minority classes and creating rich connections among tail nodes and others. Then, the enhanced multi-relation graph is fed into a GNN encoder to get node embeddings. Afterward, a class-aware logit adjustment module is designed to adjust the pre-softmax logit during model training, which enables the model to learn larger margins between minority and majority classes. To evaluate the performance of AD-GSMOTE, we build a new real-world graph (Twitter-Drug) to classify user roles in the drug trafficking community. The excellent performance on three real-world graphs demonstrates the effectiveness and efficiency of AD-GSMOTE compared with state-of-the-art methods. Source code and dataset are available at https://github.com/graphprojects/AD-GSMOTE.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WSDM '25, March 10–14, 2025, Hannover, Germany.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1329-3/25/03
https://doi.org/10.1145/3701551.3703553

KEYWORDS

Class imbalance, Topology imbalance, Graph neural network, Multi-relation Graph

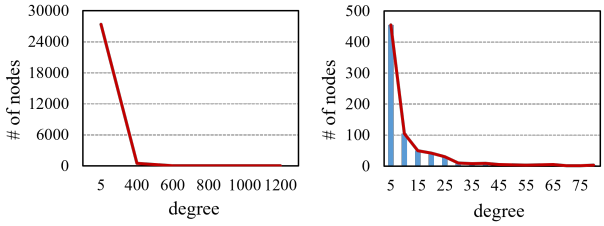
ACM Reference Format:
Yiyue Qian[†,‡], Tianyi Ma[‡], Chuxu Zhang, and Yanfang Ye*. 2025. Adaptive Graph Enhancement for Imbalanced Multi-relation Graph Learning. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining (WSDM '25), March 10–14, 2025, Hannover, Germany.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3701551.3703553

1 INTRODUCTION

In recent years, with the extraordinary performance in modeling relational data, graph representation learning has attracted considerable attention in various domains, such as social networks, molecule networks, commercial networks, etc. [25–27, 41–43, 46, 47, 49]. For instance, a growing number of works employ graph learning models to detect anomalous activities (a.k.a. anomaly detection), such as drug trafficking detection [18, 24, 48], malware detection [29–31], and bot account detection [52]. Graph Neural Networks (GNNs), as the mainstream graph representation learning method, has demonstrated effectiveness in learning graph representations over most well-built benchmark data sets [13, 39, 50].

However, most GNNs still have limitations in dealing with real-world graph datasets in the following aspects: (i) Nodes are inherently class-imbalanced in many real-world graphs. Some classes have significantly limited nodes for model training, which are called minority classes. For instance, most users on social networks are benign users, while only an extremely small proportion of users are abnormal users (e.g. drug traffickers and fake reviewers). In this case, GNNs would overclassify the majority classes but underclassify the minority classes, limiting their effectiveness for many real-world applications. (ii) Node degrees vary considerably across real-world graphs. For many real-world graph datasets, the node degrees approximately follow the power law distribution [16], as illustrated in Figure 1. (a) on Twitter-Drug data. The degree imbalance (topology imbalance) is another inherent problem in most real-world graph data. However, the performance of most GNNs depends heavily on the graph structure, which directly causes the

[†]The work is not related to the position at Amazon.
[‡]Equally contributed.
*Corresponding author.



(a). Degree distribution of all nodes. (b). Degree distribution of misclassified nodes via GCN.

Figure 1: Node degree distribution over Twitter-Drug data.

problem that the embeddings of low-degree (tail) nodes are unsatisfactory due to very scarce connections with others. (iii) Although several recent works [4, 36] have been proposed to handle both issues, they mainly focus on homogeneous graphs. In multi-relation graphs, the extent of topology imbalance varies significantly across different relation types. For instance, nodes 1 and 5 under relation 1 in Figure 2 have very scarce connections, and they are considered as tail nodes. However, nodes 1 and 5 are not regarded as tail nodes under relation r. Instead, node 8 is considered a tail node due to limited connections with other nodes. Therefore, the aforementioned limitations inspire us to investigate the following research problem: *How do we design an effective and efficient graph representation learning framework to simultaneously handle the class imbalance and the topology imbalance problems in real-world graphs?*

Generic works against the class imbalance problem can be categorized into three streams, i.e., post-hoc correction [10, 38], loss modification [9, 35], and re-sampling [2, 22]. However, directly applying these methods to graph data would cause inevitable problems as neighbor nodes are not involved in learning node representations. For instance, GraphSMOTE [51]) extends the classic interpolation method SMOTE [2] to graph by designing a re-sampling module and an edge predictor to synthesize nodes and edges for minority classes. However, these methods overlook the inherent topology imbalance problem in graphs.

In consideration of the influence of topology imbalance on GNNs, some works [37, 44] have studied the topology imbalance issue by employing the degree-specific transformations on nodes to improve the model performance. For instance, a recent work, Tail-GNN [16], proposes to learn a neighbor translation from the high-degree (head) nodes and further transfer the neighbor translation to the tail nodes. Besides, some recent works [4, 36] realize the co-occurrence between class imbalance and topology imbalance and have started to explore some methods to handle both imbalance issues in homogeneous graphs. A recent work called TOPOAUC [4] designs a unified topology-aware AUC optimization framework to handle both imbalance issues in homogeneous graphs. However, these works fail to handle the real-world multi-relation graphs and the designed methods for homogeneous graphs cannot be directly applied to multi-relation graphs.

To tackle the above challenges, we develop a novel model called AD-GSMOTE (as illustrated in Figure 2) that not only targets narrowing the structure gap (topology imbalance) between tail nodes and head nodes but also tries to enlarge the representation margins between the minority and majority classes (class imbalance). From Figure 1. (b) that illustrates the degree distribution of misclassified nodes in minority classes via GCN [13] on Twitter-Drug dataset, we find out that most misclassified nodes are tail nodes while most head nodes gain relatively excellent performance. Motivated by the findings, instead of indiscriminately generating synthetic nodes for all nodes in minority classes, we design a topology-aware adaptive node generator to generate synthetic nodes for tail nodes in minority classes and propose an efficient triadic edge generator to create connections for synthetic nodes and tail nodes. By this means, the graph is enhanced with more nodes to reduce the size gaps between minority and majority classes, and richer connections among tail nodes and others to narrow the structure gaps between tail nodes and head nodes, simultaneously. As this work studies the imbalance issues in multi-relation graphs, each enhanced graph with the specific relation is fed to the GNN encoder to get the node embeddings. Afterwards, we fuse node embeddings under different relation types via a semantic (relation) attention module and further design a class-aware logit adjustment module to adjust the pre-softmax logit during model training, enabling the model to learn a relatively large margin between the minority and majority classes. To conclude, the major contributions of this work are:

- *Novelty:* We design a novel model called AD-GSMOTE including the topology-aware graph augmentation and the class-aware logit adjustment to simultaneously handle class and topology imbalance problems in multi-relation graphs. To the best of our knowledge, AD-GSMOTE is the first to work against both imbalance problems in multi-relation graphs.
- *New Collected Data:* To evaluate AD-GSMOTE well, we collect a new real-world dataset (Twitter-Drug) w.r.t. the user roles classification in the drug trafficking community. Twitter-Drug will be public upon acceptance, which will contribute to the research communities of graph learning and fraud detection.
- *Effectiveness and efficiency:* We conduct comprehensive experiments on three real-world graphs (including Twitter-Drug). Results demonstrate the effectiveness and efficiency of AD-GSMOTE compared with state-of-the-art imbalanced graph methods.

2 RELATED WORK

Graph Neural Networks. Graph neural networks (GNNs), considering both the node features and the graph structure, have become the state-of-the-art approach to learning graph representations. Existing GNNs can be generally divided into two streams, spectral-based GNNs, and spatial-based GNNs. Spectral-based GNNs [1, 8] aim to present nodes in graphs and perform convolution in the spectral space. For instance, Henaff et al. extended the spectral networks to learn the node embedding in graph [8]. Spatial-based GNNs [7, 21] usually consider the relational structure information between nodes and aggregate the information of nodes based on the local structural information. For example, GCN [13] implements layer-wise propagation rule to learn the node embedding. R-GCN [34] is designed as a relational graph neural network for multi-relation graphs, which aims to learn the node embeddings under different relationships with different weight matrices. However, the problems that nodes in minority classes get biased representations and the node degrees vary considerably have been marginalized in most GNN models.

