electronics

MDPI

Article

Meta-Hybrid: Integrate Meta-Learning to Enhance Class Imbalance Graph Learning

Liming Ran, Hongyu Sun *, Lanqi Gao, Yanhua Dong * and Yang Lu

College of Mathematics and Computer, Jilin Normal University, Siping 136000, China; ranliming@mails.jlnu.edu.cn (L.R.); 223181040@mails.jlnu.edu.cn (L.G.); luyang@jlnu.edu.cn (Y.L.)
* Correspondence: hongyu@jlnu.edu.cn (H.S.); computerdyp@jlnu.edu.cn (Y.D.)

Abstract: The class imbalance problem is a significant challenge in node classification tasks. Since majority class samples dominate imbalanced data, the model tends to favor the majority class, resulting in insufficient ability to identify minority classes. Evaluation indicators such as accuracy may not fully reflect the model's performance. To solve these undesirable effects, we propose a framework for synthesizing minority class samples, GraphSHX, to balance the number of samples of different classes, and integrate the XGBoost model for node classification prediction during the training process. Conventional graph neural networks (GNNs) yielded unsatisfactory results, possibly due to the limited number of newly generated nodes. Therefore, we introduce a meta-mechanism to deal with small-sample problems, and employ the meta-learning approach to enhance performance on small-sample tasks by learning from a large number of tasks. An empirical evaluation of node classification on six publicly available datasets demonstrated that our balanced data set method outperforms existing optimal loss repair methods and synthetic node methods. The addition of the XGBoost model and meta-learning improves the accuracy by more than 5% to 10%, with the overall accuracy of the improved model being 15% higher than that of the baseline method.

Keywords: graph neural network; class imbalance node classification; ensemble learning

check for updates

Citation: Ran, L.; Sun, H.; Gao, L.; Dong, Y.; Lu, Y. Meta-Hybrid: Integrate Meta-Learning to Enhance Class Imbalance Graph Learning. *Electronics* **2024**, *13*, 3769. https://doi.org/10.3390/electronics13183769

Academic Editor: Aryya Gangopadhyay

Received: 11 July 2024
Revised: 13 September 2024
Accepted: 20 September 2024
Published: 22 September 2024

1. Introduction

Graph data is extensively utilized across various domains. Node classification represents a crucial task in the realm of graph data applications and constitutes a focal point within the domain of graph neural networks [1,2]. Nevertheless, prevailing research on this task often assumes class balance [3], which may lead to practical loopholes due to the prevalent nature of class imbalance in real-world datasets [4,5]. For instance, in a citation network [6], articles on machine learning far outnumber those on cryptography. In binary problems such as anomaly detection tasks encompassing malicious activities like brushing behavior, equipment failures, or credit card frauds, these instances typically constitute only a minute fraction of the overall dataset. Training directly on unbalanced class data tends to bias models towards dominant classes, resulting in overfitting or inadequate recognition of minority classes while evaluation metrics may not fully reflect model performance [3,6].

Addressing class imbalance has long been an enduring concern within the field of machine learning [3]. Two primary approaches are employed for mitigating this issue: one involves adjusting loss functions or sample weights to rectify biases and rebalance influence across different classes; the other entails synthetic sampling methods that alter dataset composition by generating additional samples. This paper focuses on synthetic sampling methods, because topological information can also be synthesized for graph data, and better performance can be obtained empirically by using the inherent properties of graph structure.

The class imbalance causes the minor subspace to be squeezed by the major subspace in the training process. Minor test samples are difficult to include in their correct latent subspaces, leading eventually to unsatisfactory classification performance during the inference phase [7]. The purpose of synthesizing samples is to enlarge the boundary of the minor class at the class decision boundary. However, simply synthesizing would inevitably violate the neighbor class subspace since the boundary of the minor class is shared with its neighbors, thus degrading the neighbor class and causing misclassification. Therefore, we propose a GraphSHX framework to enlarge the subspace of minor classes while avoiding the deterioration of neighboring ones. In addition, we integrate the XGBoost [8] model in the training process for node classification prediction. The XGBoost model can provide a node classification method different from the neural network model, supplementing the shortcomings of the neural network model, thereby improving the performance of the overall model. We incorporate meta-learning into GNNs. When dealing with class imbalance problems using standard GNNs, there is no inherent mechanism to specifically address the challenge of insufficient samples of secondary classes, which often results in low accuracy. However, meta-learning, with its suitability for few-shot learning, can mitigate the shortcomings of imbalanced data by improving the learning process for minority class samples. Meta-learning excels at extracting universal patterns and principles from a limited number of samples. This capability enables the model to discover valuable features within minority class samples and apply these learned insights to the overall classification task. This is crucial for handling imbalanced data, as it helps the model better differentiate between minority and majority classes, thereby enhancing the recognition and classification performance for minority classes.

2. Related Work

2.1. Class Imbalance Problem

The class imbalance problem is a common issue in various machine learning tasks. This problem arises when the number of samples in the major class significantly exceeds that in the minor class within the training set. Generally, there are two main approaches for handling this issue: one involves modifying the loss function to make the model pay more attention to the minority class, as discussed in [9–14]; the other involves altering the sample class distribution by either increasing the number of minority class samples or decreasing the number of majority class samples, as outlined in [15–19]. However, unlike other forms of data, directly applying synthetic sample methods to graph data is not feasible due to the inherent topological structure of graphs. Therefore, when synthesizing nodes, we also need to generate corresponding connecting edges.

2.1.1. Loss-Modifying Approaches

Cost-sensitive learning is one of the most representative methods for loss repair. Elkan et al. [9] proposed a cost-sensitive loss function that considers the different costs associated with misclassifying different classes. This approach uses a cost matrix to represent the misclassification costs. By assigning higher penalties to misclassifications of the minority class and lower penalties to those of the majority class, the overall error can be minimized. The design of such a loss function enables more emphasis on the classification accuracy of the minority class when training decision trees, thereby improving the overall performance of the classifier on imbalanced datasets. However, the design of the cost matrix often relies on domain knowledge and experience, which can be somewhat subjective. Without accurate prior knowledge, determining an appropriate cost matrix can be challenging, prompting many researchers to conduct more specific studies. Lin et al. [10] introduced focal loss, which dynamically adjusts the weight of each sample, making the model focus more on hard-to-classify samples. Focal loss introduces a modulation factor to the standard cross-entropy loss, assigning smaller weights to easily classified samples and thus reducing their impact on the total loss. Zhang et al. [11], proposed rebalanced focal loss based on focal loss to further correct the class imbalance problem by introducing adaptive balance factors. This factor can dynamically adjust the degree of attention to each class and improve the model's ability to recognize a few classes. Cui et al. [12] proposed a class-balanced loss function based on the effective number of samples. By calculating the effective number of samples for each class, this method determines the loss weights, further improving the traditional class weight methods and making the weighting of minority class samples in the loss calculation more reasonable. Liu et al. [13] proposed balanced cross-entropy loss, which combines cross-entropy loss and class balancing strategies to deal with class imbalances by adjusting class weights and loss functions. It can effectively improve the classification performance of a few categories of samples. Li et al. [14] introduced GHM Loss, which addresses the imbalance of positive and negative samples as well as the difficulty levels of samples by bucketing and weighting gradients. This loss function is specifically designed for single-stage object detection tasks, allowing the model to handle samples of varying difficulty levels in a more balanced way during training.

2.1.2. Generative Approaches

Upsampling is a common and straightforward method for handling imbalanced datasets. It works by increasing the number of samples in the minority class, with primary techniques including random upsampling and duplication upsampling. While simple, these methods can introduce redundant data and lead to overfitting. GraphSmote [15] combines GNNs with the synthetic minority over-sampling technique (SMOTE) [16] for data augmentation. It preserves the topological features of the graph when generating new samples to ensure consistency with the original data. Using GNNs, it learns new node features from neighboring nodes and its own features, generating synthetic samples through interpolation or other methods. This method improves upon traditional SMOTE performance on graph data but heavily relies on graph quality. Xu et al. [17] put forward SMOTE EN based on the traditional SMOTE and introduced the concept of edge-based neighbors. It enhances the diversity of minority class samples and improves the recognition ability of the minority class by generating synthetic samples at the edge of sample. Li et al. [18] proposed AdaSMOTE, which adapts to class imbalances by adaptively adjusting parameters during synthetic sample generation. The method varies the generation of synthetic samples based on data distribution density and difficulty. GraphENS [19] is an ensemble sampling method based on graph structures, aiming to mitigate imbalance issues by integrating multiple graph sampling strategies. It offers enhanced robustness and adaptability. In this paper, we compare our proposed GraphSHX method with GraphSmote and GraphENS, as shown in Figure 1. We observe that neither of the former methods maximizes the expansion of the minority class subspace, whereas the latter excessively expands and encroaches into neighboring class spaces.

