HetGSMOTE: A Heterogeneous Graph Oversampling Framework

A thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

by

ADHILSHA A

HBNI Enrollment No.: **PHYS11202013003**

to the

School of Physical Sciences

National Institute of Science Education and Research

Bhubaneswar

May, 2025

DEDICATION

To those who showed me the path ahead,
to those who lifted me when I bled,
to those who hold me close still near,
and to those who vanished through the years—
you all made me who I am,
and brought this dream to where it stands.

DECLARATION

I hereby declare that I am the sole author of this thesis in partial fulfillment of the requirements for a postgraduate degree from National Institute of Science Education and Research (NISER). I authorize NISER to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature of the Student

Date: May 22, 2025

The thesis work reported in the thesis entitled "**HetGSMOTE: A Heterogeneous Graph Oversampling Framework**" was carried out under my supervision, in the **School of Physical Sciences at NISER, Bhubaneswar, India**.

(Dr. Subhankar Mishra)
Signature of the thesis supervisor
School: School of Computer Sciences
Date: May 22, 2025

(Dr. Satyaprasad P Senanayak)
Signature of the thesis supervisor
School: School of Physical Sciences
Date: May 22, 2025

ACKNOWLEDGEMENTS

Let me begin by expressing my sincere gratitude to Dr. Subhankar Mishra, my supervisor, for his unwavering support and valuable insights throughout my research. I am equally grateful to my guide from SPS, Dr. Satyaprasad P Senanayak, as well as to the members of my thesis committee, Dr. Aritra Banik and Dr. Anup Kumar Bhattacharya, for their constructive feedback that greatly improved this work.

Throughout my academic journey at NISER, I have relied on the expertise and kindness of the faculty and staff from both the School of Physical Sciences and the School of Computer Sciences. They assisted me with their resources and knowledge to the best of their abilities, always with a smile that will never leave my heart.

I am deeply grateful to the members of my lab—especially my seniors Jyotirmaya, Jyothish, Rucha, Rishi, Pankaj, and Hari—for their consistent guidance and encouragement. I also extend my heartfelt thanks to my peers and colleagues Deependra, Aritra, Sagar, and Shithij. Their insightful discussions and steady support have been invaluable to the journey that brought both me and this thesis to where we are today.

Life, much like this world, has never promised to be fair. The past five years have been a rollercoaster of a ride, but through every twist and turn, I was never alone. To my parents, who always had my back; and to my brother, who looked up to me and gave me a reason to keep striving—I owe more than words can say. To my cousins and extended family, thank you for standing by me and for filling my life with unforgettable moments.

To Parvathi, my sister by heart, you have stood beside me for nearly a decade like a constant anchor to my chaotic soul. To Anagha and Gayathri, my closest friends with whom I've shared light laughs and tangled thoughts, you two have changed me for the better in ways I can never truly repay. To Archita, who quietly warmed my heart with a softer light, you have made my future worth dreaming of. To the Second floor buddies, especially Shubhay and Tanmay, you guys have given me memories that make NISER worth remembering.

I don't expect the path ahead to be any easier, but with all of you by my side and the lessons I have learned, I am ready to pursue my dream and beyond.

ABSTRACT

Graph Neural Networks (GNNs) have been widely used to learn from graph-structured data, with heterogeneous graphs (HetGs) gaining prominence due to their ability to represent diverse real-world systems. However, class imbalance—where some node classes are underrepresented—poses a significant challenge for learning tasks like node classification on HetGs. This work introduces HetGSMOTE, a novel oversampling method that adapts SMOTE-based techniques to the heterogeneous graph setting by incorporating node-type, edge-type, and contextual metapath information into the oversampling process. HetGSMOTE constructs a content- and neighbor-type-aggregated embedding space using a base model to generate synthetic nodes and trains edge generators for each node type to model relational structures, addressing key challenges in imbalanced learning on HetGs. Extensive experiments on benchmark datasets and across various base models demonstrate that HetGSMOTE consistently outperforms baseline methods in mitigating class imbalance and boosting classification accuracy, particularly in extreme imbalance cases, while maintaining broad adaptability to different base models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graph-based learning and Graph Neural Networks (GNNs) have garnered significant attention for their capacity to model intricate relationships and dependencies within structured data [2, 3]. Among the graph-based structures, heterogeneous graphs (HetG) are ubiquitous in real-world scenarios representing a wide range of real-life data. These include bibliographic networks, social networks, recommender systems [4, 5], etc. In HetGs, entities and relations are of multiple types highlighting complex relationships, with nodes carrying both structured and unstructured content. Consequently, downstream tasks like node classification [6, 7] and link prediction [8, 9] hold significant importance. While Heterogeneous Graph Neural Networks (HGNNs) [6] excel in node classification tasks, they are typically optimized for balanced class distributions. However, in many real-world applications, certain classes may have significantly fewer instances than others, leading to suboptimal performance when using traditional baseline models, calling for imbalanced learning techniques like oversampling on HetGs.

This work proposes HetGSMOTE, a novel oversampling approach in heterogeneous graphs, which leverages Synthetic Minority Oversampling Techniques (SMOTE) [10] on the representation matrices of the nodes to mitigate the class imbalance problem. SMOTE has proven effective in balancing class distributions in various domains, including homogeneous graphs [11], but its direct application to heterogeneous graph data presents unique challenges, such as requiring node-type, edge-type, and metapath-based contextual information. To overcome these challenges, HetGSMOTE

extends the GraphSMOTE [11] approach proposed for homogeneous graphs to heterogeneous graphs. It constructs a content-aggregated and neighbor-type-aggregated embedding space that encodes node similarities, facilitating the generation of synthetic samples that preserve the contextual relationships within the graph. Simultaneously, edge generators for different node types are trained to model the relational information between nodes, ensuring that the synthesized samples retain essential structural characteristics. This way, HetGSMOTE handles the first two of the three required criteria for SMOTE-based oversampling-:

1. Generation of synthetic samples for increasing the diversity and reducing inter-class imbalance.

2. Oversampling in the safe regions of the embedding space, i.e., source samples less prone to generating noisy samples [12].

3. Oversampling on difficult-to-classify samples to overcome the intra-class imbalance [13, 14].

After the aggregation step, the related nodes are in closer clusters than less related nodes due to having similar neighborhoods, thereby producing safer spaces for oversampling via SMOTE. The contributions are-:

1. Studying the class imbalance problem in heterogeneous graphs, which has wide real-life implications.

2. Extending the GraphSMOTE oversampling approach to the heterogeneous graph with our novel HetGSMOTE strategy.

3. Demonstrating the effectiveness of our approach through experiments conducted across diverse settings and datasets, showcasing its superior performance compared to baseline methods.

Chapter 2 defines and explains the background and terms related to the context of the paper. The related works are given in Chapter 3, and the Methodology is given in Chapter 4, which proposes the HetGSMOTE framework. The experiments with related details can be found in Chapter 5 followed by the results and discussions in Chapter 6. Chapter 7 summarizes the work with conclusions and future directions.

Chapter 2

Background

A *graph*, $G(V, E)$, is a mathematical structure that represents the relationship between objects. It is made up of *vertices* (or *nodes*), which denotes the individual objects, while *edges* are the connections that link pairs of vertices. Each graph's structure helps reveal patterns and insights about how components are connected. *Homogeneous graphs* consist of a single type of node and edge, such as social networks with only user nodes and friendship links. In contrast, *heterogeneous graphs* contain multiple types of nodes and/or edges, enabling the capture of richer semantic relationships—for example, bibliographic networks with papers, authors, and venues.

Graph Neural Networks (GNNs) have proven highly effective in analyzing graph-structured data, with applications spanning from social network analysis to molecular biology and recommendation systems. By leveraging the relational information between nodes, GNNs are capable of learning meaningful representations that capture both local and global graph properties.