Imbalanced Graph Representation Learning. Existing works about imbalanced graph representation learning can be divided

into two categories, class-imbalanced graph representation learning [28, 51], and topology-imbalanced graph representation learning [16, 44]. Generic works against the class imbalance problem can be roughly divided into three streams, i.e., post-hoc correction [10, 38], loss modification [9, 35], and re-sampling [2, 22]. SMOTE [2], synthesizing minority samples with the closest sample, is one of the widely used re-sampling strategies. Recently, GraphSMOTE [51] extended SMOTE to the graph data by designing a re-sampling module and an edge predictor to determine the connectivity between synthetic nodes and other nodes in graphs. Besides, PC-GNN [15] devises a pick-and-choose strategy to handle the class imbalance issue in multi-relation graphs. All these works handle the class imbalance problem, but they ignore the fact that some minority nodes have scarce connections, which causes a bottleneck in graph representation learning.

As tails nodes with scarce connection always obtain unsatisfactory performance, especially in some scenarios when only the graph structure is provided without any attribute features, some recent works [16, 44] realize the importance of topology imbalance in graph representation learning. For instance, Demo-Net [44] treats nodes of different degrees differently based on the degree-specific transformation during model training. Tail-GNN [16] proposes to learn a neighbor translation from head nodes and further transfer the neighbor translation to tail nodes on homogeneous graphs. In recent, some works have realized the co-occurrence of class and topology imbalance issues and proposed some methods to handle both issues in homogeneous graphs. TOPOAUC [4] proposes a unified topology-aware AUC optimization framework to deal with both imbalance issues and TAM [36] proposes to compare the connectivity pattern of the nodes with the class-averaged counterpart and adaptively adjusts the margin accordingly. However, these works mainly focus on handling the imbalance issues on homogeneous graphs and most of them can not be directly extended to multi-relation graphs in real-world scenarios.

3 PRELIMINARY

Definition 3.1. **Multi-relation Class-Imbalanced Graph.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{X}, \mathcal{C})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes, $\mathcal{E} = \{E_1, \ldots, E_R\}$ is the edge set of different relation types, $\mathcal{A} = \{A_1, \ldots, A_R\}$ is the set of adjacency matrices with different relation types, $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ is the attribute feature set, and $\mathcal{C} = \{C_1, \ldots, C_C\}$ is the set of node classes. If $R > 1$, then $\mathcal{G}$ is a multi-relation graph. The class imbalance ratio CIR = $\frac{|\mathcal{V}_{C_1}|}{|\mathcal{V}_{C_2}|}$, where C_1 and C_2 denote any two classes in $\mathcal{C}$, $\mathcal{V}_{C_1}$ and $\mathcal{V}_{C_2}$ denote the node sets that belong to label C_1 and C_2, respectively. If CIR > 1, C_1 is called the majority class, and C_2 is called the minority class. Otherwise, C_1 is called the minority class, and C_2 is called the majority class. Particularly, $\forall(C_i, C_j) \in \mathcal{C}$, if CIR$\approx$1, $\mathcal{G}$ is a class-balanced multi-relation graph. Otherwise, $\mathcal{G}$ is a class-imbalanced graph.

Definition 3.2. **Topology-Imbalanced Graph.** Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{X}, \mathcal{C})$, $\mathcal{N}_i$ is the neighbor set of node v_i and $|\mathcal{N}_i|$ is the degree of node v_i. For a certain threshold of K, $\mathcal{V}_{\text{tail}}$ denotes the tail node set where $\mathcal{V}_{\text{tail}} = \{v_i : |\mathcal{N}_v| < K\}$, while $\mathcal{V}_{\text{head}}$ denotes the head node set where $\mathcal{V}_{\text{head}} = \{v_i : |\mathcal{N}_i| \geq K\}$. Based on the definition, $\mathcal{V}_{\text{tail}} \cup \mathcal{V}_{\text{head}} = \mathcal{V}$ and $\mathcal{V}_{\text{tail}} \cap \mathcal{V}_{\text{head}} = \emptyset$. In this work, as we propose to handle both the topology imbalance and the class imbalance issues simultaneously, inspired by [14], we define the topology-imbalance ratio (TIR) as TIR = $\frac{1/|\mathcal{V}_{C_1}| \cdot \sum_{v_i \in \mathcal{V}_{C_1}} |\mathcal{N}_i|}{1/|\mathcal{V}_{C_2}| \cdot \sum_{v_j \in \mathcal{V}_{C_2}} |\mathcal{N}_j|}$. If TIR$\approx$1 for all class pairs, the graph $\mathcal{G}$ represents a topology-balanced graph. Otherwise, $\mathcal{G}$ represents a topology-imbalanced graph. For a topology-imbalanced graph, we employ the degree threshold K (e.g., $K = 10$) to divide nodes in minority classes into $\mathcal{V}_{\text{tail}}$ and $\mathcal{V}_{\text{head}}$.

Definition 3.3. **Graph Neural Networks (GNNs).** Most GNNs [13, 39] follow the neighbor aggregation operation in the messaging passing framework. Specifically, each node receives and aggregates the messages from the neighbor nodes recursively in multiple layers, and GNN can be formulated as follows:

$$\mathbf{H}_i^{l+1} = \mathcal{M}\left(\mathbf{H}_i^l, \left\{\mathbf{H}_{v_j}^l : v_j \in \mathcal{N}_i\right\}; \theta^{l+1}\right), \tag{1}$$

where $\mathbf{H}_i^{l+1}$ is the representation of node v_i at $l+1$ layer and $\mathbf{H}_i^0 = \mathbf{X}_i$ is the original attribute feature. $\mathcal{M}(\cdot)$ is the message passing function for neighbor aggregation at layer $l+1$ with parameters θ. Note that AD-GSMOTE is applicable to any GNN backbone.

Problem 1. ***Graph Representation Learning on Imbalanced Multi-relation Graphs.*** *Given a real-world multi-relation graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{X}, \mathcal{C})$ that is class and topology imbalanced, we build a graph representation learning model $f_\phi : \mathcal{V} \to \mathbb{R}^d$ (with parameter ϕ) to project each node $v_i \in \mathcal{V}$ to a d-dimensional embedding, which emphasizes on enlarging the margin between minority and majority classes, and narrowing the structure gap between $\mathcal{V}_{tail}$ and $\mathcal{V}_{head}$.*

4 METHODOLOGY

In this section, we present the details of AD-GSMOTE (shown in Figure 2), including (1) topology-aware graph enhancement on multi-relation graphs and (2) class-aware logit adjustment for classification task.

4.1 Topology-aware Graph Augmentation

Classic interpolation methods (e.g., SMOTE [2]) hinge on generating synthetic samples via interpolation between the minority samples and similar samples. In recent, some works [23, 32, 51] have attempted to apply these classic interpolation methods to generate synthetic nodes in graphs. One of the classic methods, GraphSMOTE [51], extends SMOTE over homogeneous graphs to generate synthetic nodes for minority classes based on the closest neighbors. Yet, GraphSMOTE still has several limitations: (i) The proportion of information (e.g., attribute feature) that the synthetic nodes learn from other nodes is decided by random values (e.g., random δ in GraphSMOTE) following the certain distribution (e.g., uniform distribution). However, this stochastic approach may fail to capture useful information from other nodes, ultimately limiting the quality of synthetic nodes. (ii) GraphSMOTE marginalizes the topology imbalance in graph learning, which causes a bottleneck for GNN in real-world graphs. As shown in Figure 1. (b), most tail nodes in minority classes gain poorer performance compared with head nodes. (iii) The edge predictor in GraphSMOTE is not efficient as it utilizes node embeddings to reconstruct the adjacency matrix. In light of this, we design a topology-aware adaptive node generator and an efficient triadic edge generator to tackle the above issues.