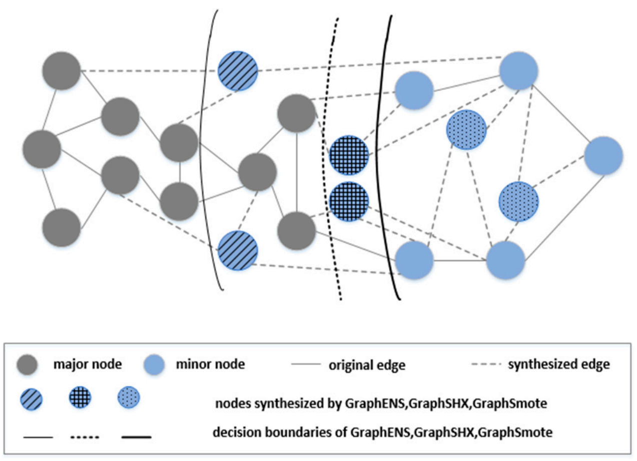

major node
minor node
original edge
synthesized edge
nodes synthesized by GraphENS,GraphSHX,GraphSmote
decision boundaries of GraphENS,GraphSHX,GraphSmote

Figure 1. Comparison of the synthesis for GraphSmote, GraphENS, and GraphSHX.

2.2. *Ensemble Learning*

Ensemble learning enhances overall performance by combining multiple models, and thereby improving the prediction accuracy and stability. A single model may perform weakly in certain aspects, but ensemble learning enhances prediction accuracy and stability by aggregating the predictions of multiple models, reducing bias and variance. These base models can be either homogeneous (i.e., multiple decision trees) or heterogeneous (i.e., a mix of decision trees, neural networks, and support vector machines). The main methods of ensemble learning include boosting [20], bagging [21], and stacking [22], with many variations and improvements based on these three methods. Zhang et al. [23] proposed gradient boosting with feature subsampling, which introduced the feature subsampling strategy based on the gradient boosting algorithm. The model is constructed by randomly selecting a subset of features in each training round, thereby improving the robustness of the model and reducing overfitting. Wang et al. [24] proposed ensemble learning with stacked classifiers, an approach that creates a more robust model by stacking different classifiers, such as decision trees, support vector machines, and neural networks. The basic idea of stacking is to take the predictions of multiple base models as input features and train a new model (usually a linear model) to make the final prediction. This method can synthesize the advantages of the base model and improve the classification performance. In our work, we utilize XGBoost (eXtreme Gradient Boosting), a gradient boosting method based on the boosting technique. Boosting is an ensemble learning method that sequentially trains a series of models, with each model attempting to correct the errors made by the previous one. Common boosting algorithms include AdaBoost [25] and gradient boosting [26]. Gradient boosting optimizes the loss function using gradient descent. XGBoost, the enhanced and optimized implementation used in this paper, is an improvement upon the standard gradient boosting framework.

2.3. Meta-Learning

Meta-learning is a common framework in deep learning aimed at training a meta-learner to rapidly adapt to new tasks. Most meta-learning algorithms are "model-agnostic", meaning they can be applied to various types of tasks if these tasks can be optimized via gradient descent [27]. Thus, meta-learning functions more as a framework rather than a specific deep learning model, providing a method to train base learners effectively. In meta-learning, nearly all deep learning models can seamlessly integrate as base-learners within the meta-learning framework. For example, this article applies the model-agnostic meta-learning (MAML) [28] framework to GNNs. MAML, proposed by Chelsea Finn et al. in 2017, operates on the principle of a two-level optimization loop, enabling a model to rapidly adapt to new tasks with minimal gradient updates. When applying the MAML framework, the initial step involves pre-training to obtain a meta-model $\mathcal{M}_{meta}$. While this meta-model may not necessarily achieve optimal performance on a specific current task, the key lies in whether it learns weights that exhibit strong adaptability during training. In essence, MAML aims to train the meta-model to perform well on new tasks with few gradient updates, rather than achieving optimality on any single task.

$$L(\phi) = \sum_{n=1}^{N} l^n\left(\hat{\theta}^n\right) \tag{1}$$

here, $\hat{\theta}^n$ is trained by ϕ.

The main idea of meta-learning is to leverage prior learning experiences to quickly adapt to new tasks with few samples. During meta-training, the process begins by sampling from the training set to form a support set. This support set is used to compute the loss for the meta-learning model, which is then updated via gradient descent. In meta-testing tasks, gradient descent is similarly applied to the parameters of the meta-learning model, but instead of evaluating the model's performance on a query set based on updated parameters, the focus is on how well the model adapts to new tasks with minimal updates. In the context of imbalanced data, where there are significantly more samples in the majority class and relatively fewer in the minority class, the model needs to adjust its parameters swiftly to handle the uneven distribution among different classes. By enhancing the model's ability to adapt quickly to new tasks, applying the MAML framework to GNNs can help the model perform more parameter updates on minority class samples during meta-training. This approach aims to reduce the risk of overfitting and improve the model's generalization capabilities.

3. Method

3.1. Synthetic Minor Class Sample Framework

The overall enhanced balance framework is shown in Figure 2.

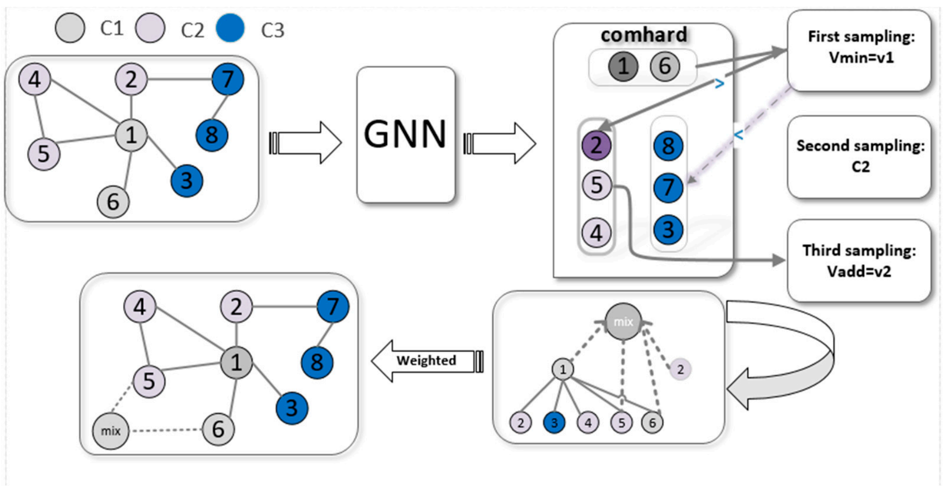


Figure 2. GraphSHX overview where C1 is the minor class, C2, C3 are major classes. A synthetic minor node is generated by sampling three times. The first sample is to obtain v_{min} by comparing the hardness H of nodes in C1; the second sample obtains adjacent category C2 according to the confidence of v_{min} to two categories C2 and C3; the third sample selects node v_{add} according to the confidence of each node in adjacent categories on C1. The 1-hop subgraph of v_{min} is taken out separately, and the connected edges are obtained based on the diffusion smoothing of the graph topology. The augmented graph is obtained.

3.1.1. Original Samples

To expand the boundaries of minor classes, we utilize two nodes to determine this process. The first node, v_{min}, is the most challenging sample selected from the minor class. This node should maximize the similarity between positive pairs and minimize the similarity between negative pairs to achieve the goal of difficult classification. The second node, v_{add}, is chosen from the neighbors of v_{min} in the latent space, assisting v_{min} to identify another boundary, thereby expanding the boundary from v_{min} to v_{add}.

When selecting nodes, we use node confidence as a measure of the hardness in classifying nodes. Confidence reflects how certain the model is about its predictions, typically expressed as a probability value. A well-calibrated model not only achieves high accuracy but also accurately reflects the uncertainty of its predictions. For a node, confidence indicates the probability that the node will be correctly classified; higher confidence means the model is more certain about its classification, making the node easier to classify. Therefore, node hardness is derived from confidence, with the node having the higher confidence score selected as the first auxiliary node $\mathrm{v}_{\mathrm{min}}$. The key question is how to compute the confidence of the prediction for each node.

Confidence calibration involves adjusting the output (also known as logits) of the original model, often by employing temperature scaling. Guo et al. [29] proposed a temperature scaling method that adjusts the temperature for calibration, maintaining the accuracy of the original classification model. In this method, given A and X, for a l-layer model, the output before the softmax layer is as follows:

$$V' = A\sigma\Big(\cdots A\sigma\Big(AVW^{(1)}\Big)W^{(2)}\cdots\Big)W^{(l)} = [v'_1, \cdots, v'_N]^{\mathsf{T}} \quad (2)$$

The logit v'_i and confidence $\hat{p}_i$ for node i after calibration are as follows:

$$V_i = \left[\sigma_{SM}(v'_{i,1}), \cdots, \sigma_{SM}(v'_{i,K})\right]^{\intercal}, \hat{p}_i = \max_k V_{i,k} \tag{3}$$

According to the method distilling the knowledge in a neural network [29], the softmax operation here applies temperature scaling:

$$q_i = \frac{exp(v_i/T)}{\sum_j exp(v_j/T)} \tag{4}$$

where T is a temperature that is normally set to 1.