General GNNs update the state (or embedding) of each node in a graph based on the states of its neighbors [1]. The following equation characterizes them:

$$h_v^{(t)} = \text{UPDATE}(h_v^{(t-1)}, \text{AGGREGATE}(\{\{h_u^{(t)} : u \epsilon N(v)\}\})) \tag{2.1}$$

The AGGREGATE function collects information from the neighbors, and the UPDATE function updates the node embedding using aggregated information. This information can be used for performing node-level, edge-level, or graph-level tasks, such as classification or regression. Examples include user-bot prediction, friendship recommendation, and protein property prediction [4].

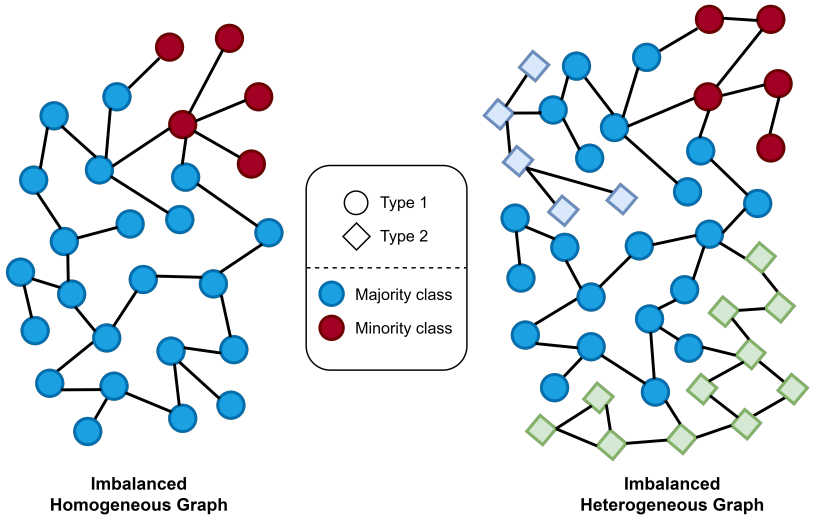

Figure 2.1: Illustration of homogeneous and heterogeneous graphs with node imbalance

In the context of node classification and graph datasets, a common challenge is *class imbalance*, where certain node classes are underrepresented. These underrepresented classes are referred to as the *minority class*. This imbalance can bias learning algorithms toward majority classes, reducing the performance on minority classes. The issue is particularly pronounced in heterogeneous graphs, where diverse structures and relations further complicate representation learning. Figure 2.1 illustrates homogeneous and heterogeneous graphs, with the presence of node imbalance.

To address class imbalance in node classification tasks, oversampling techniques are commonly employed to increase the representation of minority classes. In graph learning, *oversampling* refers to the generation or duplication of synthetic node representations from underrepresented classes to overcome the class imbalance during training. Unlike traditional settings, oversampling in graphs must consider not only the feature space but also the structural and relational context of each node, making the design of effective oversampling strategies particularly challenging in heterogeneous graphs where node types and relationships vary.

The *Synthetic Minority Over-sampling Technique (SMOTE)* [10] is one of the widely used methods among them. Rather than simply duplicating existing samples, SMOTE generates new synthetic samples by interpolating between the representation vectors of selected minority class samples. By using SMOTE in a feature space where content and neighborhood aggregation are accounted for, SMOTE can be adapted to Graph datasets. This is the intuition behind GraphSMOTE [11] and our work HetGSMOTE. A glossary of Machine Learning terms is given in Appendix A.

Chapter 3

Related Works

Class imbalance, where one class significantly outnumbers another, leads to biased models and poor generalization, evident in tasks such as fraud detection, rare disease identification, and bot recognition. Addressing this issue involves algorithm-level, data-level, and hybrid strategies [11]. Data-level methods, such as oversampling and data augmentation, increase minority class samples. Algorithm-level approaches include cost-sensitive techniques, ensemble learning, and threshold adjustments [16]. Hybrid methods combine these strategies, e.g., classifier-specific models [15].

Oversampling generates synthetic minority samples with methods like SMOTE, which interpolates between samples. Variants of SMOTE have evolved through various enhancements aimed at improving data representation. Some methods including DBSMOTE [27], and k-means SMOTE [14], focus on generating synthetic samples within the minority class space with smaller scales. Methods like borderline-SMOTE [12], ADASYN [13], and Adaptive-SMOTE [24] also generates samples in difficult regions within the minority class. These improvements help mitigate overgeneralization by filtering out potential noise or by strategically generating additional samples within specific regions of the minority class. A newer technique, NaNG-SMOTE [25], addresses the same obstacles by using a natural neighborhood graph and subgraph cores of the minority class to generate synthetic samples while filtering noise based on edge characteristics. Oversampling has proven effective in numerous machine learning domains, addressing the issue of limited minority data [15] and improving model performance.

HetG learning has evolved from manual feature engineering to representation learning, broadly categorized into shallow and deep models [17]. Shallow models include random walk-based strategies like metapath2vec, which use metapath-guided random walks, and decomposition methods such as HERec [5]. DeepWalk introduced SkipGram embeddings for node co-occurrence probabilities. Variants like Spacey, JUST, and HHNE [4] add enhancements like jump-stay strategies and heterogeneous walks. Deep models leverage Heterogeneous Graph Neural Networks (HGNNs), with unsupervised methods like HetGNN [6], and semi-supervised approaches like HAN [7] using attention mechanisms. Advanced HGNNs, like MAGNN and HGT [19], refine intra- and inter-metapath aggregations. Techniques such as encoder-decoder models [4] and adversarial frameworks like GraphGAN [20] further enrich HetG representation learning. Oversampling techniques for graph data, like GraphSMOTE [11], address the class imbalance in homogeneous graphs by generating synthetic nodes and connections. Other methods, such as GraphMixup [23] and Graph-DAO [22], use latent space sampling and semantic relations. Despite progress, SMOTE has found limited applications in graph-based learning. For HetGs, there are even fewer techniques, such as FincGAN [21], an adversarial GAN-based approach, and BARE[26], which leverages student-teacher networks to distill knowledge from real nodes for improved learning. This motivates our work on the extension of GraphSMOTE to heterogeneous graphs.

Chapter 4

Methodology

4.1 Problem Statement

Consider a Heterogeneous Graph (HetG), denoted as $\mathcal{G} = (V, A, F)$. Here, $V = \{v_1, \ldots, v_n\}$ is the node set, and $V_t \in V$ signifies its subset containing nodes of type t. A is the set of adjancency matrices, where $A_{tu} \in A$ denotes the adjancency matrix between nodes of types t and u. $A_{tu} \in \mathbb{R}^{n_t \times n_u}$, where n_t and n_u are the numbers of nodes of types t and u, respectively. Finally, F is the set of attribute matrices, where $F_i^t \in F$ represents the i-th attribute matrix for nodes of type t. $F_i^t \in \mathbb{R}^{n_t \times d}$, where d is the embedding dimension. Specifically, $F_i^t[v_j, :] \in \mathbb{R}^{1 \times d}$ is a row vector representing the i-th attribute embedding for the node v_j of type t. Class information for nodes of type t is denoted by $Y \in \mathbb{R}^{n_t}$. In most real-world scenarios, only a subset of Y_t may be available during training, denoted by $Y_t' \in Y_t$ for the labeled subset of nodes $V_t' \in V_t$. The subset is usually small. If there exist k classes, represented as $c_1,\ c_2, \ldots, c_k$ and $|c_i|$ is the number of samples associated with i-th class, we can measure the degree of class imbalance using the imbalance ratio (I), defined as:

$$(\mathrm{IR})_i = \frac{|c_i|}{\max_j(|c_j|)} \tag{4.1}$$

where $\max_j |c_j|$ is the size of the largest class. A perfectly balanced dataset, where each class has the same number of nodes, corresponds to $(\mathrm{IR})_i = 1$ for all classes. Conversely, lower $(\mathrm{IR})_i$ values indicate a higher class imbalance, with some classes having significantly fewer nodes than others.