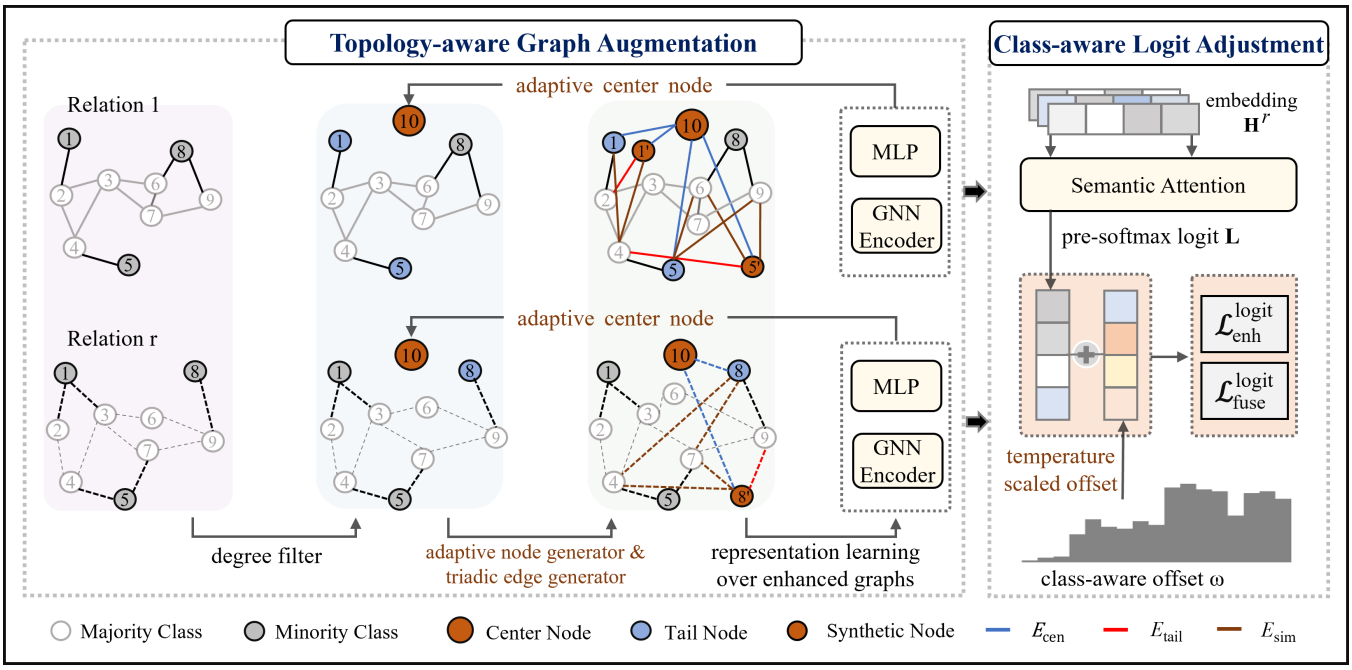


Figure 2: The overall framework of AD-GSMOTE: (a) It first leverages the degree filter to select tail (blue) nodes for minority classes (black nodes). Then, it generates the adaptive center node (brown node 10) based on the nodes that were correctly classified in the previous training process. It then designs the adaptive node generator to generate synthetic nodes (brown) for tail nodes and the triadic edge generator to create connections (i.e., E_{cen}, E_{tail}, and E_{sim}) for tail and synthetic nodes. Afterward, the enhanced graph under each relation r will be fed to a GNN-based classifier to obtain the classification results for generating the next-round adaptive center node dynamically; (b) It obtains the fused node embeddings with different relations via a semantic attention module and further designs a class-aware adjustment module to adjust the pre-softmax logits L via a temperature-scaled class-aware offset during model training. All parameters would be updated via optimizing $\mathcal{L}_{\text{enh}}^{\text{logit}}$ and $\mathcal{L}_{\text{fuse}}^{\text{logit}}$.

4.1.1 Adaptive Node Generator in Multi-relation Graphs. Existing works [16, 17, 44] find out that the node degrees vary considerably and approximately follow the power-law distribution in most real-world graphs. Generally speaking, the embeddings of tail nodes often cannot be accurately learned due to the scarce connections in graphs. As the node connections may vary significantly across different relation types in multi-relation graphs, it is ineffective to implement these methods designed for homogeneous graphs by simply converting the real-world multi-relation graphs to homogeneous graphs. For instance, Nodes 1 and 5 under relation 1 in Figure 2 are tail nodes while they are head nodes in relation r, showing that they have relatively sufficient connections under relation r while scarce connections under relation 1. In this case, Nodes 1 and 5 under relation 1 will gain insufficient information while they will receive relatively sufficient information during information propagation under relation r. If we simply leverage the imbalance method designed for homogeneous graphs, they are unable to tell the connection varies under different relationships, ultimately affecting the quality of the enhanced graph structure. Considering this, instead of indiscriminately generating synthetic nodes for all minority nodes in most works [32, 51], we emphasize improving the representation of tail nodes by adaptively generating synthetic nodes for tail nodes in minority classes under each relation type.

Specifically, we first divide the nodes in each minority class into two sets (i.e., $\mathcal{V}_{\text{tail}}^r$ and $\mathcal{V}_{\text{head}}^r$) via a degree threshold K, where r represents the relation type. Mention that, $\mathcal{V}_{\text{tail}}^r$ and $\mathcal{V}_{\text{head}}^r$ are different with different relation type r. For each node $v_{\text{tail}\_i}^r \in \mathcal{V}_{\text{tail}}^r$, unlike existing works (e.g., GraphSMOTE) that directly find the closest neighbor based on the feature similarities, we design an adaptive center node $v_{\text{cen}\_m}^r$ (brown center node 10 in Figure 2) for each minority class C_m to facilitate the generation of synthetic nodes. Particularly, we select nodes from which our model made accurate decisions during the previous training process of the downstream task. These correctly classified nodes serve as representatives of the minority class C_m. Here we use $\mathcal{V}_{\text{center}} = \left\{v_{\text{cen}\_1}^1, \ldots, v_{\text{cen}\_1}^R, \ldots, v_{\text{cen}\_m}^1, \ldots, v_{\text{cen}\_m}^R\right\}$ to denote the adaptive center node set, where m is the number of minority classes and R is the number of relation types. Afterward, we design an adaptive node generator that our synthetic nodes can consistently learn from both the adaptive central representative nodes and the most similar nodes within the same class, simultaneously. Based on this, we formally define the attribute feature of center node $v_{\text{cen}\_m}^r$ for relation type r in minority class C_m and the most similar node v_{sim_i} w.r.t. tail node v_{tail_i} as:

$$\begin{aligned} \mathbf{X}_{\text{cen}\_m}^r &= \text{mean}(\mathbf{X}_i \mid \mathbf{Y}_i = C_m \text{ and } v_i \in \mathcal{V}_{\text{correct}\_m}^r), \\ \mathbf{X}_{\text{sim}\_i} &= \mathbf{X}_j, \text{ where } j = \text{argmin}_k \left\| \mathbf{X}_{\text{tail}\_i} - \mathbf{X}_k \right\|. \end{aligned} \tag{2}$$

Here $\mathbf{Y}_{\text{tail}\_i} = \mathbf{Y}_k$. $\mathbf{X}_{\text{cen}\_m}^r$ is the feature vector of center node $v_{\text{cen}\_m}^r$, $\mathcal{V}_{\text{correct}\_m}^r$ is the correctly classified node sets of the minority class

