mathematics

MDPI

Article

Edge and Node Enhancement Graph Convolutional Network: Imbalanced Graph Node Classification Method Based on Edge-Node Collaborative Enhancement

Jiadong Tian [1,2], Jiali Lin [3] and Dagang Li [1,2,*]

[1] School of Computer Science and Engineering, Macau University of Science and Technology, Macau 999078, China; 2009853gia30009@student.must.edu.mo
[2] Zhuhai-M.U.S.T. Science and Technology Research Institute, Zhuhai 519031, China
[3] Key Laboratory of Neuroeconomics, Guangzhou Huashang College, Guangzhou 511300, China; jiali_lin@gdhsc.edu.cn
* Correspondence: dgli@must.edu.mo

Abstract: In addressing the issue of node classification with imbalanced data distribution, traditional models exhibit significant limitations. Conventional improvement methods, such as node replication or weight adjustment, often focus solely on nodes, neglecting connection relationships. However, numerous studies have demonstrated that optimizing edge distribution can improve the quality of node embeddings. In this paper, we propose the Edge and Node Collaborative Enhancement method (ENE-GCN). This method identifies potentially associated node pairs by similarity measures and constructs a hybrid adjacency matrix, which enlarges the fitting space of node embedding. Subsequently, an adversarial generation strategy is employed to augment the minority class nodes, thereby constructing a balanced sample set. Compared to existing methods, our approach achieves collaborative enhancement of both edges and nodes in a concise manner, improving embedding quality and balancing the training scenario. Experimental comparisons on four public graph datasets reveal that, compared to baseline methods, our proposed method achieves notable improvements in Recall and AUC metrics, particularly in sparsely connected datasets.

Keywords: edge enhancement; node enhancement; graph convolutional networks; imbalanced graph; node classification

MSC: 68R10; 68T05

Academic Editor: Sergio Gómez

Received: 23 February 2025
Revised: 13 March 2025
Accepted: 20 March 2025
Published: 22 March 2025

Citation: Tian, J.; Lin, J.; Li, D. Edge and Node Enhancement Graph Convolutional Network: Imbalanced Graph Node Classification Method Based on Edge-Node Collaborative Enhancement. *Mathematics* **2025**, *13*, 1038. https://doi.org/10.3390/math13071038

1. Introduction

Imbalance of data in graphs is prevalent in many real-world scenarios. In the field of anomaly detection [1], such as in the task of credit card fraud recognition, it is essential to predict the category of other users based on the labels of known users, such as fraudulent or normal users. Typically, the number of fraudulent users is significantly smaller than that of normal users, resulting in a class imbalance problem that needs to be addressed. Similarly, in the field of disease prediction [2], accurately diagnosing rare positive cases is of critical importance. In these tasks, we often encounter a severe class imbalance issue, yet the accuracy of the minority class is crucial. Training in such imbalanced scenarios may fail to recognize minority nodes, as nodes from the majority class tend to dominate and skew the training process toward the majority class. Many classical graph datasets also reflect this problem. The CiteSeer dataset [3] contains citation relationships between

papers. In this dataset, minority class nodes make up only 7.9% and have an average of only 1.37 connections per node.

Currently, the methods based on nodes usually adopt traditional methods, such as resampling and reweighting to cope with data imbalance scenarios [4]. ReNode (RN) [5] enhances the training weights of labeled nodes closer to the topological center. An Imbalanced Generative Adversarial Graph Convolutional Network (ImGAGN) [6] designs a graph structure data generator that synthesizes a set of composite nodes to simulate minority class nodes. A Dual-Regularized Graph Convolutional Network (DR-GCN) [7] incorporates conditional adversarial regularization layers and latent distribution alignment regularization layers for data generation. GraphMixup [8] implements a generative approach for node features and connections to generate the desired number of samples adaptively. These methods often focus on generating new nodes, while the generation of new nodes must consider both the properties of the new nodes and the connectivity between the new nodes and the original graph. These issues often result in overly complex generator designs. Virtual nodes are generally randomly connected to minority class nodes on the original graph, which greatly disrupts the structure of the original minority class nodes and is not beneficial for the classifier to learn decision boundaries.

The above approaches do not improve the overall graph topology at the connection level. However, numerous research studies demonstrate that enhancing the overall graph topology can lead to superior node embeddings and subsequently improve the performance of subsequent tasks. A Geometric Graph Convolutional Network (Geom-GCN) [9] aggregates the neighborhood and potential space of nodes and improves the embedding quality of nodes through this two-level aggregation. A Universal Graph Convolutional Networks (U-GCN) [10] goes a step further, improving the performance of the GCN by integrating K-neighbors, first-order neighbors, and second-order neighbors. In addition, some methods introduce the concept of clique to update edges by graph density and subgraphs, so as to update the graph structure [11,12]. These findings indicate that an improved graph structure has a positive effect on node neighborhood aggregation.

In addition to the theoretical and empirical studies mentioned above, the algorithm of this paper is also inspired by an objective fact. We believe that nodes with similar attributes should have a higher probability of being connected, although these connections are sometimes absent in the original graphs. For instance, in a citation network, two similar articles should theoretically have a citation relationship, but the relationship may be missing because the articles were published at the same time, or the authors did not follow each other's work. Also, in social networks, connections between nodes are generally constructed through relationships between users. If two people know each other but do not follow each other, this will result in some missing information in the dataset. This phenomenon is relatively common in reality. Edge enhancement can compensate for this connectivity, so that the dataset can more accurately reflect the real world. However, edge enhancement methods are generally computationally complex and have limitations in terms of the number of edges that can be updated.

We aimed to leverage the advantages of edge enhancement and node enhancement while ensuring the algorithm design remains as streamlined as possible. So we designed the ENE-GCN (Edge and Node Enhancement Graph Convolutional Network) method. In our implementation, edge enhancement first calculates the node similarity to enhance the edges between nodes with higher similarity. Then, the node embeddings are derived through semi-supervised embedding, followed by generating minority class data to balance the training dataset. This implementation provides flexibility to adjust the number of enhanced edges and allows direct generation of virtual nodes. It effectively circumvents

the complex calculations associated with edge enhancement, as well as the intricate design and implementation required for generating virtual nodes on the original graph.

The main contributions of this study are as follows:

1. We proposed a method that combines edge enhancement and node enhancement. The method first performs edge enhancement, then generates data to balance the training scenario after node embeddings are obtained.
2. We performed classification experiments and ablation experiments on several imbalanced graph datasets. The experimental results show that the proposed algorithm is superior to the baseline methods, especially for sparsely connected nodes. The ablation experiments further demonstrate that both steps, edge enhancement and node enhancement, positively improve the classification results.
3. We compared the classification results under various parameter settings and discussed a reasonable range of value for the parameter.