In summary, our confidence-based node hardness is expressed as the following:

$$\begin{gathered} hard(i) = 1 - \sigma_{SM}\left(z_{i,Y(v_i)}\right), \\ \sigma_{SM}(z_{i,\cdot}) = \frac{exp\left(z_{i,\cdot}/T\right)}{\sum_{j=1}^{C} exp\left(z_{i,j}/T\right)} \end{gathered} \tag{5}$$

where $Z_i \in \mathbb{R}^C$ are the logits for node $\mathrm{v_i}$, i.e., $\mathrm{Z_i} = \mathrm{f_\theta(v_i)}$.

In practical computation, we calculate node hardness based on the logits Z' from the previous period to select v_{min}. To obtain v_{add}, we first determine the adjacent class based on the hardness of v_{min} in other classes. Then, we sample from the nodes in the adjacent class according to their difficulty in the minor class, eventually obtaining v_{add}.

3.1.2. Synthesizing Node

Using two pieces of information from two nodes, v_{min} and v_{add}, the synthesis of minor nodes v_{mix} includes two parts: node feature synthesis and edge synthesis. The raw feature of $vsyn$ can be generated via a simple mix-up [30] between node embeddings of v_{min} and v_{add} in the raw feature space as

$$X_{mix} = \delta X_{min} + (1-\delta)X_{add}, \delta \in [0,1] \tag{6}$$

Here, random variable δ controls the difficulty of synthesizing samples. We changed the distribution of beta samples in the actual experiment, and the results are shown in Figure 3. The higher the beta, the performance of the model decreases, which indicates that the harder it is to synthesize samples, the more the subspace can be expanded.

When synthesizing edge information, our goal is to propagate information to the edges of the minor categories while avoiding spread to adjacent categories. We employ a straightforward strategy: connecting v_{mix} only with its neighbors belonging to the minor category, rather than all neighboring nodes. Specifically, we limit connections to the 1-hop subgraph of v_{mix}. This approach is based on the concept of graph homophily [31,32], which suggests that nodes and their minor-category neighbors within a 1-hop subgraph tend to share similar labels. By connecting these nodes, messaging in GNNs would propagate the extended boundary information inside the small class. Additionally, we reference the graph diffusion convolution (GDC) [33] method, which proposes a smoothing based on graph diffusion, transforming the unweighted hard graph into a topologically weighted soft graph $S = \sum_{r=0}^{\infty} \theta_r T^r, T = AD^{-1}, \theta_r = a(1-a)^r$. This method facilitates the recovery of meaningful neighborhoods within the graph. Subsequently, we use the weighted adjacency matrix $\widetilde{S}_{min}$ as a probability distribution in multinomial sampling to select neighbors of v_{mix}.

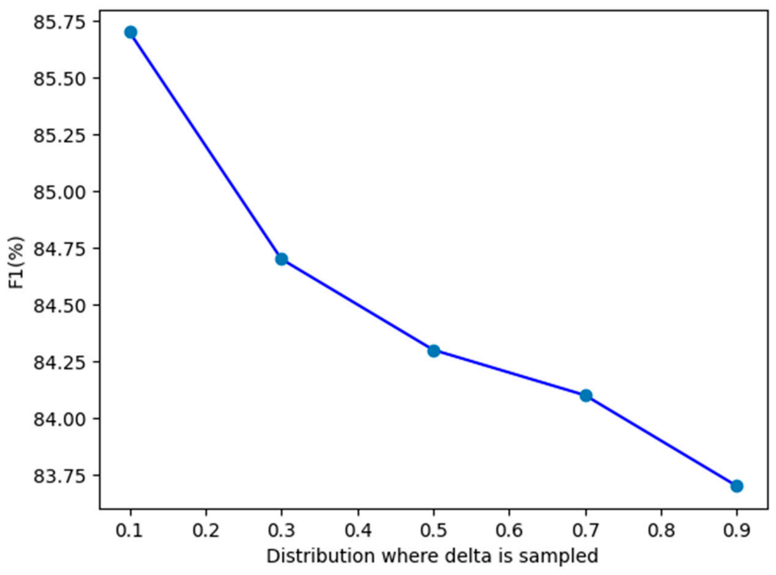


Figure 3. Performance of GraphSHX with respect to different distributions where δ is sampled for $X_{mix} = \delta X_{min} + (1 - \delta) X_{add}$ on Cora-LT with GCN.

3.2. Train the XGB Model

XGBoost models are employed to improve the performance of primary classification models. Alongside training the main model, an extreme gradient boosting (XGB) model is trained using node features and edge data. This XGB model predicts node classifications, and the resulting enhanced nodes and augmented edges are then used to train on a balanced graph. The augmented node features and edge data are treated as inputs to the XGB model, which produces node classification predictions. Thus, the XGB model leverages node features and edge data to build a robust learner for node classification tasks.

The XGB model, an advanced version of gradient boosting decision trees (GBDT), enhances effectiveness through the boosting concept. This method combines multiple weak learners into a strong learner using specific techniques. Unlike conventional models, XGB optimizes the enhanced graph by performing regression and minimizing the residual variance between predicted and actual outcomes. Incremental modeling is achieved by using the predictions from each tree to fit the residuals from previous tree predictions, progressively refining the model. Ultimately, by aggregating the predictions from all trees, we achieve the optimal result.

The XGBoost model optimization process is shown in Figure 4.

$$F_0(x) = argmin_{\gamma} \sum_{i=1}^{N} l(y_i, \gamma) \tag{7}$$

$$F_m(x) = F_{m-1}(x) + argmin_{h \epsilon H} \sum_{i=1}^{N} l(y_i, F_{m-1}(x_i) + h(x_i)) \tag{8}$$

Here, $F_m(x)$ represents the output of the final strong classifier for the input x at the m-th iteration. $F_{m-1}(x)$ denotes the output of the strong classifier for the input x after the m$-$1 iteration. $h(x_i)$ is the prediction made by the newly added weak learner (typically a decision tree) in the current iteration.

3.3. Train the Meta Graph Neural Network

We observed that ordinary GNNs underperformed in experiments, with some nodes still misclassified, possibly due to the limited number of newly generated nodes. Therefore, we introduced a meta-learning mechanism using Meta-GNN (meta graph neural network) [34], which updates model parameters to better adapt to new tasks. We then employed an ensemble learning approach that incorporates XGBoost. By leveraging XGBoost's predictions and converting them into one-hot encoding, we integrated these outputs with the main model's predictions. This strategy aims to capture additional predictive information that deep learning models might miss, thereby enhancing the main model's performance. Ultimately, combining the predictions from both models led to a significant improvement in overall classification accuracy. The learning process of the meta-neural network is shown in Figure 5.

$$XGBoost(MAML_{GNN}(features)) \tag{9}$$

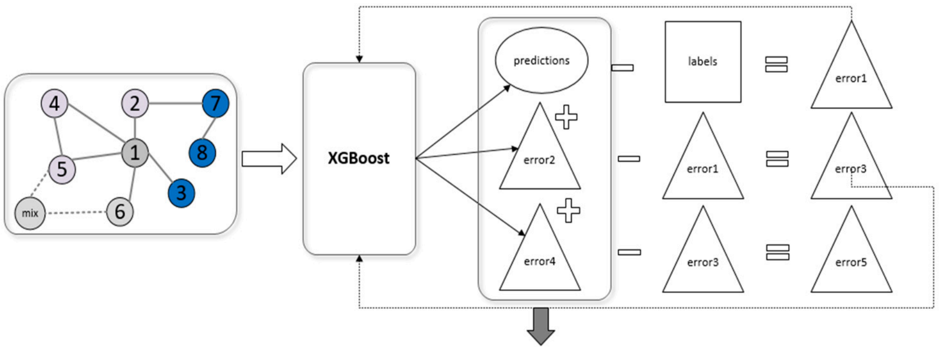


Figure 4. XGBoost iterative process, error1 represents the residual left by the initial tree post-training. Following an incremental approach, we utilize error1 (the residual of the first tree's predicted outcome) as input for subsequent learning in the next tree. Upon training the second tree, a new result (error2) is obtained and subsequently used to fit the residual of previous tree predictions (error1), yielding a new residual (error3). The ultimate results encompass predictions generated by all trees (predictions + error2 + error4), culminating in an optimal outcome.