C_m under relationship r obtained from the previous training process. Note that, for the first training process, we simply take an average over feature vectors of all nodes in C_m as $\mathbf{X}^r_{\text{cen\_}m}$. Besides, $\mathbf{X}_{\text{sim\_}i}$ represents the feature vector of the most similar node $v_{\text{sim\_}i}$ w.r.t. $v_{\text{tail\_}i}$, measured by Euclidean distance in the attribute feature space. Next, for each tail node $v^r_{\text{tail\_}i}$, after obtaining the center node $v^r_{\text{cen\_}m}$ and the most similar node $v_{\text{sim\_}i}$, the synthetic node $v^r_{\text{syn\_}i}$ can be generated via:

$$\begin{aligned}\mathbf{X}^r_{\text{syn\_}i} = \mathbf{X}_{\text{tail\_}i} &+ W_{\text{cen}} \odot (\mathbf{X}_{\text{tail\_}i} - \mathbf{X}^r_{\text{cen\_}m}) \\ &+ W_{\text{sim}} \odot (\mathbf{X}_{\text{tail\_}i} - \mathbf{X}_{\text{sim\_}i}),\end{aligned} \tag{3}$$

where $\mathbf{X}^r_{\text{syn\_}i}$ is the attribute feature of node $v^r_{\text{syn\_}i}$, $v^r_{\text{syn\_}i}$ belongs to the same minority class C_m of $v^r_{\text{tail\_}i}$, $v^r_{\text{cen\_}m}$, and $v_{\text{sim\_}i}$. $\odot$ is the Hadamard product, $W_{\text{cen}} \in \mathbb{R}^{1\times 1}$ is the learnable index to model the proportion of information that $v_{\text{tail\_}i}$ learns from $v^r_{\text{cen\_}m}$, and $W_s \in \mathbb{R}^{1\times 1}$ is another learnable weight index to learn the percentage of information that $v_{\text{tail\_}i}$ gains from $v_{\text{sim\_}i}$. Note that, the attribute feature $\mathbf{X}_{\text{tail\_}i}$ and $\mathbf{X}_{\text{sim\_}i}$ are fixed, while the feature $\mathbf{X}^r_{\text{cen\_}m}$ of the center node $v^r_{\text{cen\_}m}$ for minority class C_m is adaptively generated based on $\mathcal{V}^r_{\text{correct\_}m}$ obtained from the previous training process. Here we use $\mathcal{V}_{\text{syn}}$ to denote the synthetic nodes set. Therefore, the node set including all original nodes and synthetic nodes can be represented as $\mathcal{V}_{\text{enh}} = \mathcal{V} \cup \mathcal{V}_{\text{syn}}$ and the attribute features set can be denoted as $\mathcal{X}_{\text{enh}} = \mathcal{X} \cup \mathcal{X}_{\text{syn}}$. Mention that, the number of synthetic nodes $\mathcal{V}^r_{\text{syn\_}m}$ hinges on the over-sampling scale $\beta = \frac{|\mathcal{V}_c|}{|C|\cdot|\mathcal{V}^r_{\text{tail\_}m}|}\cdot\gamma$, where γ denotes the hyper-parameter of the control factor, $|\mathcal{V}_c|$ is the number of all labeled nodes, $|C|$ is the number of classes. Therefore, the number of synthetic nodes in minority class C_m under relation r is calculated as $|\mathcal{V}^r_{\text{syn\_}m}| = |\mathcal{V}^r_{\text{tail\_}m}| \cdot \beta$.

4.1.2 Triadic Edge Generator in Multi-relation Graphs. Up to now, the synthetic nodes are generated in an adaptive manner by constantly learning valuable information from the adaptive center nodes and the most similar nodes. However, these nodes are still isolated from the original graph $\mathcal{G}$, and the problem that tail nodes have scarce connections has not been addressed yet. To tackle these issues, some works create new connections among nodes by training an edge generator based on the node embeddings, which needs a lot of resources to train a qualified edge generator. For instance, GraphSMOTE utilizes node embeddings to reconstruct the adjacency matrix and further train the edge generator based on existing edges. However, this strategy is not efficient enough for real-world large datasets. In view of this, we devise a straightforward but effective triadic edge generator in multi-relation graphs to create connections based on the following strategies:

(i) Connecting Center Node: For each tail node $v^r_{\text{tail\_}i} \in \mathcal{V}^r_{\text{tail}}$ and each synthetic node $v^r_{\text{syn\_}i} \in \mathcal{V}^r_{\text{syn}}$ under relation r, we generate an edge connecting the tail/synthetic node with the corresponding center node $v^r_{\text{cen\_}m}$ (blue edges in Figure 2) so that the message of the center node can be propagated to the tail/synthetic nodes during graph model training. Based on this, the generated edges set for tail and synthetic nodes is denoted as:

$$E^r_{\text{cen}} = \left\{(v^r_i, v^r_{\text{cen}}) \mid v^r_i \in \left\{\mathcal{V}^r_{\text{tail}} \cup \mathcal{V}^r_{\text{syn}}\right\}\right\}. \tag{4}$$

(ii) Inheriting Neighbors of Tail Node: As each synthetic node $v^r_{\text{syn\_}i}$ is derived from the tail node $v^r_{\text{tail\_}i}$, we believe that node $v^r_{\text{syn\_}i}$ should inherit the direct connections (i.e., first-hop neighbors) of node $v^r_{\text{tail\_}i}$ (red edges in Figure 2) such that the synthetic node can receive the message from first-hop neighbors of the tail node during the messaging propagation. The generated edge set based on this strategy is denoted as:

$$E^r_{\text{tail}} = \left\{(v^r_{\text{syn\_}i}, v^r_j) \mid v^r_{\text{syn\_}i} \in \mathcal{V}^r_{\text{syn}} \text{ and } (v^r_{\text{tail\_}i}, v^r_j) = 1\right\}. \tag{5}$$

(iii) Absorbing Neighbors of Similar Node: Besides the learned information from the most similar node $v_{\text{sim\_}i}$, the tail node $v^r_{\text{tail\_}i}$ and the synthetic node $v^r_{\text{syn\_}i}$ also absorb the first-hop neighbors from node $v_{\text{sim\_}i}$. Here we can consider node $v_{\text{sim\_}i}$ as the "sibling" of node $v^r_{\text{tail\_}i}$ and $v^r_{\text{syn\_}i}$. Consequently, we force direct neighbors of the sibling to transmit messages to the tail node and the synthetic node during graph training. With this strategy, the generated edges set (brown edges in Figure 2) is denoted as:

$$E^r_{\text{sim}} = \left\{(v^r_i, v^r_j) \mid v^r_i \in \left\{\mathcal{V}^r_{\text{tail}} \cup \mathcal{V}^r_{\text{syn}}\right\} \text{ and } (v^r_{\text{sim\_}i}, v^r_j) = 1\right\}. \tag{6}$$

Following the aforementioned triadic strategies, the synthetic edge set is denoted as $\mathcal{E}_{\text{syn}} = \left\{E^1_{\text{cen}}, E^1_{\text{tail}}, E^1_{\text{sim}}, \ldots, E^R_{\text{cen}}, E^R_{\text{tail}}, E^R_{\text{sim}}\right\}$, where R is the number of relation types. The enhanced edge set can be denoted as $\mathcal{E}_{\text{enh}} = \mathcal{E} \cup \mathcal{E}_{\text{syn}}$. Accordingly, the new adjacency matrix should be $\mathcal{A}_{\text{enh}} = \left\{A^1_{\text{enh}}, \ldots, A^R_{\text{enh}}\right\}$. Hence, the enhanced multi-relation graph is denoted as $\mathcal{G}_{\text{enh}} = (\mathcal{V}_{\text{enh}}, \mathcal{E}_{\text{enh}}, \mathcal{A}_{\text{enh}}, \mathcal{X}_{\text{enh}}, C)$.

4.1.3 GNN Encoder. After obtaining each semantic-specific (relation r) enhanced graph $\mathcal{G}^r_{\text{enh}}$, we design a GNN-based classifier containing a GNN encoder and a fully connected layer to get the correctly classified nodes set $\mathcal{V}^r_{\text{correct\_}m}$. Here, the node embedding $\mathbf{H}^r$ in each semantic graph $\mathcal{G}^r_{\text{enh}}$ is formulated as:

$$\mathbf{H}^r = \widetilde{A}^r_{\text{enh}} \text{ReLU}(\widetilde{A}^r_{\text{enh}} \mathbf{X} W_0) W_1. \tag{7}$$

Here we take a two-layer GCN [13] as an example to learn the node embeddings. $\widetilde{A}^r_{\text{enh}}$ is the symmetric normalized adjacency matrix in $\mathcal{G}^r_{\text{enh}}$, $\mathbf{X}$ is the attribute feature, W_0 and W_1 are the weight matrices of the first layer and second layer in GNN encoder, respectively. We then feed $\mathbf{H}^r$ to a fully connected layer and further obtain the correctly classified node set $\mathcal{V}^r_{\text{correct\_}m}$ for minority class C_m under relation type r. Furthermore, we learn the adaptive center node $v^r_{\text{cen\_}m}$ via Eq. 2 for the next-round generation of synthetic nodes. Note that, the GNN-based classifier for each relation-specific enhanced graph shares the weight matrix. Besides, our triadic edge generator is straightforward and lightweight. With these advantages, AD-GSMOTE is efficient over large graphs. Details about efficiency comparison are discussed in Section 5.4.