In this study, a semi-supervised approach is employed to classify nodes on diverse

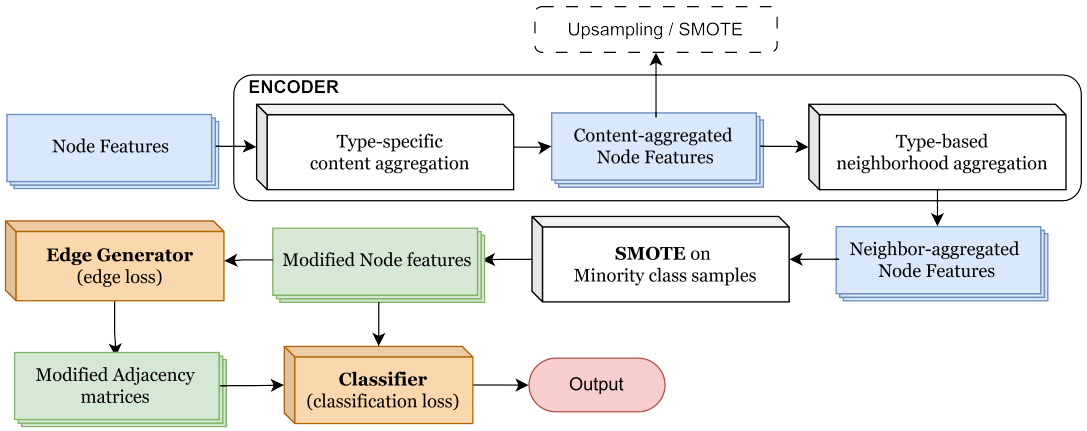

Figure 4.1: HetGSMOTE Pipeline

graphs in a transductive setting. The entire graph is available for training and testing, with only a few nodes labeled. These selected nodes are divided into different sets for training, validation, and testing during the learning process. Given the heterogeneous graph $\mathcal{G}$ with an imbalanced class distribution and a smaller subset of nodes V_t' with labels, our objective is to use the HetGSMOTE framework to generate synthetic nodes, V_t^{syn}, and synthetic edges,A_{tu}', till the class imbalance has been overcome. Hence, we will be able to train a node classifier f that performs well for both the majority and minority classes, that is, $f(\mathcal{G}) \rightarrow Y$.

4.2 HetGSMOTE framework

As shown in Fig. 4.1, the pipeline of HetGSMOTE consists of four parts-:

- Encoder
- SMOTE-based oversampling
- Edge Generators
- Classifier

4.2.1 Encoder

The Encoder is a feature extractor based on models like HGNN [6], MAGNN [19], and HAN [7] that operate on Heterogeneous Graphs. The encoder is used to generate learned embeddings for the different types of nodes of the heterogeneous graphs, followed by the content aggregation and type-specific neighbor aggregation, specific to the base model. The output of this layer is node feature vectors in the embedding space that are then used for oversampling via SMOTE.

Content Aggregation: In this layer, the attribute matrix for each node is concatenated along the embedding dimension and then passed through a linear layer and nonlinearity to reduce its dimension back to the original embedding dimension, thereby combining the contextual information from all generated attributes of the nodes.

$$F^t[v_j, :] = \sigma\left[W_1 \cdot \left(\bigoplus_{i=1}^{n} F_i^t[v_j, :]\right)\right] \tag{4.2}$$

where F_i^t is the i^{th} attribute matrix for the node v_j, n is the total number of attribute matrices, F^t is the content aggregated matrix, W_1 is the weight matrix, t the node type and σ refers to activation function like ReLU.

Type Based Neighbor Aggregator: Here, the content-based aggregated representation matrix for each node type is concatenated with the neighbor type-based aggregated feature matrix and then reduced using a linear layer. Firstly, a list of frequently occurring neighbor nodes along different random walks is extracted for each node. The top k frequently occurring neighbor nodes of each type are taken from the random walks and given the same importance as direct edges, giving a total of t^2 adjacency matrices where t is the number of node types.

For each node, the neighbor type-based aggregations are carried out by extracting the embedding of the neighbor nodes using the corresponding adjacency matrix,

according to the base model. These type-based neighbor aggregated matrices G_t are later combined using attention weights. This final aggregated matrix is concatenated with the node's content aggregated matrix and reduced using a linear layer. The equations are:

$$X^t[v_j,:] = F^t[v_j,:] \oplus \sum_t \alpha^t_{v_j} G^t[v_j,:] \tag{4.3}$$

$$G^t[v_j,:] = \frac{1}{|N^t(v_j)|} \sum_{v \in N^t(v_j)} \sigma\left(W_2 \cdot \left(F^t[v,:]\right)\right) \tag{4.4}$$

where X^t is the final embedding matrix for nodes of type t, N^t is the neighbor set of type t for the node, and $\alpha^t_{v_j}$ are the attention weights for combining neighbor aggregated matrices G_t. The attention weights, $\alpha^t_{v_j}$, are given by:

$$\alpha^t_{v_j} = \frac{\exp(\sigma(W_3 \cdot (G^t[v_j,:] \oplus F^t[v_j,:])))}{\sum_{v \in N^t(v_j)} \exp(\sigma(W_3 \cdot (G^t[v,:] \oplus F^t[v,:])))} \tag{4.5}$$

where σ is the LeakyReLU activation function. The given neighborhood aggregation in the equation 4.4 is for for HGNN model. Similarly, other base models, HAN and MAGNN, can also be used.

4.2.2 SMOTE

After projecting the nodes into the embedding space, some samples are selected randomly from the minority class of the training set to oversample via SMOTE. SMOTE generates synthetic nodes in the minority class by interpolating between nearest neighbor samples the same class. For a chosen minority node, v_j, with label Y_{v_j} and let its nearest neighbor from the same class be denoted by $\mathrm{NN}(v_j)$. This $\mathrm{NN}(v_j)$ is found by calculating the Euclidean distances between v_j and other nodes of the minority class in the embedding space, as shown below.

$$\mathrm{NN}(v_j) = \arg\min_{v \in V_t} \|X^t[v_j,:] - X^t[v,:]\|, \text{ s.t. } Y_{v_j} = Y_v \tag{4.6}$$

Using this nearest neighbor, synthetic nodes that inherit the same label as the source nodes can be generated. The synthetic node embeddings X_s^t (s implies synthetic) is generated via interpolation such that it lies in the vicinity of the source nodes, that is, on the line joining the two.

$$X_s^t[v_j, :] = (1 - r) \cdot X^t[\text{NN}(v_j), :] + r \cdot X^t[v_j, :] \tag{4.7}$$

where r is a random number such that $r \in [0, 1]$. New nodes are generated until the imbalance is removed. The newly generated samples, belonging to the objective synthetic node set V_t^{syn}, are appended to the original feature matrix. Subsequently, the adjacency matrices for the augmented graph A_{tu}, where u represents any arbitrary node type, are inherited from the adjacency matrices of the original graph A_{tu}^0.

By incorporating the encoder layer before oversampling, GraphSMOTE ensures that the generated samples are less prone to being noisy. This methodology stems from the observation that, following the aggregation phase, nodes belonging to the same class tend to have embeddings that are closer in the embedding space since they share similar neighbors. This facilitates the clustering of source nodes, defining distinct decision boundaries between classes. As a result, the oversampling process generates samples within these closely clustered regions, contributing to the technique's effectiveness.