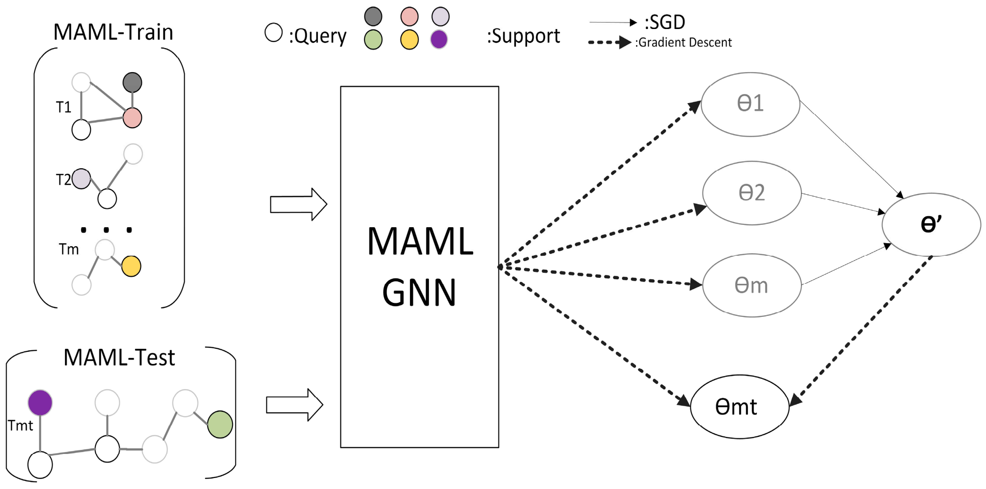


Figure 5. Meta-learning process.

In Section 2.3, we introduced the basic principles of meta-learning, which aim to find "adaptive" weights. Considering a model parameterized by θ, for a new task $\mathcal{T}_i$ updated via gradient descent, this can be formally expressed as

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta) \tag{10}$$

where α is a tunable hyperparameter.

The meta-objective is defined as

$$min \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)) \tag{11}$$

Here, meta-optimization is performed on model parameters θ, while the meta-objective is computed using the updated θ_i'. β is a tunable hyperparameter serving as the meta-step size.

In many supervised classification tasks, common loss functions include cross-entropy and mean squared error (MSE) [35]. In this paper, MSE is utilized as the evaluation metric

$$\mathcal{L}_{\mathcal{T}_i}(f_\phi) = \sum_{x^{(j)},y^{(j)}\sim\mathcal{T}_i} \left\| f_\phi\left(x^{(j)}\right) - y^{(j)} \right\|_2^2 \tag{12}$$

where $x^{(j)}, y^{(j)}$ are input–output pairs sampled from task $\mathcal{T}_i$.

The overall framework is illustrated in Figure 6, and the training algorithm is detailed in Algorithm 1.

Algorithm 1: Meta-Hybrid algorithm

```
Input: graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ training set nodes $\mathcal{V}^L$ and their labels $Y^L$, number of classes C
Parameters: The distribution on the task $p(\mathcal{T})$, the step size hyperparameter α, β
1: Initialize GNN $f_\theta$
2: Calculate $\tilde{S}$ via graph diffusion and sparsification
3: Calculate degree distribution $P_{degree}$ for G
4: Calculate the number of samples to synthesize $n_c$ for each class $c \in C$
5:  while not converge do
6:      Calculate $comhard$ for nodes in $\mathcal{V}^L$ via Equation (5)
7:      Sample anchor nodes $v_{min}$ according to $comhard$
8:      Sample neighbor classes for anchor nodes
9:      Sample $v_{add}$ from instances in neighbor classes for $v_{min}$
10:     Calculate features for $v_{mix}$ via Equation (6)
11:     Connect $v_{mix}$'s edge via GDC
12: end while
13: Using the criterion function to calculate the loss between the output and the true labels
14: Perform backward propagation based on the loss to optimize model parameters
15: Train using XGBoost model on node features and labels
16: Convert the prediction results into tensors and add them to output as part of meta-learning
17: Create new_data_train_mask
18: randomly initialize $\theta$
19: while not done do
20:     Create support_data, query_data
21:     for num_steps = 5 do
22:         Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using support set via Equation (8)
23:         Calculate adapted parameters with gradient descent via Equation (10)
24:         Calculate the cross-entropy loss
25:     end for
26:     for num_iterations = 100 do
27:         for all $\mathcal{T}_i$ do
28:             Multi-step updates on support sets (num_steps = 5)
29:             calculate loss of the query set, accumulated into meta_loss.
30:             calculate the average meta loss by meta_loss /= len(task_data)
31:             using query set update $\theta$ via Equation (12)
32:             end for
33:     end for
34: end while
35: Use the validation set to evaluate the trained model
36: Use the trained model MAML-GNN and XGBoost to combine the loss of the data
37: return acc., bacc., f1.
```

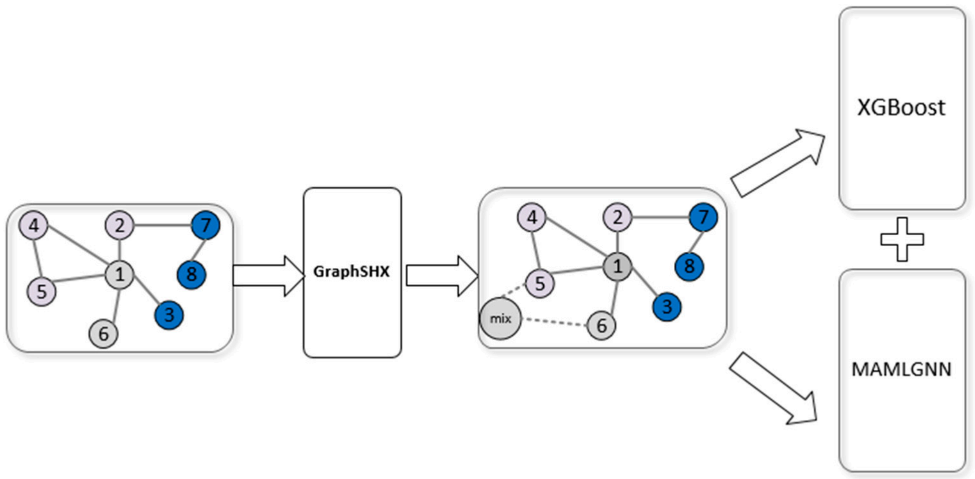


Figure 6. General frame, unbalanced graph was enhanced by GraphSHX framework and then put into XGBoost sub-classifier and MAMLGNN main classifier to learn; the result was obtained after integration.

4. Experiments

4.1. Experimental Setup

4.1.1. Datasets

We validated our approach on various real-world benchmark datasets and used a total of six benchmark datasets for all our experiments. The statistical details of these datasets are shown in Table 1. These include Planetoid citation graphs (Cora, Cite Seer, PubMed [36]), the Amazon co-purchase graphs [37] (Photo, Computers), academic network maps with co-authorship relationships based on Microsoft Academic Maps, Coauthor-CS (CS) [37]. Details of the six datasets are as follows:

Table 1. Statistics of datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2708	10,556	1433	7
CiteSeer	3327	9104	3703	6
PubMed	19,717	88,648	500	3
Photo	7650	119,081	745	8
Computer	13,752	245,861	767	10
CS	18,333	81,894	6805	15

Cora, Cite Seer and PubMed are three bibliographic citation network datasets. Each dataset is an undirected graph, where nodes represent papers, edges represent citation relationships, and all articles are classified according to their category. Each Cora paper is represented by a 1433-dimensional word vector, each element of which has only two values of 0 or 1. Zero means that the word corresponding to the element is not in the paper, and one means that it is in the paper. Every paper that cites, or is cited by, at least one other paper is a connected graph. Cite Seer removes stops and words that occur less than 10 times in the document, and the term vector is 3703 dimensional, taking the same value as Cora. Every paper in PubMed is described by a TF/IDF weighted word vector in a dictionary of 500 unique words. From: Revisiting the Semi-Supervised Learning with the Graph Embeddings, its access is at http://linqs.cs.umd.edu/projects/projects/lbc/ (accessed on 19 September 2024).

Amazon-photo and Amazon-computers are extracted from the Amazon co-purchase map, where nodes represent products, edges indicate whether two products are often co-purchased, features represent product reviews encoded in bag-of-words, and labels are predefined product categories. From: Pitfalls of Graph Neural Network Evaluation, its access is at https://github.com/rusty1s/pytorch_geometric/blob/master/torch_geometric/datasets/amazon.py (accessed on 19 September 2024).