4.2 Class-aware Logit Adjustment

4.2.1 Attention-based Semantic Fusion. Until now, we have obtained the node embeddings in each semantic graph. The semantic-specific node embeddings in different semantic (relation type) graphs reflect nodes from different perspectives. Therefore, we employ an attention module to fuse node embeddings and design a class-aware logit adjustment module to enhance our model. Specifically, for each node $v_i \in \mathcal{V}_{\text{enh}}$, the fused node embedding $\mathbf{Z}_i$ is given as

$\mathbf{Z}_i = \sum_{r=1}^{R} \alpha_i^r \cdot \mathbf{H}_i^r$, where $\alpha_i^r = \frac{\exp(w_i^r)}{\sum_{r=1}^{R} \exp(w_i^r)}$. Here r is the relation type and $\mathbf{H}_i^r$ is the node embedding generated via GNN. We follow this work [40] to learn the attention score w_i^r.

4.2.2 Class-aware Logit Adjustment. After embedding fusion, we propose a class-aware logit adjustment module to enlarge the margins between the logits of minority classes and majority classes. In general, logit adjustment is based on the label frequencies and would be applied either post-hoc on a trained model or as a modification of the training loss [5, 20, 45]. However, these methods (i.e., post-hoc on a trained model or modify the training loss) have some subtle limitations. For instance, existing work [20] proves that simple loss modification would sacrifice the consistency that underpins the softmax cross-entropy, and further cause the sub-optimal performance. To this end, we design a class-aware logit module to adjust the logits for both minority and majority nodes during model training. Specifically, for each node $v_i \in \mathcal{V}_{\text{enh}}$, we first apply the fused node embedding $\mathbf{Z}_i$ to a fully connected layer to get the pre-softmax logits $\mathbf{L}_i$. Instead of directly training our model with the softmax cross-entropy, we apply the label-dependent offset to calibrate the logits $\mathbf{L}_i$. By this means, the logit adjusted by the class-aware offset is aware of the real class distribution and further learns to mimic the distribution during model training. The logit adjusted softmax cross-entropy loss $\mathcal{L}_{\text{fuse\_}i}^{\text{logit}}$ for node v_i can be defined as:

$$\mathcal{L}_{\text{fuse\_}i}^{\text{logit}} = -\sum_{c=1}^{C} Y_{i,c} \cdot \log \frac{\exp \mathbf{L}_i^c + \tau_{\text{logit}} \cdot \log \omega_c}{\sum_{c' \in C} \exp \mathbf{L}_i^{c'} + \tau_{\text{logit}} \cdot \log \omega_{c'}}, \tag{8}$$

where $Y_{i,c} = 1$ if node v_i belongs to class c, and $\mathbf{L}_i^c$ is the logits of node v_i on class c. Here ω is the empirical class frequencies on training node sets, ω_c is the frequency of class c, and τ_{logit} is the hyper-parameter temperature index to calibrate the logits for learning the true class distribution in a better manner. In this way, our model is encouraged to learn a relatively large margin between the logits of minority and majority classes.

4.2.3 Overall Optimization Objective. By calibrating the logits, our loss $\mathcal{L}_{\text{fuse}}^{\text{logit}}$ is aware of the true class distribution. Besides, we also calibrate logits generated by the GNN-based classifier over each semantic-specific enhanced graph $\mathcal{G}_{\text{enh}}^r$. The optimization objective for each graph $\mathcal{G}_{\text{enh}}^r$ is formulated as:

$$\mathcal{L}_{\text{enh\_}i}^{\text{logit\_}r} = -\sum_{c=1}^{C} Y_{i,c} \cdot \log \frac{\exp \mathbf{L}_i^{r,c} + \tau_{\text{logit}} \cdot \log \omega_c}{\sum_{c' \in C} \exp \mathbf{L}_i^{r,c'} + \tau_{\text{logit}} \cdot \log \omega_{c'}}. \tag{9}$$

Similar to $\mathbf{L}_i^c$, we feed the semantic node embedding $\mathbf{H}_i^r$ under relation r, to a fully connected layer to get the pre-softmax logits $\mathbf{L}_i^{r,c}$. To conclude, the overall optimization objective is designed as:

$$\mathcal{L}_{\text{overall}} = \sum_{r=1}^{R} \sum_{v_i \in \mathcal{V}_{\text{enh}}^r} \mathcal{L}_{\text{enh\_}i}^{\text{logit\_}r} + \sum_{v_i \in \mathcal{V}_{\text{enh}}} \mathcal{L}_{\text{fuse\_}i}^{\text{logit}}, \tag{10}$$

where $\mathcal{V}_{\text{enh}}^r$ is the synthetic node set of the enhanced graph $\mathcal{G}_{\text{enh}}^r$ with updated edges. By minimizing $\mathcal{L}_{\text{overall}}$, all parameters in AD-GSMOTE will be updated.

5 EXPERIMENTS

In this section, we first introduce newly collected real-world dataset, Twitter-Drug, and two benchmark datasets. Afterward, we conduct comprehensive experiments to evaluate the effectiveness and efficiency of AD-GSMOTE. The data statistics is listed in Table 3.

5.1 Dataset

New Real-world Dataset. We collect a new real-world graph called **Twitter-Drug** to analyze the user roles (i.e., drug seller, drug user, drug discussant, and normal user) in drug trafficking communities on social media (i.e., Twitter). Twitter-Drug is a multi-relation graph having class imbalance and topology imbalance issues. Specifically, we obtain 25,046 normal users, 445 drug sellers, 1,932 drug users, and 522 drug discussants. The constructed multi-relation graph Twitter-Drug contains 27,945 nodes and 715,467 edges. Each node represents a user with 384-dimensional features. Three types of relations are introduced as follows: (i) U-T-U connects users when one replies/quotes/retweets/likes another user's tweets; (ii) U-F-U connects users if one follows the other; (iii) U-K-U connects users when text info (e.g., post) contain the same keyword.

Existing Benchmark Datasets. YelpChi graph dataset [33] contains Chicago hotel and restaurant reviews on Yelp. Nodes are reviews with 100-dimensional features, and graph includes three types of relations: (i) R-U-R for reviews by the same user; (ii) R-S-R for reviews of the same product with identical star ratings; (iii) R-T-R for reviews of the same product posted within the same month. **Amazon** graph dataset [19] includes product reviews in the Musical Instrument category. Nodes represent users with 100-dimensional features, and the graph has three types of relations: (i) U-P-U for users reviewing the same product; (ii) U-S-U for users sharing a star rating within a week; (iii) U-V-U for users with top-5% mutual TF-IDF review similarities.

5.2 Performance Comparsion

5.2.1 Baseline Methods. To evaluate AD-GSMOTE, we compare it with thirteen baseline methods which are divided into five groups: feature-based method (G1), graph representation learning models (G2), generic methods against class imbalance data (G3), graph models for addressing topology imbalance problem (G4), graph models against the class imbalance issue (G5). For G1, we feed the attribute feature to the SVM classifier [11]. For G2, we implement four classic graph learning methods: GCN [13], GAT [39], GraphSAGE [7], and R-GCN [34], to learn the node embeddings. For G3, we implement three generic popular methods against class imbalance: re-weighting [6], over-sampling [10], and SMOTE [2], to handle the node embeddings generated by the GNN encoder. For G4, we reproduce two popular models, i.e., Demo-Net [44] and Tail-GNN [16], for solving the topology imbalance issues on graphs. Besides, for G5, we reproduce three models against the class imbalance on graphs: GraphENS [23], GraphSMOTE [51], and PC-GNN [15].