4.2.3 Edge Generator

Since the synthetic nodes are isolated from the graph, they may lack the essential relational information in their encoding. Hence, an edge generator neural network model generates the edges for these synthetic nodes. The edge generator is trained to reconstruct the adjacency matrix for real nodes using their node representations, which can later be used for effective prediction of the edge between synthetic nodes.

The newly generated adjacency lists for each synthetic node are appended to the original matrix A^0_{tu}. The generator uses a vanilla design for simplicity, using the weighted inner product of the feature embedding of the respective node types between which the edges are to be generated.

$$\widehat{A}_{tu} = \sigma(X^t \cdot W_4 \cdot X^u) \odot A_{tu} \tag{4.8}$$

where $\widehat{A}_{tu}$ is the predicted adjacency matrix between node type t and u for the augmented graph and σ is the sigmoid function. To facilitate learning, the adjacency matrix is element-wise multiplied ($\odot$) by the inherited adjacency matrix from the oversampling step A_{tu} to remove the far unrelated edges from being learned. Separate generators are trained for edges between the different combinations of edge types. The edge Generator is trained using the loss function given by:

$$L_{\mathrm{e}} = \sum_{u,t} \|\widehat{A}^0_{tu} - A^0_{tu}\|^2 \tag{4.9}$$

where $\widehat{A}^0_{tu}$ refers to the predicted adjacency matrix for the real nodes, i.e., a graph without synthetic nodes.

Two strategies [11] are employed with the edge generator to incorporate predicted edges for synthetic nodes into the augmented adjacency matrix. In the first strategy, a threshold η (set to 0.5) is applied on the predicted synthetic edges, as given in equation (4.10). Here, the generator is optimized solely for the edge reconstruction task.

$$(A'_{tu})_{ij} = \begin{cases} 1, & \text{if } (\widehat{A}_{tu})_{ij} \geq \eta, \\ 0, & \text{otherwise.} \end{cases} \tag{4.10}$$

Here, A'_{tu} is the final adjacency matrix where new synthetic nodes and edges are inserted into A^0_{tu}, which is then sent to the classifier. For the second strategy, we keep the synthetic edges as soft edges instead of binary ones:

$$(A'_{tu})_{ij} = (\widehat{A}_{tu})_{ij} \tag{4.11}$$

In this scenario, gradients with respect to A'_{tu} can be backpropagated from the classifier. Consequently, both edge prediction loss and node classification loss can be used to optimize the edge generator.

4.2.4 Classifier

A simple GNN block is used as a classifier for the framework. It contains a heterogeneous neighbor aggregator layer similar to the one used in the encoder. This gives us freshly aggregated synthetic node embeddings with their relational information using the predicted edges of the synthetic nodes from the edge generator step. The final embedding is passed through an MLP head for classification.

$$P_{v_j} = \sigma(\text{MLP}(X'[v_j, :])) \tag{4.12}$$

where P_{v_j} is the prediction for node v_j with fresh embedding X', and σ is softmax function. The classifier module uses the cross-entropy loss given by:

$$L_{\text{cls}} = \sum_{v_j \in V'} \sum_{c} \left[1\left(Y_{v_j} == c\right) \cdot \log(P_{v_j})\right] \tag{4.13}$$

where Y_{v_j} is the true class of node v_j in the oversampled set V', and P_{v_j} is the predicted probability of node v_j belonging to class c. For evaluation, the class with the highest probability is taken as the predicted class, Y'_v, for node v:

$$Y'_v = \arg\max_c P_v. \tag{4.14}$$

4.3 Baseline settings and proposed Settings

For the purpose of the study, some baseline oversampling techniques will be compared against various proposed settings based on the HetGSMOTE pipeline in the previous section. These different experimental settings and baselines are listed in Table 4.1.

Table 4.1: Overview of Baseline settings and Proposed Settings

Setting	**Description**
no	Original datasets without oversampling.
up	Upsampling by duplicating source nodes. Synthetic node adjacency list: $A_{tu}[v_j, :] = A^0_{tu}[v, :]$.
smote	SMOTE on raw embedding space post-content aggregation. Synthetic node adjacency list: $A_{tu}[v_j, :] = \min(1, A^0_{tu}[v_1, :] + A^0_{tu}[v_2, :])$
reweight	Cost-sensitive approach adding higher weight to the loss function for minority classes.
embed-sm	SMOTE applied after neighbor aggregation, without edge generator. Assumes sufficient relational information is transferred post-interpolation.
em-smote	SMOTE after neighbor aggregation with an inherited adjacency matrix, showing edge generator impact.
gsmT	HetGSMOTE variant with edge generator trained only on edge loss.
gsmO	HetGSMOTE variant with edge generator trained on both edge loss and classification loss.
gsm-preT	HetGSMOTE with pre-trained edge generator and encoder on edge prediction task, fine-tuned on edge loss.
gsm-preO	Similar to gsm-preT, but edge generator fine-tuned on both edge loss and classification loss.

Among the proposed settings, we have chosen to perform pretraining of the Encoder and Edge Generator on edge prediction task in some settings (settings with 'pre') to study how models' prior knowledge of graph structure helps the framework. Following the equation (4.10), the 'T'-type setting trains the edge generator using only the edge prediction loss. In contrast, the 'O'-type setting uses both edge prediction and classification losses to train or fine-tune the edge generators, as motivated by the soft synthetic edge formulation in Equation (4.11). The effect of these different settings is studied later.

Chapter 5

Experiments

5.1 Optimization

The optimization objective for HetGSMOTE involves the optimization of weights for the encoder, edge generator, and classifier. As discussed before, we have two loss functions: the edge loss (L_e) from the edge generator and the classification loss (L_{cls}) from the classifier. They are combined as $L = \lambda \cdot L_e + L_{cls}$, where λ is a hyperparameter that controls the contribution from the different tasks. The resulting objective function of HetGSMOTE is the same as for GraphSMOTE:

$$\min_{\theta,\phi,\varphi}(L_{\text{cls}} + \lambda \cdot L_{\text{e}}) \tag{5.1}$$

where θ, ϕ, and φ are the parameters from the encoder, edge generator, and node classifier, respectively. As discussed before in Section 4.3, we are also pretraining the encoder and edge generator using L_e, in some settings.

5.2 Datasets

This work uses heterogeneous datasets of various sizes to show the domain independence of our method. These include AMiner A-II and DBLP bibliographic datasets, movie collaboration-based IMDb, and biomedical PubMed datasets.

Table 5.1: Information about datasets (labeled node type are marked with *)

Dataset	Nodes	Edges	Labels
AMiner-AII	author*: 20171 paper: 13250 venue: 18	author-paper: 42379 paper-paper: 14583 paper-venue: 13250	Classes: 4 Minority: 3
IMDb	movies*: 4666 directors: 2271 actors: 5845	movie-actor: 13990 movie-director: 4666	Classes: 3 Minority: 2
DBLP	paper*: 14328 author: 4057 term: 7723 conference: 20	author-paper: 19645 paper-term: 85810 paper-conference: 14328	Classes: 4 Minority: 3
PubMed	nodes: 63109 4 types	edges: 244986 10 types	Classes: 8 Minority: 6

AMiner-AII [6]: This is an academic dataset that includes paper publications in top venues related to artificial intelligence and data science from year 2006 to 2015. Each paper contains bibliographic content information such as 128-dimensional title embedding and a 128-dimensional abstract embedding. The author and venue attributes are extracted from the random walks in the graph using Par2Vec [18].

IMDb [7]: This is a subset of the Internet Movie Database (IMDB) dataset collected from [7]. This is a movie-based dataset where movies can belong to one of the three classes (action, comedy, drama) according to their genre. Their features are derived from the representation words of the plot keywords, a 128-dimensional embedding. The attributes of other nodes are determined from the random walks using Par2Vec [18].