2. Related Research

2.1. Node Enhancement Methods

At present, Graph Neural Network (GNN) methods usually deal with the node classification problem in imbalanced datasets by generating new nodes or edges or using weights. These methods include ImGAGN [6], DR-GCN [7], Reweighted Adversarial Graph Convolutional Networks (RA-GCN) [13], GraphSMOTE [14], GraphMixup [8], and others. These approaches generally focus on designing graph data generators to synthesize a set of minority class nodes to alleviate node imbalance. Furthermore, ReNode [5] introduced the idea of weight, which assigns different weights to different categories of nodes in the training process by measuring the imbalance of nodes. Balanced Neighbor Exploration (BNE) [15] introduced the idea of balanced neighborhood exploration by assigning multiple labels to nodes connecting different classes, thus mitigating the imbalance issue.

The aforementioned generative methods often focus on generating new nodes similar to the minority class nodes in the original graph data, which can improve the classifier's attention to minority classes. However, the generation of new nodes must consider both attribute and edge generation, which often leads to the need for complex design and implementation of the generator. Regarding the connection of new nodes to the original graph, the existing algorithms generally add new nodes directly to the original graph, which may destroy the structure of the nodes in the original graph. RN and BNE avoid the complex design and computation involved in generative methods. However, they cannot optimize decision boundaries, which may even make them more ambiguous. Moreover, on extremely imbalanced graph datasets, they fail to fully balance the dataset.

2.2. Edge Enhancement Methods

There are many methods to adjust the distribution of edges directly by updating the topology of the graph. Learning Discrete Structures (LDS) [16] use a two-layer framework to complement the connections in the original graph by learning the probabilities of edge distributions in the graph. Deep Iterative and Adaptive Learning for Graph Neural Networks (DIAL-GNN) [17], based on node similarity, proposes an iterative method to search for hidden connections to enhance the original graph structure. Geom-GCN [9] and Deformable GCNs [18] achieve outstanding performance by identifying and adding higher-order neighbors. In addition to the direct addition of edges, the introduction of weights can also be considered as a way to update the graph topology. The Graph Attention Network (GAT) [19] and GATv2 [20] adjust the edge weights for node embedding by introducing an attention mechanism. Learnable Graph Convolutional Attention Networks (L-CAT) [21] add convolutional operations in front of the attention mechanism and fuse

three types of GNN models to improve the neighborhood aggregation process. In addition, D2GCN [22] and SDGCN [23] utilize other classes of nodes in neighboring nodes to enhance the node representation. FAGCN [24] and BAGCN [25] try to introduce negative weights to strengthen the collection of features of different types of nodes in neighboring nodes.

In addition to directly modifying edge distributions, many researchers have attempted to change neighborhood distributions by considering hidden edge distributions in higher-order neighborhoods. The Graph Aggregating-Repelling Network (GARN) [26] extracts homogeneous and heterogeneous information of neighboring nodes by fusing low-pass and high-pass graph filters. MixHop [27] and HN-GCCF [28] leverage higher-order information through concatenation. GCN-IED [29] incorporates both direct edges, which rely on local neighborhood similarity, and hidden edges, obtained by aggregating information from multi-hop neighbors. Graph diffusion techniques based on PageRank [30] also keep evolving due to their simplicity and effectiveness. GPR-GNN [31] introduces a new architecture that adaptively learns the GPR weights so as to jointly optimize node feature and topological information extraction. HAPGNN [32] proposes a reinforced adaptive method, which pays more attention to important connections and assigns independent learnable weights to such connections. Multi-PageRank Gravity Centrality (PRGC) [33] proposes a new method to calculate the distance between two nodes, thereby optimizing the graph structure, and then proposes an improved Gravity Centrality to identify key nodes. There are also methods [34–37] that aim to make better use of the information in higher-order neighbors. However, these methods usually require careful tuning of hyperparameters, and they often fall short in updating the full graph.

Node enhancement class methods attempt to classify nodes on imbalanced graphs by improving node representation. Edge enhancement class methods show that improvements in edges positively affect the embedding results. However, few methods combine edge enhancement with node enhancement for classifying nodes on imbalanced graphs. Given this, our approach optimizes the embedding results by enhancing the edges and then generating nodes to balance node numbers, ultimately achieving better classification results.

3. Proposed Method

3.1. Preliminary

In this paper, we focus on the task of node binary classification in imbalanced graphs. Before introducing our proposed ENE-GCN method, we need to briefly introduce the basic concepts required for our approach.

Imbalanced Graph: A graph is considered imbalanced when there is a significant difference in the number of nodes in different categories. In an imbalanced graph, a general model may learn a decision boundary that cannot efficiently classify a small number of samples. Since the majority class nodes dominate and overwhelm the minority class nodes, the training process is biased toward the majority class.

Generative Adversarial Networks: GANs consist of generators and discriminators and are mainly used in the field of data enhancement. The main goal of the generator is to generate fake data that simulate the underlying distribution of real data to fool the discriminator. On the other hand, the discriminator aims to correctly classify the real training data and the fake data generated by the generator. The generator and the discriminator engage in an adversarial process to make the generated data approximate the target data distribution, ultimately achieving the purpose of using the generator for data augmentation.

3.2. Overview and Implementation

Objectively, nodes with similar attributes should have a higher probability of being connected, but these connections are sometimes absent in the original graphs. Therefore, we propose the ENE-GCN, which combines edge enhancement and node enhancement, to address the problem of imbalanced classification in graphs. Firstly, the similarity of nodes is computed based on their attributes, and the original graph data are enriched with edges according to this similarity. Then, node embedding is obtained through semi-supervised embedding, followed by generating minority class data, ensuring a balanced number of nodes across different categories. The edge enhancement technique compensates for potentially missing connection information in the original graph and strengthens the clustering of nodes within the same category, leading to clearer decision boundaries. After obtaining the embedding results, the balanced dataset is then obtained by adversarial generation, which avoids the complex process of generating nodes directly on the graph, thereby making both the design and the computation simpler and more intuitive. The structure of the edge enhancement method is shown in Figure 1. The solid red circles represent the minority class nodes, and the solid blue circles represent the majority nodes. The solid black lines and red dotted lines represent the actual edge and the enhanced edge, respectively. The hollow red circles represent the minority class node data generated using GAN. After embedding, the smaller dots represent the embedding vectors of the various types of nodes.

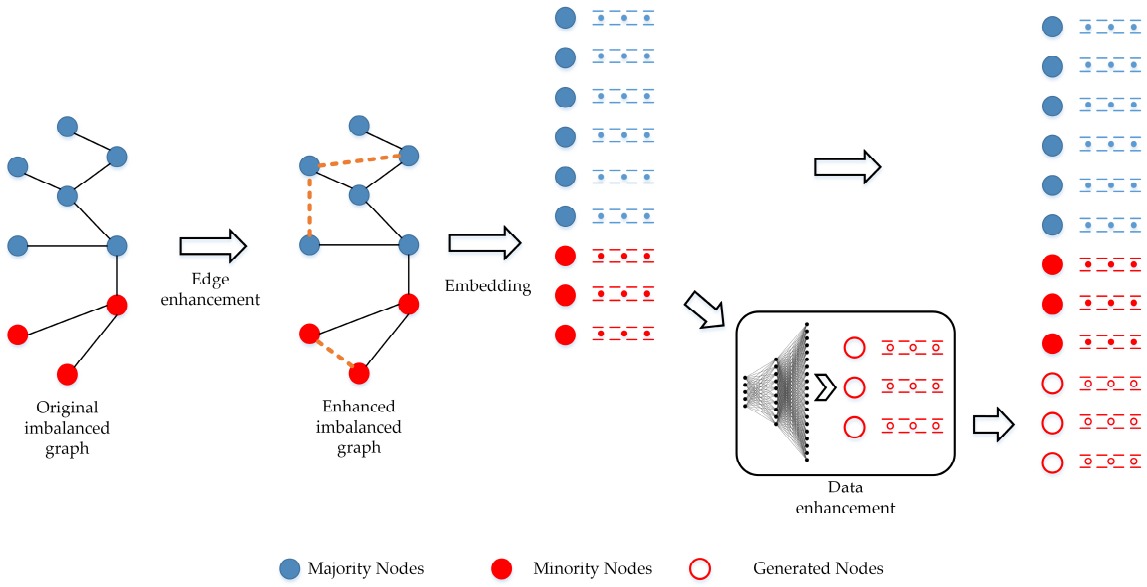


Figure 1. The overall structure of the ENE-GCN.