5.2.2 Experimental Settings. All experiments are conducted under the Ubuntu 16.04 OS environment, with an Intel i9-9900k CPU and a Nvidia A40 Graphic Card. We train all methods with a fixed number of epochs (i.e., 500). Besides, all methods are trained 5 times, and the average performance on testing data is reported. We use

Table 1: Performance (mean ± sd) comparison among various methods on Twitter-Drug for user roles multi-class classification.

Setting		Twitter-Drug-5%		Twitter-Drug-10%		Twitter-Drug-20%		Twitter-Drug-40%	
Group	Model	Macro-F1	GMean	Macro-F1	GMean	Macro-F1	GMean	Macro-F1	GMean
G1	Feature+SVM [11]	51.38 ± 2.38	8.71 ± 7.45	55.64 ± 1.32	11.14 ± 9.65	58.83 ± 0.41	23.80 ± 3.83	60.92 ± 0.66	20.43 ± 10.6
G2	GCN [13]	57.72 ± 1.48	42.90 ± 4.72	59.32 ± 1.21	44.66 ± 2.37	60.46 ± 1.47	45.61 ± 2.64	62.68 ± 2.40	48.40 ± 3.75
	GAT [39]	56.45 ± 1.78	40.53 ± 5.21	58.45 ± 2.05	40.05 ± 4.20	60.54 ± 1.99	40.75 ± 1.78	61.36 ± 2.02	46.48 ± 2.35
	GraphSAGE [7]	53.37 ± 1.46	32.35 ± 5.70	56.63 ± 2.00	36.17 ± 3.50	59.54 ± 1.99	38.75 ± 2.42	60.73 ± 3.16	39.52 ± 2.04
	R-GCN [34]	52.99 ± 1.01	51.14 ± 4.33	54.17 ± 3.03	51.67 ± 4.02	56.15 ± 3.18	52.07 ± 3.36	56.38 ± 4.58	52.70 ± 2.19
G3	Re-weighting [6]	58.28 ± 1.72	43.40 ± 4.82	59.75 ± 1.00	45.19 ± 1.96	61.63 ± 1.88	47.35 ± 3.08	63.66 ± 2.40	48.95 ± 3.93
	Over-sampling [10]	58.10 ± 0.84	47.63 ± 3.21	61.01 ± 1.24	51.92 ± 1.82	61.26 ± 0.76	53.57 ± 1.42	62.35 ± 2.00	54.56 ± 2.41
	SMOTE [3]	58.83 ± 0.83	48.21 ± 3.66	61.38 ± 0.55	52.80 ± 1.44	61.57 ± 0.78	53.67 ± 1.34	63.78 ± 1.84	55.41 ± 2.47
G4	Demo-Net [44]	56.35 ± 1.98	40.02 ± 3.31	58.59 ± 1.89	45.85 ± 2.73	60.72 ± 0.59	47.82 ± 1.17	62.42 ± 1.07	50.08 ± 2.02
	Tail-GNN [16]	54.98 ± 0.51	45.24 ± 1.47	59.01 ± 0.93	52.09 ± 2.04	59.56 ± 0.58	53.11 ± 1.31	60.38 ± 1.12	54.34 ± 1.96
G5	GraphENS [23]	54.47 ± 0.55	52.06 ± 5.06	58.85 ± 0.88	57.56 ± 1.20	62.10 ± 0.39	61.62 ± 0.76	64.46 ± 0.28	63.70 ± 1.03
	GraphSMOTE [51]	60.42 ± 1.45	52.54 ± 3.83	63.45 ± 0.98	54.12 ± 1.75	65.51 ± 0.81	56.24 ± 1.47	66.57 ± 2.05	58.31 ± 2.37
	PC-GNN [15]	58.64 ± 1.24	51.51 ± 3.22	61.50 ± 1.44	52.37 ± 2.54	62.45 ± 1.13	53.54 ± 2.05	65.54 ± 2.14	54.45 ± 2.84
Ours	**AD-GSMOTE**	66.02 ± 0.45	56.78 ± 2.14	68.51 ± 0.67	60.07 ± 1.32	70.35 ± 1.24	62.05 ± 1.73	73.43 ± 1.83	65.59 ± 2.85

Table 2: Performance (mean ± sd) comparison among various methods on YelpChi and Amazon for review binary classification.

Setting		YelpChi-5%		YelpChi-40%		Amazon-5%		Amazon-40%	
Group	Model	Macro-F1	GMean	Macro-F1	GMean	Macro-F1	GMean	Macro-F1	GMean
G2	GCN [13]	46.08 ± 0.98	10.16 ± 2.34	47.20 ± 2.37	14.65 ± 6.74	48.35 ± 0.26	24.14 ± 1.96	53.42 ± 1.48	30.08 ± 3.92
	GAT [39]	48.04 ± 1.57	13.24 ± 3.24	48.57 ± 2.05	16.25 ± 5.24	49.45 ± 0.98	28.34 ± 1.57	55.24 ± 2.54	35.24 ± 5.21
	GraphSAGE [7]	49.59 ± 2.41	18.24 ± 7.20	49.88 ± 1.80	19.22 ± 5.68	60.15 ± 1.57	55.24 ± 3.42	63.24 ± 1.33	57.17 ± 4.21
	R-GCN [34]	50.92 ± 1.97	33.04 ± 7.80	52.13 ± 3.78	35.45 ± 8.78	57.44 ± 4.31	39.00 ± 7.39	60.09 ± 2.56	42.03 ± 9.46
G3	Re-weighting [6]	53.38 ± 2.82	36.71 ± 5.65	55.36 ±2.30	40.90 ± 2.38	65.04 ± 2.06	60.03 ± 5.87	67.12 ± 1.70	60.77± 4.31
	Over-sampling [10]	54.45 ± 3.24	37.15 ± 4.18	56.24 ± 3.24	41.21 ± 2.14	66.45 ± 3.04	62.04 ± 4.83	68.47 ± 3.01	62.47 ± 3.42
	SMOTE [2]	56.25 ± 3.14	39.41 ± 3.01	58.45 ± 2.78	43.24 ± 1.58	68.45 ± 3.41	65.57 ± 4.14	70.34 ± 2.77	66.51 ± 4.15
G4	Demo-Net [44]	50.53 ± 0.45	48.85 ± 0.78	50.90 ± 2.80	51.18 ± 0.59	67.63 ± 2.31	56.90 ± 3.07	74.35 ± 0.97	61.76 ± 1.15
	Tail-GNN [16]	59.73 ± 0.31	40.95 ± 0.50	61.34 ± 0.16	43.16 ± 0.34	87.54 ± 0.98	82.51 ± 1.59	89.11 ± 0.86	83.07 ± 1.52
G5	GraphENS [23]	58.18 ± 1.64	37.72 ± 5.13	61.71 ± 0.42	44.20 ± 0.22	85.98 ± 1.06	84.81 ± 2.24	86.60 ± 0.83	85.38 ± 2.49
	GraphSMOTE [51]	60.45 ± 2.13	53.45 ± 2.00	63.24 ± 1.58	54.42 ± 2.17	87.71 ± 2.02	84.35 ± 2.45	90.04 ± 2.23	86.91 ± 2.11
	PC-GNN [15]	65.83 ± 0.61	68.72 ± 4.80	66.95 ± 1.78	70.77 ± 1.59	85.45 ± 1.82	88.78 ± 0.49	86.61 ± 2.83	90.18 ± 0.99
Ours	**AD-GSMOTE**	66.93 ± 1.56	71.88 ± 0.51	71.34 ± 1.49	74.97 ± 0.31	90.46 ± 0.36	85.60 ± 0.78	91.50 ± 0.67	87.65 ± 0.69

Table 3: Statistics of three datasets. (CIR: class imbalance ratio, TIR: topology imbalance ratio).