DBLP[19]: This is a subset of the DBLP computer science bibliography website dataset collected from [19] containing bibliographic information for four node types with corresponding binary attributes of 256 dimensions. The representation words of

their paper keywords describe the author node's features.

PubMed [8]: This is bio-medical data describing relations between genes, diseases, chemicals, and species with corresponding attributes of 200 dimensions each. The links include gene-gene interactions, gene-disease associations, chemical-species relationships, etc. There are eight target labels for the disease nodes.

5.3 Experimental Settings:

To compare the performance of the methods, the baseline oversampling methods and HetGSMOTE are compared under various settings. These different experimental settings and baselines were listed in Table 4.1. We have both different edge loss variants as well as pretraining variants among these proposed variants, which will be studied later.

The experiments were conducted with HetGSMOTE with three base models: HGNN, HAN, and MAGNN. These models were evaluated in combination with 4 datasets: AMiner-AII, IMDb, DBLP, and Pubmed with different imbalance ratios ranging from 0.1 to 0.9 in steps of 0.1.

5.4 Evaluation Metric

In this study, I use three evaluation metrics for the imbalanced classification problem as used in GraphSMOTE. They are accuracy (ACC), mean AUC-ROC score (AUC), and mean F-1 score (F1).

Accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \tag{5.2}$$

Accuracy measures the proportion of nodes that are correctly classified. While it can be misleading in imbalanced scenarios due to its insensitivity to class distribution,

it remains useful in evaluating model performance after class imbalance has been addressed, such as through oversampling.

F1-Score is calculated, for each class, as:

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{5.3}$$

where,

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \quad \mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

Here, TP (True Positives) is the number of nodes that are correctly predicted as belonging to the class, FP (False Positives) is the number of nodes that are incorrectly predicted as belonging to the class, and FN (False Negatives) is the number of nodes from the class that was incorrectly predicted. A high F1 score reflects a good balance between precision and recall, meaning that the model is efficient in identifying positive cases while minimizing false positives and false negatives.

The AUC score, for each class, is calculated as:

$$\mathrm{AUC} = \int_0^1 \mathrm{TPR}(\mathrm{FPR})\, d(\mathrm{FPR}) \tag{5.4}$$

where,

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \quad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

Here, TPR (True Positive Rate) and FPR (False Positive Rate) are calculated as given above, where TN (True Negatives) is the number of nodes that are correctly predicted as not belonging to the class. AUC score tells us about the model's ability to rank a randomly selected positive example higher, compared to a randomly selected negative one.

Both AUC and F1 are non-weighted averaged over all classes to exhibit their performance over minority classes better. The threshold for the F1-score is allowed to be adaptively adjusted to give the best results for comparative study.

5.5 Configuration

All experiments have been done on a 64-bit system with Nvidia GPU (A100 80GB PCIe) and ADAM optimization. Three learning rates were tested and set on 1e-4 for all the datasets and models as per the performance with the weight decay 5e-4. The hyperparameter λ is set to 1e-6 for gsm-preO and gsmO settings, whereas it was set to 10 for others. The models are trained for 200 epochs since convergence occurred around 120-160. The embedding dimension was taken between 128 and 256, depending on the specific dataset and its attributes. The evaluation of each of the baselines and settings was done 10 times on 10 training subsets for reducing bias, with defined seeds for reproducibility, and the average results are presented for different imbalance ratios, up-sampling ratios, and training sizes of the majority class.

Chapter 6

Results and Discussion

Since the results were obtained across all base models, datasets, and imbalance ratios with 3 different metrics, the subsequent sections show the most significant results among each choice depending on the context of evaluation. Due to the limited space, the results shown here only contain the mean metrics across 10 training subsets as mentioned before. Bold values indicate the highest and underline indicate the second highest among its counterparts in each imbalance-ratio/ upsampling-ratio/training size, depending on the results.

6.1 Influence of Up-sampling Ratio

The *upsampling ratio* defines the fractional increase in minority samples to overcome the imbalance. An upsampling ratio of 1 implies the minority size was doubled by oversampling. Experiments with the upsampling ratio highlight the learning curve of our oversampling technique in different oversampling stages and the overgeneration problem when too many synthetic nodes are generated or overcompensated. In the experiments, the upsampling ratio scale is varied as $[0.2, 1.2]$ in 0.2 steps to obtain results for both high and low-ratio cases while keeping the imbalance ratio to 0.5. The results were obtained from HetGMOTE with HGNN base model on IMDb dataset and are tabulated in Table 6.1. We make the following observation from the table:

- HetGSMOTE variants show a similar trend where gsm-preO outperforms all the baselines regularly from the 0.6 ratio threshold in the ACC metric, as highlighted in the table. This shows that the generated node samples have an impact

on the performance without fail.

- For lower ratios, reweight, smote, and embed-sm give better results in 3 out of 8 cases. Baseline em-smote performing better might indicate the less importance of the edge generators when synthetic samples are low, which can affect the classification negatively compared to simpler oversampling.

- Another trend involves the decrease in the performance of 3 out of 4 proposed approaches as we go from 1.0 to 1.2, likely due to the overgeneration of synthetic samples, which affects the embedding quality.

Table 6.1: Test ACC vs Upsampling Ratio (HGNN base model and IMDb dataset)

Setting	**0.2**	**0.4**	**0.6**	**0.8**	**1.0**	**1.2**
no	0.5047	0.4956	0.5054	0.5011	0.5047	0.5091
up	0.5001	0.5013	0.505	0.504	0.4987	0.5092
smote	**0.5066**	0.4998	0.504	0.4976	0.5021	0.5045
reweight	0.4997	0.5011	0.5038	0.5031	0.4996	0.4997
embed-sm	0.5031	0.5000	0.4963	0.5056	0.5022	0.4963
em-smote	0.4951	**0.5078**	0.4976	0.5021	0.5033	0.4944
gsm-T	0.5002	0.5015	0.4926	0.4944	0.501	0.4917
gsm-O	0.5012	0.4921	0.5010	0.5010	0.4946	0.4900
gsm-preT	0.5018	0.5062	0.5060	0.5025	**0.5095**	0.5060
gsm-preO	0.5035	0.5065	**0.5062**	**0.5084**	0.5090	**0.5148**

6.2 Influence of Training size

Here, *Training size* refers to the maximum size of a minority class used for training. This experiment evaluates how well HetGSMOTE-based approaches perform on a small labeled training subset of the graph that emulates real-life scenarios and also in this semi-supervised setting. In the experiments, the training size varies between numbers less than 150 depending on the maximum size of available labeled nodes for

each class in each dataset. The validation and testing are taken to be higher than the training size. The training size more or less goes from extremely small (5%) of total size to small (10%) of the total size. The imbalance ratio and upsampling ratio are taken to be 0.5 and 'balanced,' respectively. The experiment was conducted on the DBLP dataset, where the largest minority class was available for this study. The results are tabulated in Table 6.2. We make the following observation from the table:

- HetGSMOTE variants show a similar trend where these variants perform weakly at extremely small training sizes, 50 and 80, and are outperformed by reweight. It may be because of the possible overcompensation of synthetic nodes by oversampling compared to the total number of target nodes, which might introduce noise in the embedding space.

- The performance of HetGSMOTE tends to improve as the training size increases and surpasses the rest of the baselines after the aforementioned thresholds (above 80 in this case).