For all nodes in the graph, we first calculate their two-by-two similarity by cosine similarity:

$$\cos(a, b) = \frac{\sum_{i=1}^{d}(a_i \times b_i)}{\sqrt{\sum_{i=1}^{d}(a_i)^2} \times \sqrt{\sum_{i=1}^{d}(b_i)^2}}, \quad (1)$$

where a, b denote any two different nodes, and d denotes the node feature dimension; we take the part of the results with the highest similarity among them as the edges to be enhanced.

Here, a hyperparameter k is introduced, representing the percentage of the enhanced edges to the original edges, and these enhanced edges form the set E:

$$E = \{(a,b) | \text{top}(k * N) \text{ in } \cos(a,b))\}, \quad (2)$$

where N is the number of nodes and the adjacency matrix formed by the set E is:

$$A'(a,\ b) = \begin{cases} 1, (a,b) \in E \\ 0, (a,b) \notin E. \end{cases} \tag{3}$$

The enhanced adjacency matrix is obtained by integrating it into the adjacency matrix of the original graph as follows:

$$A_{en} = A + A', \tag{4}$$

where A is the original graph adjacency matrix, and A_{en} denotes the adjacency matrix after edge enhancement. Subsequently, semi-supervised embedding is performed on the enhanced graph using two-layer GCN, and the activation function uses ReLU to get the embedding result of each node:

$$X = \mathrm{ReLU}\left(\hat{A}_{en}\ \mathrm{ReLU}\left(\hat{A}_{en}\ XW^{(0)}\right)W^{(1)}\right), \tag{5}$$

where, $\hat{A}_{en}$ represents the normalized adjacency matrix of A_{en}.

After obtaining the embedding results, X is decomposed into X_{maj} and X_{min}, which contain majority and minority class samples, respectively. For X_{min}, a generative adversarial network (GAN) is used to augment its data volume, and both the generator and the discriminator employ multi-layer neural networks. The objective function of the GAN model is defined as:

$$\min_G \max_D V(D,G) = E_{x\sim P_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{6}$$

where x represents real minority class data from X_{min}, and z is a random noise variable. The generator generates a targeted number of virtual minority class data X'_{min}, such that $X_{min} + X'_{min}$ achieves a balance with X_{maj}. Finally, the balanced dataset is classified by a multi-layer fully connected neural network:

$$Y_{pred} = \mathrm{MLP}\left(X_{min} + X'_{min} + X_{maj}\right). \tag{7}$$

Table 1 shows the pseudo code of the ENE-GCN, which is divided into four phases, namely the edge enhancement phase, the embedding phase, the data generation phase, and the classification phase. In the edge enhancement phase, we need to calculate the similarity between nodes and merge edges with higher node similarity into the original dataset. In the embedding phase, we train an embedding model using nodes with known labels in the training set. We then use this embedding model to embed all nodes and obtain the embedding results for all nodes.

In the adversarial generation phase, we only need to manipulate the embedding results of the minority class in the training set. This process only requires manipulating the vectors, so the generator and discriminator can be designed simply, such as a fully connected network. After training is completed, the generator is used to generate the required number of virtual minority class data to balance the data for the classifier in the next step. After completing the above steps, the dataset is processed into a normalized, data-balanced dataset, so that most conventional machine learning models are competent for the classification.

Table 1. Pseudo code of the ENE-GCN.

Input: Training graph data: $G(A, X, Y)$; M: Mask of the nodes, $M_{train} : M_{validate} : M_{test} = 7 : 1 : 2$ L: Label of the nodes, $L_{maj} \gg L_{min}$ *k*: Enhancement coefficient to set the number of enhanced edges G: Generator for generating virtual minority class node embedding D: Discriminator for determining the results of the generator MLP: neural network for classifying node embedding results Output: Classification results for nodes in the test set
Edge enhancement: 1. $\cos(a, b) = \frac{\sum_{i=1}^{d}(a_i \times b_i)}{\sqrt{\sum_{i=1}^{d}(a_i)^2} \times \sqrt{\sum_{i=1}^{d}(b_i)^2}}$ 2. $E = \{(a, b) \mid \text{top } (k * N) \text{ incos}(a, b))\}$ 3. $A'(a,\ b) = \begin{cases} 1, (a,b) \in E \\ 0, (a,b) \notin E. \end{cases}$ 4. $A_{en} = A + A'$
Node embedding: 5. for each epoch: 6. $\quad X = \text{ReLU}\left(\hat{A}_{en}\ \text{ReLU}\left(\hat{A}_{en}\ XW^{(0)}\right)W^{(1)}\right)$ 7. $\quad \text{Loss} = -\sum_{i \in M_{train}} \sum_{min}^{maj} Y_i \ln X_i$ 8. $\quad$ update GCN parameters W by minimizing Loss; 9. end for
Generate virtual node embedding: 10. for each epoch: $//X \in M_{train}$ and $X \in L_{min}$ 11. $\quad Loss_D = \sum[\log D(X) + \log(1 - D(G(Z)))]$ 12. $\quad$ update D 13. $\quad Loss_G = \sum[\log(D(G(Z)))]$ 14. $\quad$ update G 15. end for 16. $X'_{min} = G(Z)$
Classification: 17. $Y_{pred} = \text{MLP}(X_{min} + X'_{min} + X_{maj})$

3.3. Time Complexity

The time complexity of the proposed method is as follows. The complexity of the similarity computation part is $O(d\,n^2)$, where d represents the dimension of the node features. The time complexity of the GCN embedding step is $O(K|E|d_G + Knd_G{}^2)$. Here, K represents the number of GCN layers, $|E|$ denotes the number of edges, and d_G signifies the dimensionality of the GCN hidden layers. The time complexity of updating either the generator or the discriminator is $O((L-1)n_g H^2 + n_g n_{min}^2)$, where L denotes the number of fully connected layers, n_g represents the number of samples to be generated, n_{min} indicates the number of minority samples, and H signifies the dimensionality of the hidden layer in both the generator and discriminator. The final complexity of the fully connected classifier is of the same order of magnitude as the generator and is no longer considered separately. Therefore, the total complexity of ENE is $O(dn^2 + K|E|d_G + Knd_G^2 + (L-1)n_g H^2 + n_g n_{min}^2)$.