Dataset	# of nodes	CIR	TIR	Relation	# of relation
Twitter-Drug	27,945	56.3 : 4.4 : 1.2 : 1.0	1.0 : 37.4 : 29.1 : 21.6	U-T-U	392,190
				U-F-U	69,675
				U-K-U	253,602
YelpChi	45,954	5.9 : 1.0	1.1 : 1.0	R-U-R	49,315
				R-S-R	3,402,743
				R-T-R	573,616
Amazon	11,944	13.5 : 1.0	2.9 : 1.0	U-P-U	175,608
				U-S-U	3,566,479
				U-V-U	1,036,737

5%/10%/20%/40% samples for training, 20% for validation, and the remaining data for testing. Following existing works in evaluating class imbalance and topology imbalance classification [15, 51], we adopt two metrics, i.e., Macro F1 score (**Macro-F1**) and GMean score (**GMean**) to evaluate all models. For all experiments, we utilize Adam [12] as the optimizer with learning rate and weight decay. For instance, the learning rate and weight decay over the Twitter-Drug dataset are 0.005 and 1e-4, respectively. With the grid search, hyper-parameters in AD-GSMOTE are different for different datasets. Let us take Twitter-Drug as an example. The degree threshold K is set as 10. In most cases, K is in the range of (5, 40). Besides, the control factor γ of the over-sampling scale is set as 0.7, and the temperature index τ_{logit} for calibrating the logits in Eq. 8 and Eq. 9 is set as 1.2.

5.2.3 Experimental Results. Table 1 shows the performances of all models over Twitter-Drug with different percentages (i.e., 5%, 10%, 20%, and 40%) of training data for user role classification. Purple-shaded numbers indicate the best results, and gray-shaded numbers represent the runner-up performance. According to Table 1, we conclude that: (i) Merely considering content (Feature vector) is not supportive enough to learn user representations. By comparison with G1 and G2, we find that rich relations among users increase the model performance to a large extent. (ii) Among three graph methods (i.e., GCN, GAT, and GraphSAGE) designed for homogeneous graphs, GCN has the best performance over Twitter-Drug. R-GCN outperforms the other three models in GMean but fails to outperform them in Macro-F1. (iii) In G3, we find that all generic models in G3 improve both the Macro-F1 and GMean scores to different degrees. SMOTE gains the best GMean performance in G3, demonstrating that model training benefits from generating synthetic nodes for minority classes. (iv) From G4, and G5, we conclude that merely considering either the topology imbalance (i.e., DEMO-Net and Tail-GNN) or the class imbalance (i.e., GraphSMOTE, GraphENS, and PC-GNN) within Twitter-Drug is

not effective enough. When considering both imbalance issues, AD-GSMOTE outperforms all models over three datasets, showing the superiority of our model in multi-class user role classification and review binary classification. Besides, Table 2 lists the comparison performances on YelpChi and Amazon. From this table, we can conclude that: (i) Generic methods against the class imbalance problem (G3), graph methods against topology (G4) or class imbalance (G5) improve the performance to some extent. PC-GNN, as the state-of-the-art fraud detection model against class imbalance problems in multi-relation graphs, shows its effectiveness. (ii) AD-GSMOTE outperforms all baseline models on the YelpChi dataset and achieves comparable performance over the Amazon dataset.

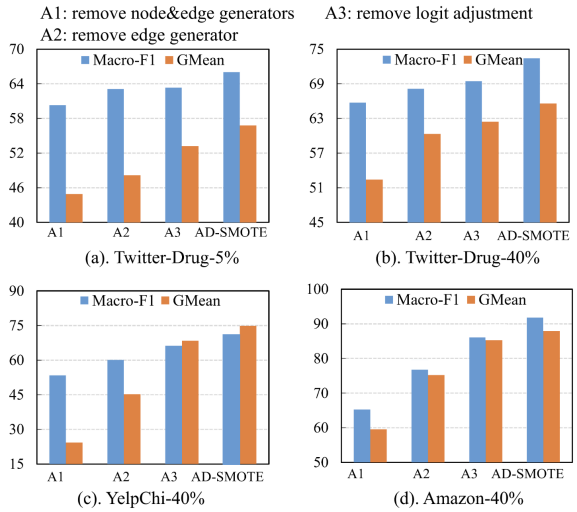


Figure 3: Performances of model variants over three datasets.

5.3 Ablation Study

To show the effectiveness of each component in AD-GSMOTE, we further analyze the contribution of each component, i.e., node and edge generators (A1), edge generator (A2), and logit adjustment (A3), by removing it separately. The results of model variants are shown in Figure 3. First, we remove node and edge generators from AD-GSMOTE (A1) and conclude our adaptive node generator and triadic edge generator have the biggest contribution to AD-GSMOTE. Afterward, we remove the edge generator (A2) and simply build connections for synthetic nodes by duplicating first-hop neighbors of the corresponding tail nodes. The performance decreases obviously. Lastly, we remove the class-aware logit adjustment module (A3) and find that the performance also decreases, showing the effectiveness of A3.

5.4 Efficiency Analysis

To discuss the efficiency of AD-GSMOTE, we conduct additional experiments on YelpChi data (the largest data) and further analyze the performance and running time among AD-GSMOTE and some baseline models in Table 4. The time and space complexity of AD-GSMOTE is proportional to $O(|\mathcal{V}|)$, which shows that our model incurs linear w.r.t the number of nodes. From Table 4, we find out that AD-SMOTE gains the best performance over YelpChi-40% with the least time than other models; Besides, some models are efficient and effective on small datasets (i.e., Amazon), but it is not efficient over large-scale data. For instance, the space complexity of GraphSMOTE is $O(|\mathcal{V}|^2)$, which is infeasible for large datasets. To conclude, Table 4 empirically demonstrates the efficiency of AD-GSMOTE.

Table 4: Performance and running time (minutes) comparison among models over YelpChi-40%.

Model	Tail-GNN	GraphSMOTE	PC-GNN	GraphENS+TAM	AD-GSMOTE
Marco-F1	61.46 ± 1.83	65.41 ± 1.97	67.02 ± 2.80	65.35 ± 3.15	71.34 ± 1.49
GMean	46.18 ± 3.19	59.45 ± 2.31	73.62 ± 1.93	58.53 ± 2.89	74.97 ± 0.31
Run Time	20.05 ± 9.45	148.37 ± 5.41	104.68 ± 1.06	20.34 ± 0.35	12.33 ± 0.60

5.5 Over-sampling Scale Analysis

We further discuss the influence of the control factor γ for the over-sampling scale β within AD-GSMOTE. We vary γ in the range of $\{0.3, 0.5, 0.7, 1.0, 1.3, 1.5\}$. Figure 4 shows the Macro-F1 of AD-GSMOTE with different over-sampling scales γ over three datasets. we observe that a suitable over-sampling scale generates high-quality synthetic nodes so that our model can gain better performance over a more balanced graph. If γ is too small or large, the number of synthetic nodes in minority classes would be insufficient or redundant for training, and further, the performance would not be enhanced.

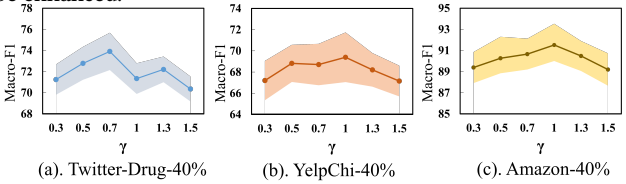


Figure 4: Macro-F1 performance of AD-GSMOTE with different control factors γ over three datasets.

6 CONCLUSION

In this paper, we design a novel model called AD-GSMOTE to handle class and topology imbalance issues in multi-relation graphs. Specifically, we build a topology-aware adaptive node generator and an efficient triadic edge generator to enhance the semantic graphs with different relation types. After fusing the node embeddings generated by the GNN encoder over the enhanced semantic graphs, we design a class-aware module to calibrate the logits during the model training to enlarge the margins between the minority and majority classes. To evaluate the performance of AD-GSMOTE, we first collect a new real-world dataset called Twitter-Drug to classify user roles in drug trafficking communities. Twitter-Drug will be public upon acceptance, which will contribute to the research communities in graph learning and fraud detection. Moreover, we conduct extensive experiments on three datasets including Twitter-Drug and two benchmark data to show the effectiveness and efficiency of AD-GSMOTE in handling real-world multi-relation graphs.