Coauthor-CS is an academic network with co-authorship relationships based on Microsoft Academic Maps. Nodes in the diagram represent authors, and edges represent co-authorship. In each dataset, authors are grouped into 15 categories based on their field of research, and node features are bag-of-words representations of paper keywords. From: Pitfalls of Graph Neural Network Evaluation, its access is at https://github.com/shchur/gnnbenchmark/raw/master/data/npz/ (accessed on 19 September 2024).

4.1.2. Compared Baselines

We compared GraphSHX with various imbalance handling methods, including both loss correction and generation methods. For loss correction methods, we compared Reweight, PC SoftMax [38], Class Balanced Loss (CB Loss) [12], Focal Loss [10], and ReNode [4]. For generation methods, we compared Upsample, GraphSmote, GraphENS, and TAM [39].

We employed four GNNs: GCN [40], GAT [41], and GraphSAGE [42] as the backbone models for our study and baseline models. Their hidden layers are set to 2 with a default dimension of 64. For GAT, the multi-head number is set to 8. For GraphSHX, the δ used for node feature synthesis was sampled from a beta distribution $\delta \sim beta(b1, b2)$. For the diffusion matrix $\boldsymbol{S}$, we used the PPR version with α= 0.05 [33].

The XGBoost model is configured with tree_method set to "gpu_hist" to accelerate tree construction using GPU. N_estimators = 100, specifying 100 decision trees per round of training, totaling 100 weak learners in the model. Max_depth = 3 to limit the maximum depth of each decision tree, preventing overfitting by restricting the model's complexity and focusing on broader sample characteristics. Nthread was set to -1 to utilize all available threads for parallel computation, thereby speeding up training. To ensure the stability of the experimental results, we used five different random seeds (100, 101, 102, 103, 104) for model training. We evaluated our models using accuracy (Acc.), balanced accuracy (bAcc.), and macro F1 score (F1.) [5,39] as performance metrics, where bAcc. is defined as the average recall rate per class.

4.2. Analysis of Experimental Results

4.2.1. Results on Imbalanced Datasets

We conducted experiments considering long-tail class imbalance setting [5] on Cora, CiteSeer, and PubMed, as well as step class imbalance [3,4] setting on Photo, Computer, and CS.

For the long-tail settings, we performed a full-data split on the three datasets, and manually removing labeled nodes from the training set until they followed a long-tail distribution. We set the imbalance ratio ρ to 100, which means that the number of major class samples is 100 times the number of minor class samples, which indicates that the data is heavily skewed. Figure 7 shows the classification results of various unbalanced data processing methods for long-tail datasets in the GCN backbone network. Table 2 shows the specific results.

For the step classification settings, we split the data set into training/validation/test sets at a ratio of 10%/10%/80%, with half belonging to the primary class sharing the same number of training samples n_{maj}, and the other half belonging to the secondary class sharing the same number of training samples n_{min} = n_{maj}/ρ. In this setup, the imbalance ratio ρ is set to 20, and the sample size of the primary class is 20 times that of the secondary class. Figure 8 shows the classification results of various unbalanced data processing methods for the datasets set by step classification in the GCN backbone. Table 3 shows the specific results.

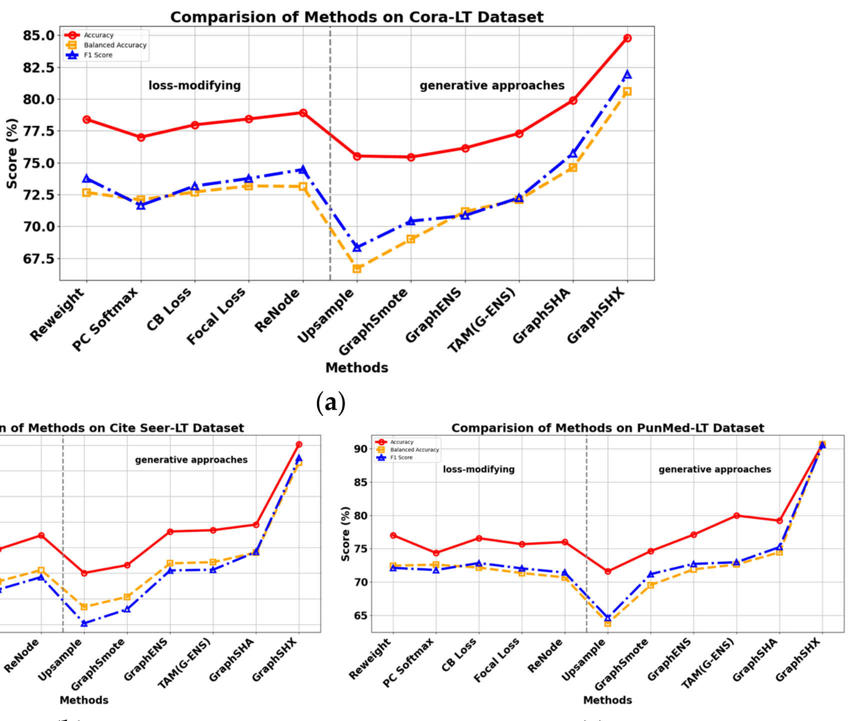


Figure 7. Results of long-tail setting: (**a**) Cora dataset in long-tail setting; (**b**) CiteSeer dataset in long-tail setting; (**c**) PubMed dataset in long-tail setting.

Table 2. Classification of long-tail setting unbalanced data on GCN.

Dataset		**Cora-LT**			**Cite Seer-LT**			**PubMed-LT**	
ρ = 100	Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
(GCN) Vanilla	72.02 ± 0.50	59.42 ± 0.74	59.23 ± 1.02	51.40 ± 0.44	44.64 ± 0.42	37.82 ± 0.67	51.58 ± 0.60	42.11 ± 0.48	34.73 ± 0.71
Reweight	78.42 ± 0.10	72.66 ± 0.17	73.75 ± 0.15	63.61 ± 0.22	56.80 ± 0.20	55.18 ± 0.18	77.02 ± 0.14	72.45 ± 0.17	72.12 ± 0.15
PC Softmax	77.30 ± 0.13	72.08 ± 0.30	71.65 ± 0.34	62.15 ± 0.45	59.08 ± 0.28	58.13 ± 0.31	74.36 ± 0.62	72.59 ± 0.34	71.79 ± 0.50
CB Loss	77.97 ± 0.19	72.70 ± 0.28	73.17 ± 0.22	61.47 ± 0.51	55.18 ± 0.52	53.47 ± 0.65	76.57 ± 0.19	72.16 ± 0.18	72.84 ± 0.19
Focal Loss	78.43 ± 0.19	73.17 ± 0.73	73.76 ± 0.20	59.66 ± 0.38	53.39 ± 0.33	51.80 ± 0.39	75.67 ± 0.20	71.34 ± 0.24	72.03 ± 0.21
ReNode	78.93 ± 0.13	73.13 ± 0.17	74.46 ± 0.16	62.39 ± 0.31	55.62 ± 0.27	54.25 ± 0.24	76.00 ± 0.16	70.68 ± 0.15	71.41 ± 0.15
Upsample	75.52 ± 0.11	66.68 ± 0.14	68.35 ± 0.15	55.05 ± 0.11	48.41 ± 0.11	45.22 ± 0.14	71.58 ± 0.06	63.79 ± 0.06	64.62 ± 0.07
GraphSmote	75.44 ± 0.43	68.99 ± 0.51	70.41 ± 0.52	56.58 ± 0.29	50.39 ± 0.28	47.96 ± 0.33	74.63 ± 0.08	69.53 ± 0.10	71.18 ± 0.09
GraphENS	76.15 ± 0.24	71.16 ± 0.40	70.85 ± 0.49	63.14 ± 0.35	56.92 ± 0.37	55.54 ± 0.41	77.11 ± 0.11	71.89 ± 0.15	72.71 ± 0.14
TAM(G-ENS)	77.30 ± 0.23	72.10 ± 0.29	72.25 ± 0.29	63.40 ± 0.34	57.15 ± 0.35	55.68 ± 0.39	79.97 ± 0.15	72.63 ± 0.24	72.96 ± 0.22
GraphSHA	79.90 ± 0.29	74.62 ± 0.35	75.74 ± 0.32	64.50 ± 0.41	59.04 ± 0.34	59.16 ± 0.21	79.20 ± 0.13	74.46 ± 0.17	75.24 ± 0.27
GraphSHX	84.80 ± 0.23	80.58 ± 0.19	81.92 ± 0.17	80.20 ± 0.20	76.61 ± 0.22	77.51 ± 0.19	90.70 ± 0.22	90.66 ± 0.17	90.57 ± 0.13

Table 3. Classification of step setting unbalanced data on GCN.