Compared to a GCN, the additional time cost of this method is only $O(d\,n^2 + (L-1)n_g H^2 + n_g n_{min}^2)$, i.e., to find the node similarity and the training part of the GAN. The GAN part is after the node embedding in our design, so its node feature dimension has been significantly reduced, i.e., H is already much more minor than d.

Therefore, it can be argued that the main increase in the time cost of this method compared to a GCN is only in the $O(dn^2)$ step of the node similarity extraction.

It is important to highlight that while the complexity analysis suggests that the increment is only $O(dn^2)$, a repeated search process is still necessary to determine the value of k. However, since the increase in computational load is linear, the theoretical complexity analysis does not fully reflect the actual computation time required. Therefore, when using this algorithm, it is necessary to weigh the value of the task against the available computing power.

4. Experiment

4.1. Experimental Setup

4.1.1. Experimental Environment

The test environment used a single machine equipped with an Intel i7 7700K at 4.50 GHz, an Nvidia RTX 3080 with 8704 CUDA cores, and 32 GB of DDR4 RAM.

4.1.2. Datasets

Four datasets, Cora [38], CiteSeer [1], PubMed [39], and Coauthor-Physics [40], were used as test datasets, all of which are publicly available and are real-world datasets. The first three datasets belong to the Planetoid paper citation graph, which is widely used in research in the field of node classification. Coauthor-Physics is an academic network with co-authorship relationships based on Microsoft Academic Graphs. It is also widely used in experiments related to the field of graph networks. This dataset has the same structure as the previous three datasets and has, on average, significantly more connections per node than the previous three, which helped us to observe how the algorithms perform on different datasets. Table 2 summarizes the statistical information for these datasets.

Table 2. The statistical information of the network datasets.

Datasets	Cora	Citeseer	PubMed	Coauthor-Physics
Number of nodes	2708	3327	19,717	34,493
Number of edges	5278	4552	44,324	247,962
Edges/Nodes	1.95	1.37	2.25	7.19
Number of classes	7	6	3	5
Feature dimension	1433	3703	500	8415
Ratio of the minority class	6.65%	7.94%	20.8%	7.98%

Cora, CiteSeer, and PubMed are all citation network datasets, which are undirected graphs. Articles are represented by nodes, and citation relationships are represented by edges. These datasets take the bag-of-words in articles as the feature vector, and the research direction as categories. The nodes of the Coauthor-Physics dataset are the authors, the node feature vectors are the paper's keywords, the edges represent the authors' collaborations on the paper, and the node categories are the authors' research fields.

In our experiments, to verify our method's effectiveness on imbalanced networks, we followed the method of the literature [6] and reconstructed four graphs as binary imbalanced graphs. The minority class represents the class with the smallest sample size in the original dataset, and the rest are all majority classes. For example, the Cora dataset contains seven classes: Case_Based (11.0%), Genetic_Algorithms (15.4%), Neural_Networks (30.2%), Probabilistic_Methods (15.7%), Reinforcement_Learning (8.0%), Rule_Learning (6.6%), and Theory (13.0%). Among them, Rule Learning accounts for 6.6% and is the smallest class, so it is regarded as the minority class, and the rest are regarded as the majority class. We randomly split each dataset into a training set, a validation set, and a test

set in the ratio of 7:1:2. It is worth noting that only one split was performed for each dataset, and all subsequent experiments and validations were performed based on the same data split results.

4.1.3. Comparison Methods and Parameter Settings

We chose seven methods for comparison, including GCN [41], GraphSAGE [42], GAT [19], GraphSMOTE [14], DR-GCN [7], RA-GCN [13], and BNE [15]. Among these methods, the first three are very classic and commonly used node classification methods, and the last four are advanced node classification methods after considering the imbalance.

The GCN is the most representative node classification method, and its node embedding results aggregate the features of itself and all neighboring nodes. GraphSAGE, on the other hand, aggregates only some features of adjacent nodes. The GAT aggregates the features of neighboring nodes onto the central vertex using attention coefficients.

The GraphSMOTE addresses the class imbalance problem by employing the Synthetic Minority Oversampling Technique (SMOTE), which generates synthetic samples from existing minority samples. The DR-GCN is also a GCN-based method for addressing imbalanced network embedding. It uses distribution alignment training to deal with data imbalances and introduces conditional adversarial training to enhance the separation of different classes. The RA-GCN trains a classifier and a weighting network using adversarial methods to ensure that the classifier pays more attention to important samples. BNE introduces the idea of balanced neighborhood exploration by assigning multiple labels to nodes connecting different classes, thereby mitigating imbalance.

The codes used in the experiments are all from the original authors. For the GCN and GAT, the order of the aggregate neighborhood is $k = 2$. For GraphSAGE, following the authors' recommendations, we set $\Delta = 2$, $S_1 = 5$, $S_2 = 5$. For GraphSMOTE, the minority class samples were generated to balance the two classes of data. For DR-GCN, we used 50% of the validation set for hyperparameter optimization, for RA-GCN, the same as for the GCN, we set $k = 2$, and the weighted network also employed a two-layer GCN. For BNE, according to the author's suggestion, the initial learning rate, exploration coefficient, and weight decay were respectively set to $\{10^{-3},\ 1.5,\ 10^{-4}\}$. To avoid overfitting, the dropout parameter for all methods was set to 0.5 by convention.

The proposed method's edge enhancement component includes only similarity computations without any parameter. The embedding step uses two layers of the GCN and uses ReLU as the activation function. The hidden layer dimension is set to 256, and the output dimension is set to 8. The dropout parameter is also set to 0.5.

As for the data generation part, the generator and the discriminator all use three-layer fully connected networks with structures of 32-128-8 and 8-128-1, respectively. The activation functions utilized are ReLU, and the chosen optimizer is Adam. The update ratio between the generator and the discriminator is set to 1:10. The final classification model also adopts a fully connected network structure of 8-8-2.

The learning rates for both the generator and the discriminator are set to 0.0001. Training is terminated prematurely when the discriminator loss approaches zero for 20 consecutive epochs. The architecture of all experimental methods is the same for all datasets.

4.1.4. Evaluation Metrics

The experiment uses Recall, Precision, and Area Under Curve (AUC) as evaluation metrics. Recall refers to the probability that an actual positive sample is predicted correctly. Precision is the percentage of all samples predicted to be positive that are actually positive. AUC is the area under the ROC (Receiver Operating Characteristic) curve, which is enclosed by the axes; the closer its value is to 1.0, the higher the validity of the method.

In the data imbalance classification scenario where more importance is attached to positive cases, these three indicators can objectively evaluate the performance of the algorithm.

4.2. Classification Results and Ablation Experiments

4.2.1. Classification Results

We repeated each experiment 10 times, taking the average as the final result. The data partitioning results presented in Section 4.1.2 are consistently applied across all experiments, thereby minimizing the potential impact of variations in data partitioning on the algorithm's performance. The results of the experiments on the four datasets are shown in Table 3, where the best results are highlighted in bold.