Table 6.2: Test ACC vs training size (HGNN base model and DBLP dataset)

Settings	**50**	**80**	**100**	**150**
no	0.7692	0.8092	0.8067	0.8213
up	0.7581	0.8115	0.8087	0.8351
smote	0.7575	0.7917	0.7991	0.8295
reweight	**0.7908**	**0.8184**	0.8222	0.8422
embed-sm	0.7335	0.8079	0.7989	0.8237
em-smote	0.7715	0.8073	0.8097	0.8225
gsm-T	0.732	0.751	0.7787	0.7968
gsm-O	0.7861	0.7871	**0.8254**	0.8417
gsm-preT	0.7746	0.8126	0.8168	0.8327
gsm-preO	0.7526	0.8082	0.8128	**0.8457**

6.3 Influence of pre-Training

As mentioned before in Section 4.3, the pre-trained settings gsm-preT and gsm-preO had encoders and edge generator retained on edge-prediction task and fine-tuned on different losses. Figure 6.1 shows the evolution of the performance of different settings across training epochs. where the pre-trained settings and their counterparts are compared, the performance for the pre-trained models was higher compared to the normal settings, gsm-T, and gsm-O. This shows that pretraining gives an edge to the model in identifying graph Structures before the actual task of Node classification takes place.

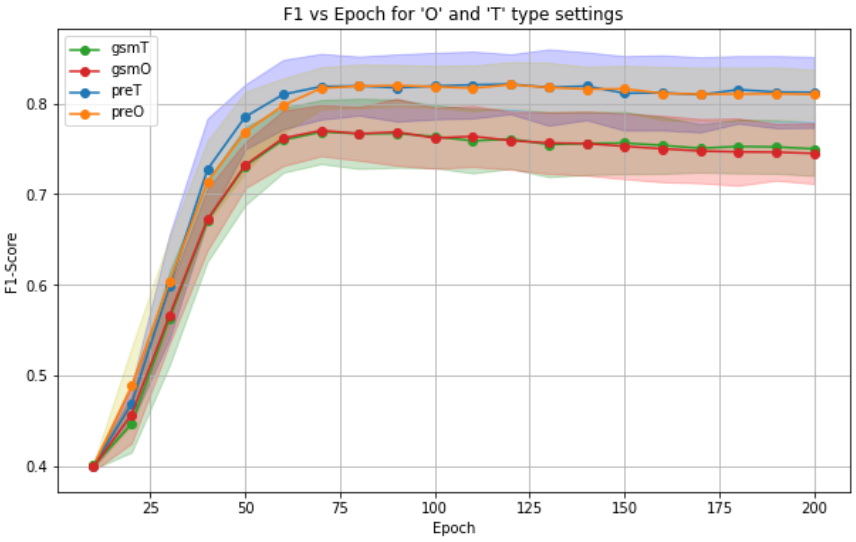

Figure 6.1: Comparison of the effect of pretraining (HGNN base model and IMDb dataset)

6.4 Performance of HetGSMOTE

To show the superior performance of the HetGSMOTE framework, we show its performance with the HAN base model on IMDb datasets. The results are tabulated in Table 6.3 for all metrics. Here, we make the following observations:

- HetGSMOTE variants have shown higher performance than their counterparts in most cases across all three metrics, showcasing its superiority over similar oversampling techniques.

- On accuracy metric, a reliable metric in the classification task, we can see that *gsm-preO* has performed the best among the proposed settings. The same is visible across other metrics too.

Table 6.3: Performance vs. imbalance ratio (HAN base model and IMDb dataset)

Metric	**Setting**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
ACC	no	0.4473	0.4575	0.4787	0.4997	0.5167	0.515	0.525	0.5162	0.518
	up	0.4428	0.4563	0.4718	0.4852	0.5108	0.5083	0.5203	0.5198	0.5233
	smote	0.4507	0.456	0.4773	0.4945	0.509	0.5248	0.5205	0.5165	0.5212
	reweight	0.452	0.4763	0.499	0.5178	0.5148	0.523	0.528	0.5312	0.5342
	embed-sm	0.4453	0.4495	0.4595	0.4813	0.4895	0.4985	0.507	0.5088	0.5187
	em-smote	0.4553	0.4873	0.4935	0.5028	0.5172	0.524	0.5288	0.5268	0.5285
	gsm-T	0.449	0.4812	0.4947	0.5013	0.5195	0.5252	**0.5333**	0.5253	0.537
	gsm-O	0.4578	0.483	0.4875	0.5052	0.5242	0.5173	0.526	0.526	0.535
	gsm-preT	0.4713	0.4835	**0.5048**	0.516	0.52	0.5338	0.5282	0.54	**0.5442**
	gsm-preO	**0.4718**	**0.4893**	0.4948	**0.518**	**0.5278**	**0.5367**	0.5305	**0.5438**	0.5338
F1-Score	no	0.5498	0.5542	0.5566	0.564	**0.5731**	0.5628	0.5702	0.566	0.5681
	up	0.5451	0.5458	0.5513	0.5534	0.5588	0.5616	0.5665	0.5609	0.5664
	smote	0.5448	0.5503	0.5487	0.5575	0.5607	0.5678	0.5666	0.5663	0.5652
	reweight	0.541	0.5509	0.5563	0.5601	0.5691	0.567	0.5693	0.5743	**0.5805**
	embed-sm	0.5444	0.5426	0.5446	0.5493	0.5551	0.5554	0.5598	0.5576	0.5707
	em-smote	0.5473	0.5532	0.5555	0.5582	0.5684	0.5677	0.5721	0.5656	0.5681
	gsm-T	0.5454	0.552	0.5615	0.5563	0.5685	0.5724	0.5726	0.5714	0.5697
	gsm-O	0.5467	0.553	0.5574	0.5572	0.567	0.562	0.5681	0.5681	0.5763
	gsm-preT	0.5514	0.5592	**0.5649**	0.5653	0.5702	**0.5728**	0.568	**0.578**	0.5785
	gsm-preO	**0.5533**	**0.561**	0.5622	**0.5705**	0.5695	**0.5728**	**0.5747**	0.5761	0.5786
AUC	no	0.6614	0.6702	0.6815	0.6987	**0.7059**	0.6991	0.7069	0.7026	0.7065
	up	0.652	0.6622	0.6685	0.6752	0.685	0.6922	0.7019	0.6942	0.7013
	smote	0.6576	0.6687	0.6697	0.6849	0.6918	0.7026	0.6988	0.7044	0.7028
	reweight	0.6454	0.6792	0.6863	0.6928	0.7033	0.7003	0.7027	0.7066	0.7162
	embed-sm	0.6456	0.651	0.6604	0.6698	0.6763	0.6828	0.6891	0.6912	0.7048
	em-smote	0.6652	0.6802	0.6904	0.6901	0.7011	0.7044	**0.7118**	0.7021	0.7064
	gsm-T	0.6582	0.6792	0.6837	0.688	0.703	0.7063	0.7038	0.7084	0.7083
	gsm-O	0.6618	0.6753	0.6828	0.6883	0.7038	0.6985	0.7044	0.7063	0.7159
	gsm-preT	0.6641	0.6803	**0.6966**	0.699	0.7049	0.7069	0.7086	0.7125	0.7129
	gsm-preO	**0.6669**	**0.6851**	0.6924	**0.705**	0.7022	**0.7131**	0.7051	**0.7131**	**0.7171**

6.5 Variation across datasets

To show the consistency of performance across different datasets, we have shown the results in Table 6.4 with the test accuracy performance of HetGSMOTE on all datasets. Here, we make the following observation:

- The HetGSMOTE variants perform comparatively better than the baselines in most cases across all datasets. Variation in accuracy can be seen across datasets, since the model performs at different levels on different datasets, irrespective of the HetGSMOTE framework, which is expected.