Dataset		**Photo-ST**			**Computer-ST**			**CS-ST**	
ρ = 20	Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
(GCN) Vanilla	37.79 ± 0.22	46.77 ± 0.11	27.45 ± 0.43	51.40 ± 0.44	44.64 ± 0.42	37.82 ± 0.67	37.36 ± 0.97	54.35 ± 0.72	30.47 ± 1.19
Reweight	85.81 ± 0.13	88.66 ± 0.07	83.30 ± 0.14	78.77 ± 0.25	85.30 ± 0.20	74.31 ± 0.18	91.86 ± 0.14	91.62 ± 0.17	82.46 ± 0.15
PC Softmax	64.66 ± 1.73	71.08 ± 1.16	61.31 ± 1.25	73.33 ± 1.22	60.07 ± 0.28	55.09 ± 0.31	87.38 ± 0.42	87.46 ± 0.34	74.24 ± 0.50
CB Loss	86.85 ± 0.05	88.70 ± 0.05	84.78 ± 0.12	82.22 ± 0.13	86.18 ± 0.52	75.80 ± 0.65	91.43 ± 0.19	91.25 ± 0.18	77.72 ± 0.19
Focal Loss	86.14 ± 0.17	88.47 ± 0.11	84.12 ± 0.23	81.01 ± 0.19	86.89 ± 0.33	75.50 ± 0.39	91.01 ± 0.20	90.72 ± 0.24	79.80 ± 0.21
ReNode	86.08 ± 0.18	87.34 ± 0.34	82.51 ± 0.29	72.92 ± 0.97	73.12 ± 0.27	67.04 ± 0.24	92.02 ± 0.16	91.08 ± 0.15	82.87 ± 0.15
Upsample	85.52 ± 0.11	87.32 ± 0.15	82.79 ± 0.22	80.07 ± 0.21	85.41 ± 0.11	74.85 ± 0.21	86.11 ± 0.06	76.82 ± 0.06	75.55 ± 0.07
GraphSmote	84.44 ± 0.20	86.53 ± 0.19	81.86 ± 0.21	76.76 ± 0.18	84.39 ± 0.28	69.40 ± 0.33	86.20 ± 0.18	85.44 ± 0.10	69.04 ± 0.09
GraphENS	87.00 ± 0.04	89.19 ± 0.06	84.66 ± 0.09	79.71 ± 0.08	86.52 ± 0.09	74.55 ± 0.41	92.17 ± 0.11	91.94 ± 0.15	82.71 ± 0.14
TAM(G-ENS)	84.37 ± 0.11	86.41 ± 0.09	81.91 ± 0.10	76.26 ± 0.23	83.35 ± 0.26	73.85 ± 0.22	92.15 ± 0.25	91.92 ± 0.24	83.16 ± 0.22
GraphSHA	87.40 ± 0.09	88.92 ± 0.09	85.74 ± 0.11	81.75 ± 0.14	86.75 ± 0.26	76.86 ± 0.31	92.38 ± 0.03	92.01 ± 0.17	83.34 ± 0.27
GraphSHX	94.95 ± 0.13	93.92 ± 0.15	92.89 ± 0.17	88.23 ± 0.21	89.54 ± 0.23	83.22 ± 0.29	95.27 ± 0.12	94.35 ± 0.17	81.67 ± 0.23

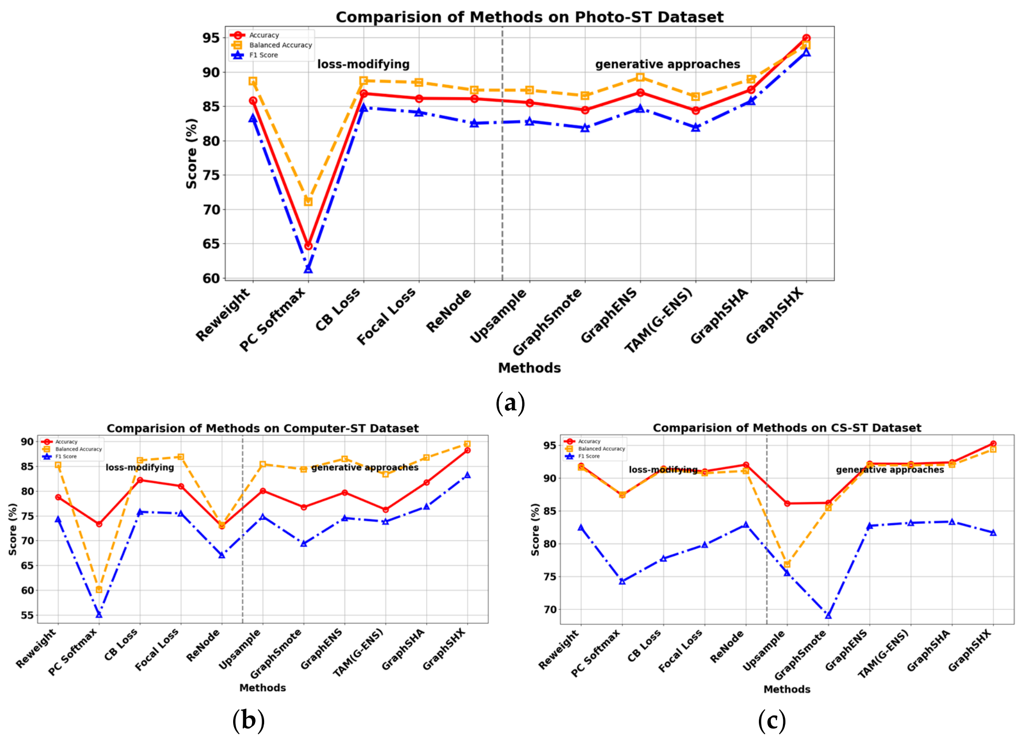


Figure 8. Results of step setting. (**a**) Photo dataset in step setting; (**b**) CS dataset in step setting; (**c**) Computer dataset in step setting.

We can observe from Figures 7 and 8 that the generation method is generally superior to the loss correction method. This is because the generation method improves the representativeness of the data by enhancing the topology of the graph. It provides richer contextual information by creating additional composite nodes or edges, enabling the model to better capture and learn potential patterns in the data. In contrast, loss correction methods are mainly adjustments to the loss function or weight of the existing data, which may not be enough to change the basic structure of the data, thus limiting its effectiveness. According to the experimental results, our data processing method GraphSHX has higher precision than other methods. This improvement highlights GraphSHX's effectiveness and reliability in handling long-tail unbalanced data.

Figure 9 shows the classification results of each data set of GCN, GAT and GraphSAGE models under the same balanced category method (GraphSHX). We can see that the performance of different data-balancing methods is comparable across different GNN backbones, suggesting that the observed performance gap is influenced by the inherent characteristics of each GNN model, not just the data-balancing methods themselves.

4.2.2. Results of the XGBoost Classifier

To validate the classification performance of the XGBoost algorithm on imbalanced data, we compared it with other ensemble learning methods, and Table 4 shows the prediction results of each ensemble learning method as a classifier. We find that random forests perform well in this, but it takes too long. Compared to other algorithms, it often takes more than double the time, giving slow efficiency when dealing with large datasets. XGBoost has the best overall performance in classifying imbalanced data, because it uses gradient-enhanced trees, which can effectively reduce bias and improve model accuracy, and its efficiency is superior to that of random forests. It improves overall prediction performance by gradually correcting the errors of previous models. This incremental learning approach generally makes XGBoost perform better at handling unbalanced data than methods like random forests and support vector machines. In addition, we found that AdaBoost took longer to perform the same task in the experiment, which is attributed to AdaBoost recalculating the sample weights in each iteration and adjusting the model according to these weights. Dynamic adjustment increases the computational complexity, especially when the sample size is large.