Table 3. The imbalanced binary node classification results on Cora, CiteSeer, PubMed, and Coauthor-Physics datasets.

Datasets	**Cora**			**Citeseer**			**PubMed**			**Coauthor-Physics**		
Metrics	**Recall**	**Precision**	**AUC**	**Recall**	**Precision**	**AUC**	**Recall**	**Precision**	**AUC**	**Recall**	**Precision**	**AUC**
GCN	0.7000	0.9333	0.8478	0.1833	0.8462	0.5901	0.8283	0.8127	0.8885	0.9187	0.9669	0.9580
GraphSAGE	0.6750	0.9000	0.8343	0.3000	0.6429	0.6419	0.8595	0.8524	0.9098	0.9261	0.9489	0.9609
GAT	0.8250	0.9428	0.9103	0.2667	0.8000	0.6301	0.8139	0.8071	0.8808	0.9113	0.9390	0.9531
G-SMOTE	0.7500	0.8823	0.8707	0.3500	0.7500	0.6694	0.8703	0.8410	0.9130	0.9224	0.9469	0.9590
DR-GCN	0.8750	0.8974	0.9332	0.6833	0.4308	0.7035	0.8980	0.8184	0.9222	0.9445	0.9291	0.9692
RA-GCN	0.8500	0.8947	0.9206	0.5806	0.5070	0.7621	0.8919	0.8219	0.9200	0.9409	0.9305	0.9674
BNE	0.9000	0.8571	0.9435	0.7000	0.4576	0.7366	0.9112	0.8083	0.9265	0.9353	0.9388	0.9651
+Edges	0.9000	0.9000	0.9457	0.4500	0.5000	0.7033	0.8727	0.8366	0.9135	0.9224	0.9469	0.9590
+Nodes	0.8750	0.8750	0.9321	0.5667	0.3579	0.7342	0.8583	0.8573	0.9100	0.9279	0.9472	0.9618
Our method	0.9250	0.9024	0.9582	0.7333	0.4800	0.7757	0.9087	0.8439	0.9318	0.9482	0.9310	0.9711

The best results are highlighted using shadows.

It can be seen that on the Cora, CiteSeer, and Coauthor-Physics datasets, both the Recall and the AUC of our method are ahead of other algorithms. On the PubMed dataset, the Recall of this paper's method is slightly lower than that of BNE, but its AUC is higher. In many binary classification problems (e.g., abnormal transaction detection and rare disease prediction), Recall is usually more important than Precision. Therefore, in this paper, the experimental results can verify the method's effectiveness.

In most cases, methods that use balancing techniques (i.e., GraphSMOTE, DR-GCN, RA-GCN, and BNE) outperform methods that do not use such techniques (i.e., GCN, GraphSAGE, and GAT) in terms of Recall and AUC metrics. However, when it comes to Precision, methods without balancing techniques often perform better than those with balancing techniques. This is consistent with expectations, as methods without balancing techniques tend to classify fewer nodes as positive, leading to higher Precision but lower Recall and AUC for such algorithms.

Meanwhile, it can be noticed that the proposed method demonstrates a more pronounced effectiveness on sparser datasets. This can be illustrated through a comparison with the BNE method. On the CiteSeer dataset (Edges/Nodes = 1.37), our method achieved improvements of 0.03 in Recall, 0.02 in accuracy, and 0.04 in AUC. In contrast, on the Coauthor-Physics dataset (Edges/Nodes = 7.19), the increases in Recall and AUC were only 0.01 and 0.01, respectively, while the accuracy even decreased by 0.01.

In addition, we conducted paired sample *t*-tests on the results of 10 experiments to test the difference between the method of this paper and other methods. The results show that the improvement measures have achieved remarkable results. Table 4 shows the *t*-test *p*-values between the improved method and the baseline methods. In the t-tests comparing our method against BNE, the *p*-values for Recall, Precision, and AUC on the Cora dataset were 5.67×10^{-3}, 2.20×10^{-5}, and 4.90×10^{-3}, respectively Similarly, the *p*-values for Recall, Precision, and AUC on the CiteSeer dataset were 3.28×10^{-4}, 1.44×10^{-3}, and 1.14×10^{-5}, respectively. All these tests rejected the null hypothesis at the significance

level of $\alpha = 0.01$. The above results indicate that the ENE-GCN had advantages over other methods in terms of edge enhancement to strengthen the category edges and data generation to balance the minority class nodes.

Table 4. *p*-values of *t*-tests between the ENE-GCN and baseline methods.

Datasets	Cora			Citeseer			PubMed			Coauthor-Physics		
Metrics	**Recall**	**Precision**	**AUC**	**Recall**	**Precision**	**AUC**	**Recall**	**Precision**	**AUC**	**Recall**	**Precision**	**AUC**
GCN	6.80×10^{-11}	2.03×10^{-3}	4.24×10^{-9}	1.00×10^{-15}	3.41×10^{-13}	1.06×10^{-10}	9.56×10^{-8}	1.21×10^{-3}	3.65×10^{-5}	5.74×10^{-7}	6.67×10^{-7}	2.36×10^{-6}
GraphSAGE	1.66×10^{-12}	2.48×10^{-1}	9.42×10^{-10}	8.27×10^{-16}	7.18×10^{-11}	6.20×10^{-10}	4.51×10^{-6}	1.89×10^{-1}	2.32×10^{-3}	8.27×10^{-5}	6.30×10^{-6}	7.43×10^{-6}
GAT	8.98×10^{-8}	4.94×10^{-4}	8.68×10^{-6}	1.24×10^{-14}	1.89×10^{-12}	1.19×10^{-9}	1.38×10^{-8}	2.79×10^{-5}	4.07×10^{-7}	3.82×10^{-7}	6.17×10^{-2}	3.46×10^{-6}
G-SMOTE	3.61×10^{-11}	8.25×10^{-6}	1.28×10^{-8}	1.63×10^{-14}	6.52×10^{-13}	2.25×10^{-10}	7.88×10^{-6}	6.74×10^{-1}	2.23×10^{-4}	7.73×10^{-7}	1.14×10^{-3}	6.21×10^{-8}
DR-GCN	3.01×10^{-6}	5.24×10^{-3}	4.10×10^{-4}	2.15×10^{-7}	2.90×10^{-6}	1.56×10^{-7}	3.55×10^{-3}	3.99×10^{-5}	2.56×10^{-2}	1.22×10^{-2}	3.26×10^{-2}	3.63×10^{-3}
RA-GCN	3.37×10^{-7}	5.47×10^{-2}	4.33×10^{-6}	1.76×10^{-10}	5.47×10^{-3}	6.80×10^{-2}	2.17×10^{-3}	3.21×10^{-3}	5.73×10^{-2}	1.50×10^{-2}	4.83×10^{-1}	1.98×10^{-2}
BNE	5.67×10^{-3}	2.20×10^{-5}	4.90×10^{-3}	3.28×10^{-4}	1.44×10^{-3}	1.14×10^{-5}	7.27×10^{-1}	4.24×10^{-6}	1.11×10^{-2}	5.66×10^{-3}	1.72×10^{-2}	1.22×10^{-2}

4.2.2. Ablation Experiments

In addition, we performed ablation experiments. These included (1) edge enhancement without node enhancement (+Edges) and (2) node enhancement without edge enhancement (+Nodes). The results of the experiments are shown in two rows (+Edges, +Nodes) in Table 3. It can be seen that both methods are significantly improved compared to the baseline methods that do not use a balancing metric. Compared to the baseline methods using the balanced metric, both methods are not inferior. Furthermore, compared to the complete algorithm, both methods perform slightly less well. This indicates that both steps help to improve the data imbalance problem.