Table 6.4: Test ACC vs imbalance ratios (HGNN base model and all datasets)

Dataset	Settings	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
IMDb	reweight	0.452	0.4763	0.499	0.5178	0.5148	0.523	0.528	0.5312	0.5342
	embed-sm	0.4453	0.4495	0.4595	0.4813	0.4895	0.4985	0.507	0.5088	0.5187
	em-smote	0.4553	0.4873	0.4935	0.5028	0.5172	0.524	0.5288	0.5268	0.5285
	gsm-T	0.449	0.4812	0.4947	0.5013	0.5195	0.5252	**0.5333**	0.5253	0.537
	gsm-O	0.4578	0.483	0.4875	0.5052	0.5242	0.5173	0.526	0.526	0.535
	gsm-preT	0.4713	0.4835	**0.5048**	0.516	0.52	0.5338	0.5282	0.54	**0.5442**
	gsm-preO	**0.4718**	**0.4893**	0.4948	**0.518**	**0.5278**	**0.5367**	0.5305	**0.5438**	0.5338
AMINER	reweight	0.922	0.949	0.9573	0.959	0.964	0.9623	0.9653	0.9627	0.9607
	embed-sm	0.9323	0.9523	0.9623	0.963	0.958	0.9673	0.9597	0.965	0.9627
	em-smote	0.9477	0.9563	0.9643	0.962	0.9653	0.962	0.9667	0.964	0.9627
	gsm-T	0.9343	0.9583	0.964	**0.9657**	0.962	**0.968**	0.9657	0.9657	**0.9677**
	gsm-O	0.9377	0.9597	0.9617	0.9633	**0.9657**	0.964	0.9613	0.962	0.967
	gsm-preT	0.9463	0.9597	0.9617	0.96	0.963	0.966	0.9633	**0.9667**	0.9657
	gsm-preO	**0.9513**	**0.9607**	**0.9647**	**0.9657**	0.964	0.964	**0.967**	0.9623	0.9673
Pubmed	reweight	0.1675	0.1681	0.1525	0.1537	0.1581	0.1575	**0.1644**	0.1562	0.1525
	embed-sm	0.1401	0.1494	0.1644	0.1506	0.1581	0.1525	0.1537	0.1531	**0.1594**
	em-smote	0.154	0.1544	0.1569	0.1531	0.1537	**0.16**	0.1562	0.1456	0.1544
	gsm-T	0.1508	0.1512	0.165	0.1469	0.1594	0.1512	0.145	0.1569	0.1544
	gsm-O	0.1477	0.1481	0.1563	0.15	0.1575	0.15	0.15	0.1494	0.1554
	gsm-preT	0.1635	0.1644	0.1688	**0.1544**	**0.1644**	0.1537	0.1512	**0.1588**	0.1562
	gsm-preO	**0.17**	**0.1713**	**0.1694**	0.1512	0.1631	0.15	0.1581	0.1562	0.1569
DBLP	reweight	0.6464	0.6897	0.7354	**0.7696**	**0.7996**	0.805	0.7756	0.7914	0.7771
	embed-sm	0.6367	0.6826	0.7047	0.7381	0.7666	0.7685	0.7679	0.774	0.7805
	em-smote	0.6447	0.6837	0.7154	0.746	0.7637	0.7751	0.7359	0.762	0.7509
	gsm-T	0.639	0.6678	**0.7518**	0.7576	0.7769	0.791	0.7857	0.7882	0.8001
	gsm-O	0.6291	0.6567	0.7149	0.7557	0.7593	**0.8079**	**0.7957**	0.7924	0.79
	gsm-preT	0.6349	0.6927	0.7331	0.7255	0.7691	0.7767	0.7945	**0.802**	**0.8002**
	gsm-preO	**0.6649**	**0.7136**	0.7441	0.7615	0.7606	0.7671	0.7639	0.7841	0.7867

6.6 Influence of base model

To study how different base models influence performance, we have evaluated the framework with HAN, MAGNN, and HGNN base models. The results are shown with test accuracy for the IMDb dataset in Table 6.5. We make the following observations:

- Here, on the IMDb dataset, HAN performed the best among its model counterparts. Since it's known that different models have different capacities to learn from data, it is expected that the performance will vary across models, which is visible in the results.
- Within each model, The HetGSMOTE framework, especially gsm-preO, has performed better than in counterpart in most cases. This shows the impact of the HetGSMOTE framework, regardless of the base models.

Table 6.5: Test ACC vs imbalance ratio (all base models and IMDb dataset)

Model	Settings	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
HAN	reweight	0.452	0.4763	0.499	0.5178	0.5148	0.523	0.528	0.5312	0.5342
	embed-sm	0.4453	0.4495	0.4595	0.4813	0.4895	0.4985	0.507	0.5088	0.5187
	em-smote	0.4553	0.4873	0.4935	0.5028	0.5172	0.524	0.5288	0.5268	0.5285
	gsm-T	0.449	0.4812	0.4947	0.5013	0.5195	0.5252	**0.5333**	0.5253	0.537
	gsm-O	0.4578	0.483	0.4875	0.5052	0.5242	0.5173	0.526	0.526	0.535
	gsm-preT	0.4713	0.4835	**0.5048**	0.516	0.52	0.5338	0.5282	0.54	**0.5442**
	gsm-preO	**0.4718**	**0.4893**	0.4948	**0.518**	**0.5278**	**0.5367**	0.5305	**0.5438**	0.5338
MAGNN	reweight	0.3567	0.348	0.3672	0.4057	0.3383	0.3523	0.3475	0.338	0.3398
	embed-sm	0.346	0.3463	0.3482	0.3503	0.3462	0.3385	0.345	0.3422	0.3628
	em-smote	0.4298	0.4703	0.4883	**0.5008**	0.4742	0.497	0.5128	0.4887	0.5038
	gsm-T	0.3663	0.3547	0.3515	0.353	0.3615	0.3562	0.3828	0.3632	0.3625
	gsm-O	0.3747	0.388	0.3593	0.3637	0.3885	0.3835	0.3668	0.3612	0.3895
	gsm-preT	0.4455	**0.4805**	0.4783	0.4972	0.506	0.518	**0.5148**	0.5182	**0.5163**
	gsm-preO	**0.4537**	0.4752	**0.489**	0.4975	**0.5095**	**0.5252**	0.5117	**0.5202**	0.509
HGNN	reweight	0.4335	0.4396	0.4556	0.4664	0.4885	0.4878	**0.508**	0.4991	0.5116
	embed-sm	0.4346	0.457	0.4658	0.4748	0.5003	0.4936	0.5026	0.5006	0.4948
	em-smote	0.4335	0.442	0.4561	0.479	0.4936	0.5013	0.4971	0.4981	0.5011
	gsm-T	0.4343	0.438	0.4606	0.475	0.4826	0.4923	0.4921	0.5061	**0.5126**
	gsm-O	0.4311	0.4331	0.4606	0.4713	0.488	0.4918	0.499	0.504	0.5058
	gsm-preT	0.4405	**0.4571**	0.4613	0.4863	0.4911	0.5028	0.5065	0.5001	0.5095
	gsm-preO	**0.444**	0.448	**0.476**	**0.4946**	**0.5026**	**0.5078**	0.4981	**0.507**	0.508

Chapter 7

Summary

7.1 Conclusions

Our findings demonstrate the effectiveness of the HetGSMOTE approach for heterogeneous graphs across various datasets and models across various class-imbalance conditions. Our approach is domain-independent, working on diverse datasets and surpassing the baselines in the majority of the cases. It is also shown that this framework is capable of handling different base models and, hence, can be further improved when a new model appears in the domain.

We have also shown the effect of training size and upsampling ratio, where a higher training size aids performance, whereas too low of an upsampling ratio hinders performance. We have also shown how the choice of pretraining has helped the model to exceed the performance. In conclusion, The effect of the HetGSMOTE framework, regardless of the base model and dataset, has been shown by its superior performance.