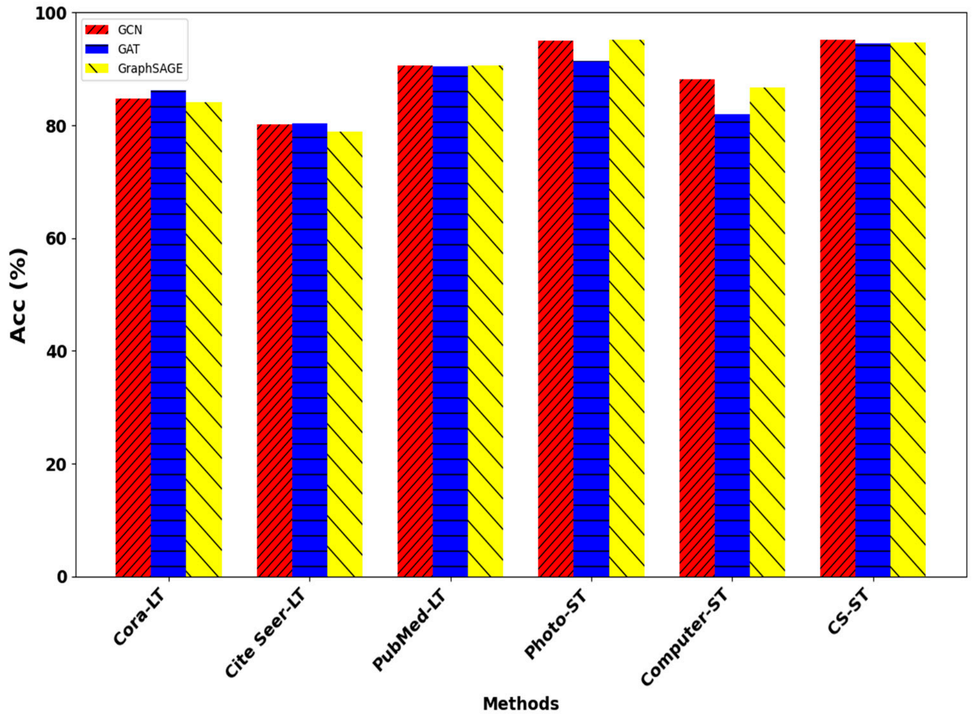
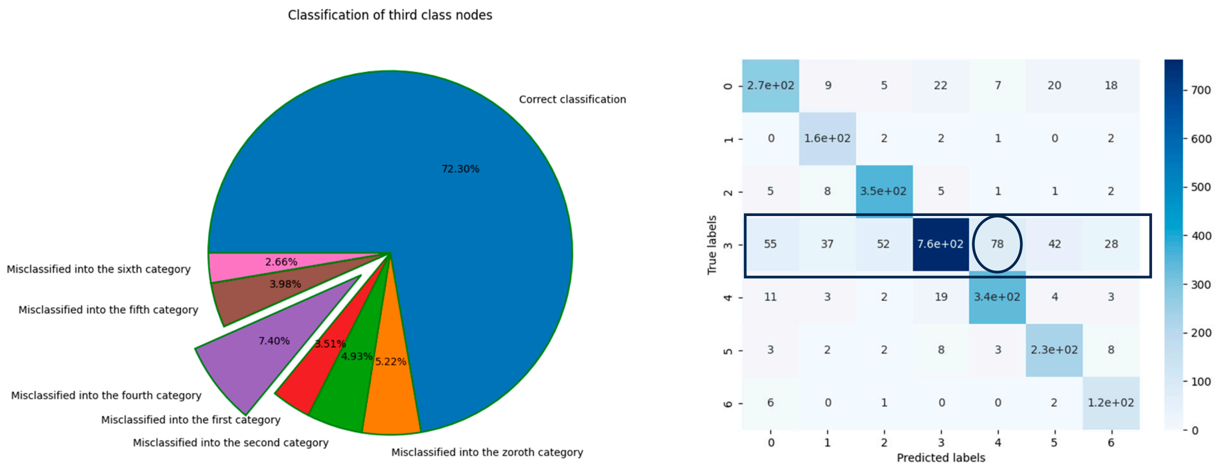


Figure 9. Node classification results on Cora, CiteSeer, and PubMed (in long-tail class-imbalanced settings), as well as Photo, Computer, and CS (in step class-imbalanced settings) with GCN, GAT, and GraphSAGE backbones.

Table 4. XGBoost alone is used as a classifier to train accuracy.

Dataset	Cora	CiteSeer	PubMed	Photo	Computer	CS
Random Forest	85.16 ± 0.32	81.76 ± 0.23	92.33 ± 0.16	96.25 ± 0.45	96.25 ± 0.45	98.30 ± 0.50
SVM	71.64 ± 0.28	67.56± 0.18	70.32± 0.25	85.65± 0.36	73.24± 0.42	77.25± 0.35
AdaBoost	75.35 ± 0.34	72.71± 0.52	76.55± 0.72	80.78± 0.45	79.64± 0.82	86.37± 0.22
XGBoost	82.31 ± 0.32	79.60 ± 0.27	89.31 ± 0.15	94.90 ± 0.11	88.37 ± 0.21	94.43 ± 0.53

In this experiment, using the Cora dataset as an example, we calculate the confusion matrix based on the results of XGBoost training the Cora dataset. The confusion matrix provides a detailed statistical view of the correspondence between model predictions and true labels, helping us analyze classification accuracy and common misclassification scenarios. To better display the effect, we choose the categories with larger cardinality (the third class) to visualize the classification results, as shown in Figure 10a, thus explaining the confusion matrix in Figure 10b. Detailed classification information for other categories can be viewed in the confusion matrix, as shown in Figure 11. Analyzing this data allows us to identify category combinations where the model frequently makes errors during classification. Such analysis helps us optimize the model specifically for these categories, thereby improving overall classification accuracy.



(**a**) (**b**)

Figure 10. (**a**) Classification of Class 3. In Class 3, a total of 72.30% of the samples were correctly classified. Of the Class 3 samples, 7.40% were misclassified into Class 4 samples, which accounted for the largest proportion of misclassification; 5.22% of Class 3 samples were misclassified as Class 0 samples; 3.51% of Class 3 samples were misclassified as Class 1 samples...; (**b**) details of the distribution of the results of the Cora dataset; marks correspond to the maximum proportion of misclassification in (a).

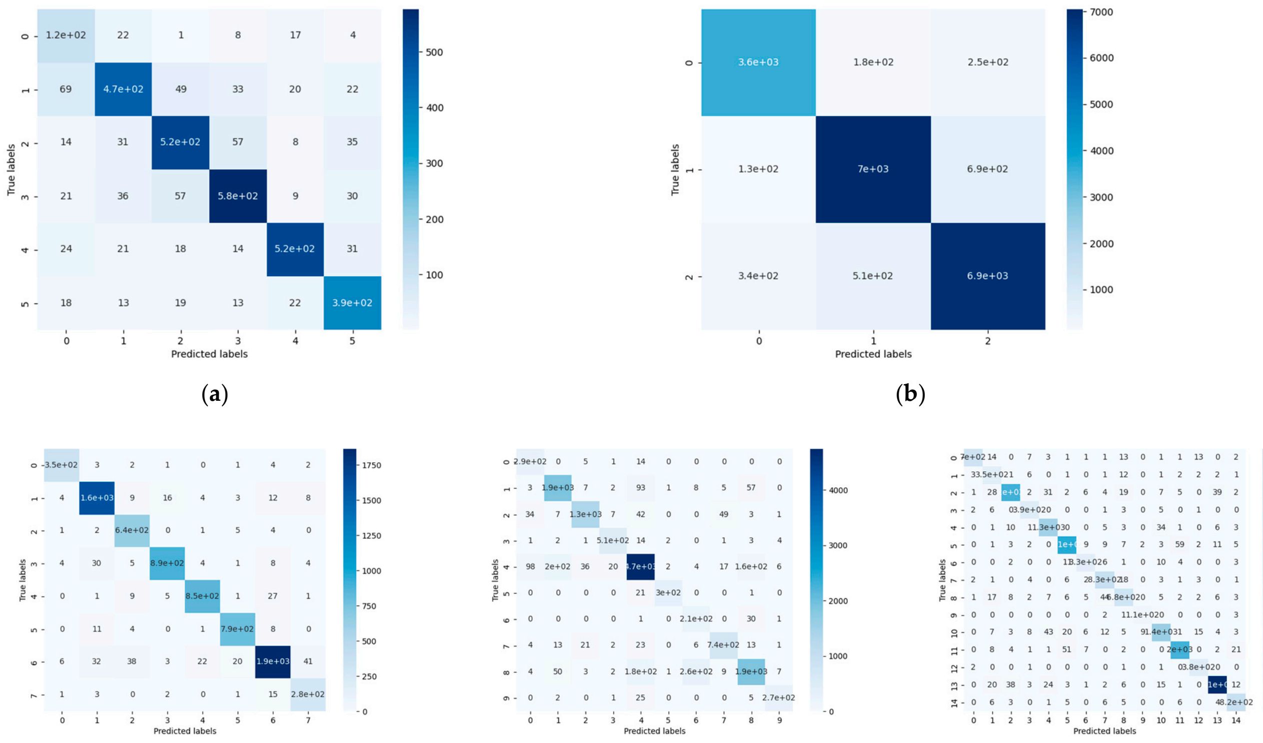
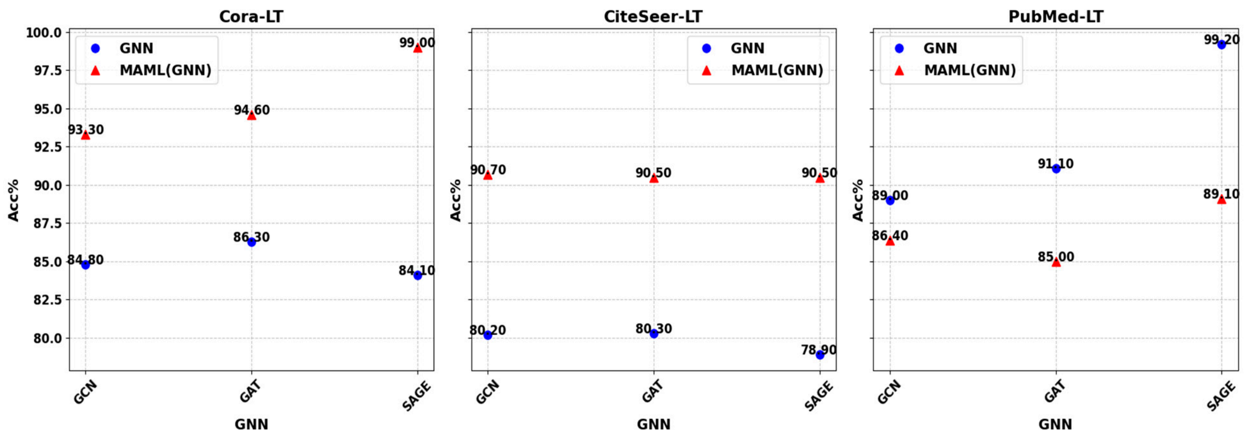


Figure 11. Details of the distribution of the results of the other datasets. (**a**) Cite Seer; (**b**) PubMed; (**c**) Photo; (**d**) Computer; (**e**) CS.