4.3. Edge Enhancement Results

To investigate how edge enhancement affects the overall graph structure, we conducted experiments using the Cora dataset and divided the edges into three categories: majority-majority (0-0), majority-minority (0-1), and minority-minority (1-1). In the original graph, the number of edges in these three categories is 5021, 153, and 255, respectively, and the number of majority class nodes and minority class nodes is 2528 and 180. We varied the parameter k from 0 to 100, indicating the number of enhanced edges from 0 to 5429.

The number of edges added to the original graph in the three categories is shown in Figure 2. The number of edges in all three categories increases linearly with the variation of k.

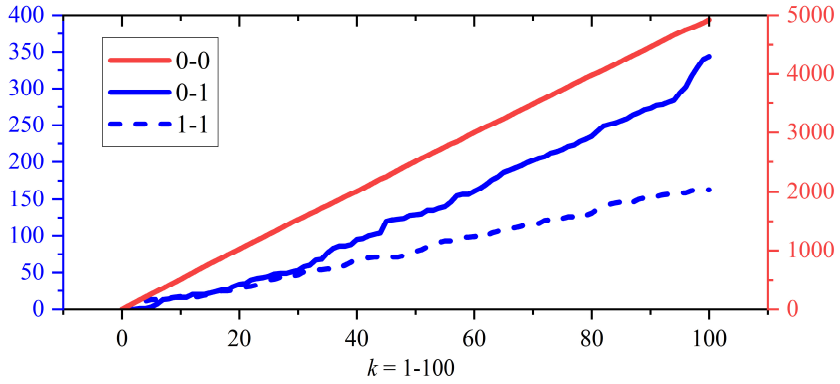
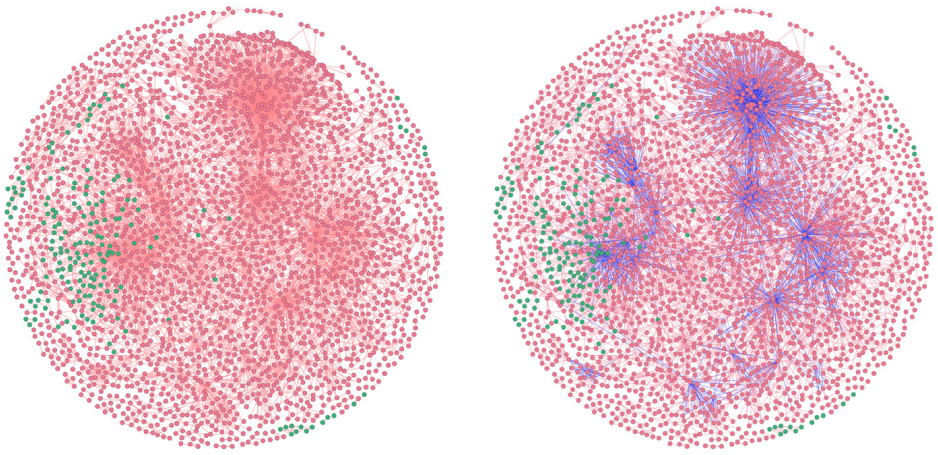


Figure 2. The number of enhanced edges for different categories (0-0 represents majority-majority, 0-1 represents majority-minority, and 1-1 represents minority-minority) when k varied from 1 to 100. The red line indicates the amount using the right axis, while the blue line represents the amount using the left axis.

The experiments showed that the classifier performs best when $k = 49$. At this time, the enhanced edges for the three categories are 2410, 124, and 73, respectively and are not

concentrated in one category. This suggests that the improvement in classification results is due to the fact that more edges improve the representation and tuning capabilities of the embedded network, and not just because the enhanced edges improve the clustering.

Figure 3 shows the visualization of the Cora dataset before and after edge enhancement. The left figure shows the original imbalanced network, while the right figure shows the network after edge enhancement. It shows that the nodes in the majority class become tighter after edge enhancement, resulting in clearer classification boundaries.



(**a**) original imbalanced network (**b**) after edge enhancement

Figure 3. Visual comparison of the Cora dataset before and after edge enhancement: (**a**) shows the original imbalanced network; (**b**) shows the network after edge enhancement. Solid green and red circles represent minority and majority nodes. Red lines represent real edges, while blue lines represent enhanced edges.

4.4. Parameter Sensitivity Analysis

This section discusses the effect of varying the coefficient k on the algorithm's performance, i.e., how many edges should be enhanced in the original dataset. We explore the continuous range of k values from 0 to 100, where $k = 0$ implies no enhancement of edges on the original dataset, and $k = 100$ implies an enhancement of 100% of the original edge count. The algorithm's AUC performance is evaluated on four different datasets. For each value of k, we performed 10 repeated experiments and calculated the average AUC, as shown in Figure 4.

On the Cora dataset, when k varied from 0 to 30, the AUC did not improve significantly and decreased slightly between 20 and 30. However, when k was set between 30 and 50, there was a significant improvement in the algorithm's AUC performance, with the highest value observed at $k = 49$. After $k = 50$, the AUC performance of the algorithm gradually decreased and eventually fell below that of the original dataset. This suggests that there is an optimal range for the number of enhanced edges and that more is not necessarily better.

For the CiteSeer dataset, the AUC performance initially decreased, then fluctuated up and exceeded the performance of the original dataset. Even at $k = 100$, there was still an upward trend, suggesting that the optimal range for k had not yet been reached and could probably be improved further. In contrast, the AUC of the PubMed and Coauthor-Physics datasets generally increased in the range of k from 0 to 30 and fluctuated in the range of k from 31 to 100, but there was no significant upward or downward trend.

The performance of the ENE-GCN varied in each dataset, but the general trend was from improving to stabilizing to decreasing. This may be related to the average number of connections per node. For example, in the Cora and PubMed datasets, with an average of 1.95 and 2.25 edges per node, respectively, the AUC reached stable and good results

around $k = 40$. However, in the CiteSeer and Coauthor-Physics datasets, each node has 1.37 and 7.19 edges on average, respectively. The former did not reach stable results even at $k = 100$, while the latter reached an optimal AUC after $k = 25$. Due to the variety of real-world datasets, the value of k should be determined by a grid search when using our method for real-world imbalanced graph datasets.