ACKNOWLEDGEMENTS

This work was partially supported by the NSF under grants IIS-2321504, IIS-2334193, IIS-2340346, IIS-2217239, CNS-2426514, CNS-2203261, and CMMI-2146076. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

REFERENCES

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* (2002).

[3] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. 2003. SMOTEBoost: Improving prediction of the minority class in boosting. In *PKDD*.

[4] Junyu Chen, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. A Unified Framework against Topology and Class Imbalance. In *MM*.

[5] Guillem Collell, Drazen Prelec, and Kaustubh Patil. 2016. Reviving threshold-moving: a simple plug-in bagging ensemble for binary and multiclass imbalanced data. *arXiv preprint arXiv:1606.08698* (2016).

[6] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. 2019. Class-balanced loss based on effective number of samples. In *CVPR*.

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*.

[8] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[9] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. 2019. Few-shot object detection via feature reweighting. In *ICCV*.

[10] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. 2020. Decoupling representation and classifier for long-tailed recognition. In *ICLR*.

[11] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural computation* (2001).

[12] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

[13] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[14] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In *NeurIPS*.

[15] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and choose: a GNN-based imbalanced learning approach for fraud detection. In *WWW*.

[16] Zemin Liu, Trung-Kien Nguyen, and Yuan Fang. 2021. Tail-gnn: Tail-node graph neural networks. In *SIGKDD*.

[17] Zemin Liu, Wentao Zhang, Yuan Fang, Xinming Zhang, and Steven CH Hoi. 2020. Towards locality-aware meta-learning of tail node embeddings on networks. In *CIKM*.

[18] Tianyi Ma, Yiyue Qian, Chuxu Zhang, and Yanfang Ye. 2023. Hypergraph Contrastive Learning for Drug Trafficking Community Detection. In *ICDM*. IEEE.

[19] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*.

[20] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. 2021. Long-tail learning via logit adjustment. In *ICLR*.

[21] Alessio Micheli. 2009. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks* (2009).

[22] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. 2020. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *ICICS*.

[23] Joonhyung Park, Jaeyun Song, and Eunho Yang. 2021. GraphENS: Neighbor-Aware Ego Network Synthesis for Class-Imbalanced Node Classification. In *ICLR*.

[24] Yiyue Qian. 2024. *Graph Representation Learning Techniques for the Combat Against Online Abusive Activity*. Ph. D. Dissertation. University of Notre Dame.

[25] Yiyue Qian, Philip Chen, Song Cui, and De Chen. 2023. Universal ring-of-abusers detection via multi-modal heterogeneous graph learning. (2023).

[26] Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2023. Adaptive Expansion for Hypergraph Learning. (2023).

[27] Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. 2024. Dual-level Hypergraph Contrastive Learning with Adaptive Temperature Enhancement. In *Companion Proceedings of the ACM on Web Conference 2024*. 859–862.

[28] Yiyue Qian, Chunhui Zhang, Yiming Zhang, Qianlong Wen, Yanfang Ye, and Chuxu Zhang. 2022. Co-Modality Graph Contrastive Learning for Imbalanced Node Classification. In *NeurIPS*.

[29] Yiyue Qian, Yiming Zhang, Nitesh Chawla, Yanfang Ye, and Chuxu Zhang. 2022. Malicious Repositories Detection with Adversarial Heterogeneous Graph Contrastive Learning. In *CIKM*.

[30] Yiyue Qian, Yiming Zhang, Qianlong Wen, Yanfang Ye, and Chuxu Zhang. 2022. Rep2Vec: Repository Embedding via Heterogeneous Graph Adversarial Contrastive Learning. In *SIGKDD*.

[31] Yiyue Qian, Yiming Zhang, Yanfang Ye, and Chuxu Zhang. 2021. Adapting Meta Knowledge with Heterogeneous Information Network for COVID-19 Themed Malicious Repository Detection. In *IJCAI*.

[32] Liang Qu, Huaisheng Zhu, Ruiqi Zheng, Yuhui Shi, and Hongzhi Yin. 2021. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *SIGKDD*.

[33] Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *SIGKDD*.

[34] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*.

[35] Zheyan Shen, Peng Cui, Tong Zhang, and Kun Kunag. 2020. Stable learning via sample reweighting. In *AAAI*.

[36] Jaeyun Song, Joonhyung Park, and Eunho Yang. 2022. TAM: topology-aware margin loss for class-imbalanced node classification. In *ICML*. PMLR.

[37] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and mitigating degree-related biases in graph convoltuional networks. In *CIKM*.

[38] Junjiao Tian, Yen-Cheng Liu, Nathaniel Glaser, Yen-Chang Hsu, and Zsolt Kira. 2020. Posterior re-calibration for imbalanced datasets. In *NeurIPS*.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.

[40] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019. Heterogeneous graph attention network. In *WWW*.

[41] Qianlong Wen, Zhongyu Ouyang, Chunhui Zhang, Yiyue Qian, Yanfang Ye, and Chuxu Zhang. 2022. Adversarial cross-view disentangled graph contrastive learning. *arXiv preprint arXiv:2209.07699* (2022).

[42] Qianlong Wen, Zhongyu Ouyang, Chunhui Zhang, Yiyue Qian, Chuxu Zhang, and Yanfang Ye. 2022. GCVR: Reconstruction from Cross-View Enable Sufficient and Robust Graph Contrastive Learning. In *UAI*.

[43] Qianlong Wen, Zhongyu Ouyang, Jianfei Zhang, Yiyue Qian, Yanfang Ye, and Chuxu Zhang. 2022. Disentangled dynamic heterogeneous graph learning for opioid overdose prediction. In *KDD*.

[44] Jun Wu, Jingrui He, and Jiejun Xu. 2019. Net: Degree-specific graph neural networks for node and graph classification. In *SIGKDD*.

[45] Han-Jia Ye, Hong-You Chen, De-Chuan Zhan, and Wei-Lun Chao. 2020. Identifying and compensating for feature deviation in imbalanced deep learning. *arXiv preprint arXiv:2001.01385* (2020).

[46] Yanfang Ye, Yujie Fan, Shifu Hou, Yiming Zhang, Yiyue Qian, Shiyu Sun, Qian Peng, Mingxuan Ju, Wei Song, and Kenneth Loparo. 2020. Community mitigation: A data-driven system for covid-19 risk assessment in a hierarchical manner. In *CIKM*.

[47] Yanfang Ye, Shifu Hou, Yujie Fan, Yiming Zhang, Yiyue Qian, Shiyu Sun, Qian Peng, Mingxuan Ju, Wei Song, and Kenneth Loparo. 2020. α-Satellite: An AI-Driven System and Benchmark Datasets for Dynamic COVID-19 Risk Assessment in the United States. *IEEE Journal of Biomedical and Health Informatics* (2020).

[48] Yiming Zhang, Yujie Fan, Wei Song, Shifu Hou, Yanfang Ye, Xin Li, Liang Zhao, Chuan Shi, Jiabin Wang, and Qi Xiong. 2019. Your style your identity: Leveraging writing and photography styles for drug trafficker identification in darknet markets over attributed heterogeneous information network. In *WWW*.

[49] Yiming Zhang, Yiyue Qian, Yujie Fan, Yanfang Ye, Xin Li, Qi Xiong, and Fudong Shao. 2020. dstyle-gan: Generative adversarial network based on writing and photography styles for drug identification in darknet markets. In *ACSAC*.

[50] Yiming Zhang, Yiyue Qian, Yanfang Ye, and Chuxu Zhang. 2022. Adapting Distilled Knowledge for Few-shot Relation Reasoning over Knowledge Graphs. In *SDM*.

[51] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *WSDM*.

[52] Yao Zhao, Yinglian Xie, Fang Yu, Qifa Ke, Yuan Yu, Yan Chen, and Eliot Gillum. 2009. Botgraph: large scale spamming botnet detection. In *NSDI*.