7.2 Future Directions

In future experiments, the work can be taken in different directions. For starters, testing the approach on tasks like link prediction, edge type classification, and node-representation learning is important for graph-based studies as they also tend to suffer from class imbalance problems. The approach can also be improved with different choices of newer models capable of transfer learning or student-teacher models to improve beyond a simple pretraining framework.

References

[1] Sanchez-Lengeling, B., Reif, E., Pearce, A., & Wiltschko, A. (2021). A Gentle Introduction to Graph Neural Networks. Distill, 6(8), 10.23915/distill.00033. https://doi.org/10.23915/distill.00033 2

[2] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. International Conference on Learning Representations. 1

[3] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. Proceedings of the 31st International Conference on Neural Information Processing Systems, 1025–1035. 1

[4] Shi, C. (2022). Heterogeneous Graph Neural Networks. In L. Wu, P. Cui, J. Pei, & L. Zhao (Eds.), Graph Neural Networks: Foundations, Frontiers, and Applications (pp. 351–369). Springer Nature Singapore. 1, 2, 3

[5] Liu, J., Shi, C., Yang, C., Lu, Z., & Yu, P. S. (2022). A survey on heterogeneous information network based recommender systems: Concepts, methods, applications and resources. AI Open, 3, 40–57. 1, 3

[6] Zhang, C., Song, D., Huang, C., Swami, A., & Chawla, N. V. (2019). Heterogeneous Graph Neural Network. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 793–803. 1, 3, 4.2.1, 5.2

[7] Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019). Heterogeneous Graph Attention Network. The World Wide Web Conference, 2022–2032. 1, 3, 4.2.1, 5.2

[8] Yang, C., Xiao, Y., Zhang, Y., Sun, Y., & Han, J. (2022). Heterogeneous Network Representation Learning: A Unified Framework With Survey and Benchmark. IEEE Transactions on Knowledge and Data Engineering, 34(10), 4854–4873. 1, 5.2

[9] Nguyen, T.-K., Liu, Z., & Fang, Y. (2023). Link Prediction on Latent Heterogeneous Graphs. Proceedings of the ACM Web Conference 2023, 263–273. 1

[10] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357. 1, 2

[11] Zhao, T., Zhang, X., & Wang, S. (2021). GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 833–841. 1, 2, 3, 4.2.3

[12] Han, H., Wang, W.-Y., & Mao, B.-H. (2005). Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In D.-S. Huang, X.-P. Zhang, & G.-B. Huang (Eds.), Advances in Intelligent Computing (pp. 878–887). Springer. 2, 3

[13] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 1322–1328. 3, 3

[14] Last, F., Douzas, G., & Bacao, F. (2018). Oversampling for Imbalanced Learning Based on K-Means and SMOTE. Information Sciences, 465, 1–20. 3, 3

[15] Kovács, G. (2019). An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets. Applied Soft Computing, 83, 105662. 3

[16] Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). SMOTEBoost: Improving Prediction of the Minority Class in Boosting. In N. Lavrač, D. Gamberger, L. Todorovski, & H. Blockeel (Eds.), Knowledge Discovery in Databases: PKDD 2003 (Vol. 2838, pp. 107–119). Springer Berlin Heidelberg. 3

[17] Zhang, Z., Cui, P., & Zhu, W. (2022). Deep Learning on Graphs: A Survey. IEEE Transactions on Knowledge and Data Engineering, 34(1), 249–270. 3

[18] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems, 26. 5.2

[19] Fu, X., Zhang, J., Meng, Z., & King, I. (2020). MAGNN: Metapath Aggregated Graph Neural Network for Heterogeneous Graph Embedding. Proceedings of The Web Conference 2020, 2331–2341. 3, 4.2.1, 5.2

[20] Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., & Guo, M. (2018). GraphGAN: Graph representation learning with generative adversarial nets. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, 2508–2515. 3

[21] Hsu, H. C., Lin, T.-L., Wu, B.-J., Hong, M.-Y., Lin, C., & Wang, C.-Y. (2024). FincGAN: A Gan Framework of Imbalanced Node Classification on Heterogeneous Graph Neural Network. ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5750–5754. 3

[22] Xia, R., Zhang, C., Zhang, Y., Liu, X., & Yang, B. (2024). A novel graph oversampling framework for node classification in class-imbalanced graphs. Science China Information Sciences, 67(6), 1–16. 3

[23] Wu, L., Xia, J., Gao, Z., Lin, H., Tan, C., & Li, S. Z. (2022). GraphMixup: Improving Class-Imbalanced Node Classification by Reinforcement Mixup and Self-supervised Context Prediction. Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part IV, 519–535. 3

[24] Pan, T., Zhao, J., Wu, W., & Yang, J. (2020). Learning imbalanced datasets based on SMOTE and Gaussian distribution. Information Sciences, 512, 1214–1233. 3

[25] Zhao, M. (2025). Synthetic minority oversampling technique based on natural neighborhood graph with subgraph cores for class-imbalanced classification. The Journal of Supercomputing, 81(1), 248. 3

[26] Jin, C., Ni, H., Miao, F., Zheng, T., Song, M., & Liu, Z. (2025). BARE: Balance representation for imbalance multi-class node classification on heterogeneous information networks. Expert Systems with Applications, 272, 126506. 3

[27] Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2012). DBSMOTE: Density-Based Synthetic Minority Over-sampling TEchnique. Applied Intelligence, 36(3), 664–684. 3

Appendix A

Machine Learning Glossary

Here is a list of domain-specific terms and their definitions in the context of training Machine Learning, given in Table form (Tables A.1 and A.2).

Table A.1: Glossary of Machine Learning Terms

Term	**Definition**
Machine Learning	A field of artificial intelligence that enables systems to learn patterns from data and make predictions or decisions without being explicitly programmed.
Dataset	A structured collection of data used for training, validating, and testing machine learning models.
Labels	Known target values or categories assigned to data instances, used for supervised learning.
Classes	The distinct categories or groups into which labeled data points are classified.
Model	A mathematical representation trained on data to perform a task such as classification or regression.
Neural Network	A type of machine learning model composed of layers of interconnected nodes that transform input data through learned weights.
Layers	Stacked units in a neural network where computations occur; typically include input, hidden, and output layers.
MLP	Multilayer Perceptron: A fully connected feedforward neural network consisting of multiple layers used for nonlinear function approximation.
Encoder	A component that transforms input data into a compact or latent representation, often used in representation learning tasks.

Table A.2: Glossary of Machine Learning Terms (continued)

Term	**Definition**
Embedding	A vector representation of discrete objects (e.g., nodes, words) in a continuous space, capturing structural or semantic similarity.
Message Passing	A core operation in graph neural networks where nodes exchange information with their neighbors to update their representations.
Classification	A task where the model predicts a discrete label or category for each input instance.
Regression	A task where the model predicts a continuous value for each input instance.
Training	The process of learning patterns in data, for performing a specific task, by adjusting model parameters to minimize a loss function.
Pretraining	The process of learning general patterns in data before training for a certain task.
Validation	The process of evaluating model performance on held-out data during training to tune hyperparameters and prevent overfitting.
Testing	The final evaluation of a trained model's performance on a separate dataset to assess generalization ability.
Loss	A measure of the difference between the model's predictions and the true values, guiding the learning process.
Backpropagation	Process of computing gradients from the loss through the model.
Optimization	The process of minimizing the loss function by updating model parameters using gradients from backpropagation.
Accuracy	The proportion of correctly predicted labels among all predictions. It is often used for node classification tasks.
F1 Score	The harmonic mean of precision and recall, providing a balanced measure of performance, especially in imbalanced datasets.
AUC (Area Under the Curve)	A performance metric that evaluates the ability of a model to distinguish between classes across different thresholds.