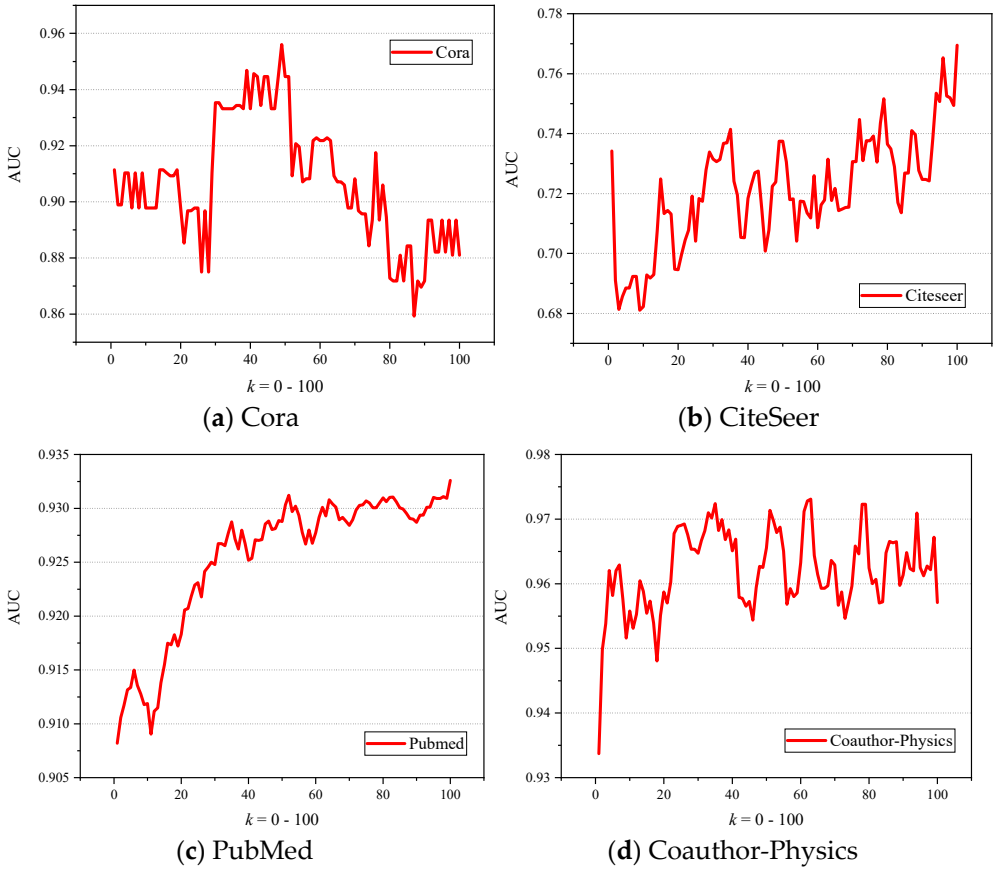


Figure 4. The AUC of the classifiers on the four datasets when k varied from 0 to 100. (**a**) AUC on Cora dataset; (**b**) AUC on CiteSeer dataset; (**c**) AUC on PubMed dataset; (**d**) AUC on Coauthor-Physics dataset.

5. Conclusions

In this paper, we proposed the ENE-GCN for the imbalanced node binary classification problem, combining edge enhancement and data enhancement. Edge enhancement exploited node similarity to construct new edges to enhance the original graph topology and optimize the node embedding results. Data enhancement achieved the balance of training scenarios by using the embedding results of minority class to generate virtual minority class nodes. We conducted a series of experiments to verify the effectiveness of the proposed method, including imbalanced node classification, ablation experiments, edge enhancement visualization, and hyperparameter sensitivity analysis. Node classification experiments showed that the Recall and AUC of our method outperformed other algorithms in most cases, especially when the dataset's Edges/Nodes were small. In addition, visualization experiments showed that the method better aggregates similar nodes, more clearly delineates decision boundaries, and expands the space of fitted node embeddings.

While we have validated the effectiveness of the ENE-GCN in certain situations, it still has some limitations. Firstly, it is an interesting topic worth exploring how to determine the enhanced edges more effectively. In this paper, we chose node similarity because it performs better on the Cora dataset than link prediction. However, the universality of

this conclusion and the underlying mechanisms remain to be explored. In addition, the selection process of the k parameter is essentially a grid search process. Although this method ultimately leads to improved classification results, it also results in a significant increase in computational workload. Therefore, when using this algorithm, it is necessary to weigh the value of the task against the available computing power. Technically, introducing Bayesian optimization is also a solution worth trying. We are convinced that solving the imbalance problem on graphs is still an active research area with great development potential and wide application prospects.

Author Contributions: Conceptualization, J.T.; data curation, J.T.; funding acquisition, D.L.; software, J.T.; supervision, D.L.; visualization, J.T.; writing–original draft, J.T. and J.L.; writing–review & editing, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the Science and Technology Development Fund (FDCT) of Macau under Grant 0095/2023/RIA2.

Data Availability Statement: The source code is available at https://github.com/tianjiadong2/Edge-Enhanced-embedding (accessed on 20 February 2025). The dataset used for this study is publicly available. The Cora, CiteSeer, PubMed, and Coauthor-Physics datasets can be downloaded at https://github.com/shchur/gnn-benchmark#datasets, accessed on 20 February 2025.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wu, L.; Zhang, S.; Chen, W.; Hao, X. Identification and Correction of Abnormal Measurement Data in Power System Based on Graph Convolutional Network and Gated Recurrent Unit. *Electr. Power Syst. Res.* **2023**, *224*, 109740. [CrossRef]
2. Xiao, L.; Sun, L.; Ling, M.; Peng, Y. A Survey of Graph Neural Network Based Recommendation in Social Networks. *Neurocomputing* **2023**, *549*, 126441. [CrossRef]
3. Lee, C.G.; Bollacker, K.; Lawrence, S. CiteSeer: An Automatic Citation Indexing System. In Proceedings of the DL '98: Proceedings of the Third ACM Conference on Digital Libraries, Pittsburgh, PA, USA, 23–26 June 1998. [CrossRef]
4. He, H.; Garcia, E.A. Learning from Imbalanced Data. *IEEE Trans. Knowl. Data Eng.* **2009**, *21*, 1263–1284. [CrossRef]
5. Chen, D.; Lin, Y.; Zhao, G.; Ren, X.; Li, P.; Zhou, J.; Sun, X. Topology-Imbalance Learning for Semi-Supervised Node Classification. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online, 6–14 December 2021; Volume 34.
6. Qu, L.; Zhu, H.; Zheng, R.; Shi, Y.; Yin, H. ImGAGN:Imbalanced Network Embedding via Generative Adversarial Graph Networks. *arXiv* **2021**, arXiv:2106.02817.
7. Shi, M.; Tang, Y.; Zhu, X.; Wilson, D.; Liu, J. Multi-Class Imbalanced Graph Convolutional Network Learning. In Proceedings of the Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence Main Track, Yokohama, Japan, 7–15 January 2020. [CrossRef]
8. Wu, L.; Lin, H.; Gao, Z.; Tan, C.; Li, S.Z. GraphMixup: Improving Class-Imbalanced Node Classification on Graphs by Self-Supervised Context Prediction. *arXiv* **2021**, arXiv:2106.11133.
9. Pei, H.; Wei, B.; Chang, K.C.-C.; Lei, Y.; Yang, B. Geom-GCN: Geometric Graph Convolutional Networks. In Proceedings of the 8th International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020. [CrossRef]
10. Jin, D.; Yu, Z.; Huo, C.; Wang, R.; Wang, X.; He, D.; Han, J. Universal Graph Convolutional Networks. In Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Online, 6–14 December 2021; Volume 34.
11. Enxhell, L.; Day, B.; Lio, P. Clique Pooling for Graph Classification. *arXiv* **2019**, arXiv:1904.00374.
12. Molaei, S.; Bousejin, N.G.; Zare, H.; Jalili, M.; Pan, S. Learning Graph Representations with Maximal Cliques. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 1089–1096. [CrossRef]
13. Ghorbani, M.; Kazi, A.; Soleymani Baghshah, M.; Rabiee, H.R.; Navab, N. RA-GCN: Graph Convolutional Network for Disease Prediction Problems with Imbalanced Data. *Med. Image Anal.* **2022**, *75*, 102272. [CrossRef]
14. Zhao, T.; Zhang, X.; Wang, S. GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. In Proceedings of the WSDM '21: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, Online, 8–12 March 2021; pp. 833–841. [CrossRef]
15. Zhu, Z.; Xing, H.; Xu, Y. Balanced Neighbor Exploration for Semi-Supervised Node Classification on Imbalanced Graph Data. *Inf. Sci.* **2023**, *631*, 31–44. [CrossRef]

16. Franceschi, L.; Niepert, M.; Pontil, M.; He, X. Learning Discrete Structures for Graph Neural Networks. In Proceedings of the The Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2019; Volume 97, pp. 1972–1982. [CrossRef]
17. Chen, Y.; Wu, L.; Zaki, M.J. Deep Iterative and Adaptive Learning for Graph Neural Networks. *arXiv* **2019**, arXiv:1912.07832. [CrossRef]
18. Park, J.; Yoo, S.; Park, J.; Kim, H.J. Deformable Graph Convolutional Networks. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 7949–7956. [CrossRef]
19. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. *Int. Conf. Learn. Represent.* **2017**, arXiv:1710.10903.
20. Brody, S.; Alon, U.; Yahav, E. How Attentive Are Graph Attention Networks? In Proceedings of the Tenth International Conference on Learning Representations, Virtual, 25–29 April 2021. [CrossRef]
21. Javaloy, A.; Sanchez-Martin, P.; Levi, A.; Valera, I. Learnable Graph Convolutional Attention Networks. *Int. Conf. Learn. Represent.* **2022**, arXiv:2211.11853. [CrossRef]
22. Duan, W.; Xuan, J.; Qiao, M.; Lu, J. Learning from the Dark: Boosting Graph Convolutional Neural Networks with Diverse Negative Samples. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 6550–6558. [CrossRef]
23. Duan, W.; Xuan, J.; Qiao, M.; Lu, J. Graph Convolutional Neural Networks with Diverse Negative Samples via Decomposed Determinant Point Processes. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 18160–18171. [CrossRef] [PubMed]
24. Bo, D.; Wang, X.; Shi, C.; Shen, H. Beyond Low-Frequency Information in Graph Convolutional Networks. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 3950–3957. [CrossRef]
25. Zhang, A.; Huang, J.; Li, P.; Zhang, K. Building Shortcuts between Distant Nodes with Biaffine Mapping for Graph Convolutional Networks. *ACM Trans. Knowl. Discov. Data* **2024**, *18*, 1–21. [CrossRef]
26. Wang, Y.; Wen, J.; Zhang, C.; Xiang, S. Graph Aggregate-Repel Network: Do Not Trust All Neighbors in Heterophilic Graphs. *Neural Netw.* **2024**, *178*, 106484. [CrossRef]
27. Abu-El-Haija, S.; Perozzi, B.; Kapoor, A.; Harutyunyan, H.; Alipourfard, N.; Lerman, K.; Steeg, G.V.; Galstyan, A. MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing. *Int. Conf. Mach. Learn.* **2019**, 21–29. [CrossRef]
28. Gong, K.; Song, X.; Li, W.; Wang, S. HN-GCCF: High-Order Neighbor-Enhanced Graph Convolutional Collaborative Filtering. *Knowl. Based Syst.* **2024**, *283*, 111122. [CrossRef]
29. He, L.; Bai, L.; Yang, X.; Liang, Z.; Liang, J. Exploring the Role of Edge Distribution in Graph Convolutional Networks. *Neural Netw.* **2023**, *168*, 459–470. [CrossRef]
30. Page, L.; Brin, S.; Motwani, R.; Winograd, T. The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Libraries Working Paper. 1999. Available online: http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf (accessed on 20 February 2025).
31. Chien, E.; Peng, J.; Li, P.; Milenkovic, O. Adaptive Universal Generalized PageRank Graph Neural Network. In Proceedings of the 9th International Conference on Learning Representations, Virtual Event, Austria, 3–7 May 2021. [CrossRef]
32. Lee, M.; Kim, S.B. HAPGNN: Hop-Wise Attentive PageRank-Based Graph Neural Network. *Inf. Sci.* **2022**, *613*, 435–452. [CrossRef]
33. Laishui, L.; Zhang, T.; Hu, P.; Bardou, D.; Dalal, S.; Zheng, Z.; Yu, G.; Wu, H. An Improved Gravity Centrality for Finding Important Nodes in Multi-Layer Networks Based on Multi-PageRank. *Expert Syst. Appl.* **2024**, *238*, 122171. [CrossRef]
34. Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.; Jegelka, S. Representation Learning on Graphs with Jumping Knowledge Networks. *Int. Conf. Mach. Learn.* **2018**, 5449–5458. [CrossRef]
35. Li, L.; Yang, W.; Bai, S.; Ma, Z. KNN-GNN: A Powerful Graph Neural Network Enhanced by Aggregating K-Nearest Neighbors in Common Subspace. *Expert Syst. Appl.* **2024**, *253*, 124217. [CrossRef]
36. Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; Koutra, D. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Adv. Neural Inf. Process. Syst. NeurIPS* **2020**, *33*, 7793–7804. [CrossRef]
37. Li, X.; Zhu, R.; Cheng, Y.; Shan, C.; Luo, S.; Li, D.; Qian, W. Finding Global Homophily in Graph Neural Networks When Meeting Heterophily. In Proceedings of the 39th International Conference on Machine Learning, PMLR 2022, Baltimore, MA, USA, 17–23 July 2022; Volume 162, pp. 13242–13256. [CrossRef]
38. McCallum, A.K.; Nigam, K.; Rennie, J.; Seymore, K. Automating the Construction of Internet Portals with Machine Learning. *Inf. Retr.* **2000**, *3*, 127–163. [CrossRef]
39. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; Eliassi-Rad, T. Collective Classification in Network Data. *AI Mag.* **2008**, *29*, 93. [CrossRef]
40. Shchur, O.; Mumme, M.; Bojchevski, A.; Günnemann, S. Pitfalls of Graph Neural Network Evaluation. 2019. Available online: https://arxiv.org/abs/1811.05868 (accessed on 20 February 2025).

41. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017; pp. 1–14. [CrossRef]
42. Hamilton, W.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the NIPS '17: Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, California, USA, 4–9 December 2017; pp. 1025–1035. [CrossRef]