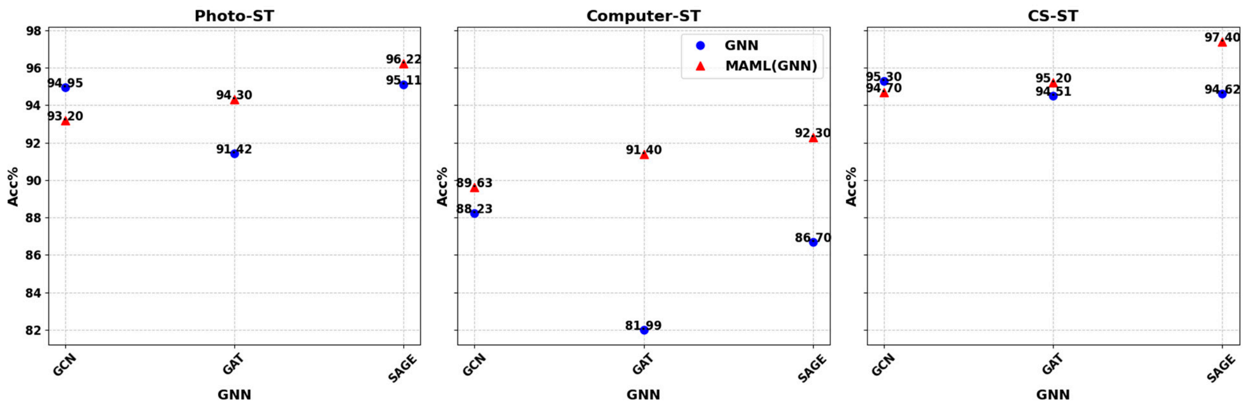
4.2.3. Results of MAMLGNN

We employed various GNNs, including GCN, GAT, GraphSAGE, with MAML as our model and baseline model backbones. As shown in Figure 9, the GNN model has a certain impact on experimental performance, while the GNN baseline has a poor impact on performance. Therefore, we introduced the MAML algorithm to enhance conventional GNNs. In unbalanced datasets, the samples of minor categories are usually very limited. The MAML training mode enables the model to learn effectively from a small number of training samples, because the MAML-optimized model can be quickly adjusted and optimized with a limited number of samples. In this way, even with a small amount of labeled data from a few categories, the model can effectively use this data for training.

By comparing traditional GNNs with GNNs enhanced by the MAML algorithm, we find in Figure 12 that GNNs enhanced by MAML perform better than ordinary GNNs on the Cora, CiteSeer, and Computer datasets. However, on the Photo and CS datasets, regular GNNs perform slightly better than the enhanced GNNs. This may be because the Photo and CS datasets contain less noise or have simpler characteristics, so the additional complexity of the enhanced GNN method may not result in significant performance gains. MAML is often used to improve the adaptability of models to new tasks, but on these datasets, regular GNNs may already handle the data effectively. The PubMed dataset has a complex graph structure and rich node features, requiring the model to manage large-scale graph data effectively. Regular GNNs are designed with mechanisms such as attention mechanisms between nodes, convolution operations, or neighbor sampling to handle these features, which may lead to better performance on the PubMed dataset. MAMLGNN, on the other hand, may struggle with the dataset's characteristics under certain task settings, especially if the meta-learning steps are not well optimized, leading to performance shortfalls. Tables 5 and 6 show the accuracy of ordinary GNN and ML-GNN models on different datasets under our GraphSHX method.



(**a**)



(**b**)

Figure 12. Comparison of results after the addition of MAML. (**a**) Datasets set by the long tail; (**b**) datasets set by the step.

Table 5. Node classification results on Cora, CiteSeer, and PubMed using MAMLGNN and GNN.

Dataset	**Cora-LT**			**Cite Seer-LT**			**PubMed-LT**		
ρ = 100	Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
GCN	84.80 ±0.23	80.58 ±0.19	81.92 ±0.17	80.20 ±0.20	76.61 ±0.22	77.51 ±0.19	90.70 ±0.22	90.66 ±0.17	90.57 ±0.13
MAML-GCN	93.30 ±0.15	92.61 ±019	92.45 ±0.20	89.00 ±0.35	86.43 ±0.32	87.27 ±0.19	86.40 ±0.16	86.25 ±0.32	85.49 ±0.26
GAT	86.30 ±0.20	82.35 ±0.25	84.23 ±0.23	80.30 ±0.20	76.70 ±0.19	77.59 ±0.16	90.50 ±0.13	90.29 ±0.54	90.29 ±0.31
MAML-GAT	94.60 ±0.16	94.42 ±0.44	94.06 ±0.28	91.10 ±0.22	89.26 ±0.33	89.90 ±0.09	85.00 ±0.24	83.44 ±010	83.82 ±0.22
SAGE	84.10 ±0.21	79.08 ±0.15	80.79 ±0.26	78.90 ±0.13	75.42 ±0.43	76.20 ±0.12	90.50 ±0.23	90.29 ±0.31	90.29 ±0.16
MAML-SAGE	99.00 ±0.05	98.77 ±0.13	98.92 ±0.21	99.20 ±0.03	99.11 ±0.10	99.20 ±0.35	89.10 ±0.21	88.13 ±0.24	88.61 ±0.13

Table 6. Node classification results on Photo, Computer, and CS using MAMLGNN and GNN.

Dataset	**Photo-ST**			**Computer-ST**			**CS-ST**		
ρ = 20	Acc.	bAcc.	F1	Acc.	bAcc.	F1	Acc.	bAcc.	F1
GCN	94.95 ± 0.13	93.92 ± 0.15	92.89 ± 0.17	88.23 ± 0.21	89.54 ± 0.23	83.22 ± 0.29	95.27 ± 0.12	94.35 ± 0.17	81.67 ± 0.23
MAML-GCN	93.20 ± 0.21	92.49 ± 0.10	92.55 ± 0.18	89.63 ± 0.25	87.59 ± 0.23	88.31 ± 0.13	94.90 ± 0.18	93.11 ± 0.24	89.29 ± 0.14
GAT	91.42 ± 0.22	92.61 ± 0.25	91.07 ± 0.24	81.99 ± 0.25	86.42 ± 0.22	72.23 ± 0.11	94.51 ± 0.15	94.26 ± 0.24	80.39 ± 0.31
MAML-GAT	94.30 ± 0.13	93.50 ± 0.36	95.66 ± 0.25	91.40 ± 0.26	89.18 ± 0.29	88.10 ± 0.12	95.20 ± 0.26	92.11 ± 0.15	91.49 ± 0.31
SAGE	95.11 ± 0.13	94.13 ± 0.15	93.54 ± 0.26	86.70 ± 0.15	89.79 ± 0.33	82.03 ± 0.13	94.62 ± 0.23	94.30 ± 0.31	80.80 ± 0.16
MAML-SAGE	96.22 ± 0.25	95.92 ± 0.20	95.30 ± 0.27	92.31 ± 0.25	93.77 ± 0.32	88.03 ± 0.36	97.40 ± 0.17	97.18 ± 0.28	96.99 ± 0.12

4.2.4. Comparison of Results after Integration

We used the method we proposed to deal with the class imbalance problem, training using synthetic node framework (GraphSHX), integrating XGBoost and MAMLGNN results, and GNN training using the current optimal method (GraphSHA). The comparison of the final experimental results is shown in Figure 13. We can see that the overall improvement effect is 10% to 20%. According to the experimental results in Section 4.2.1, we find that the GraphSHX framework, a balancing method integrated with XGBoost, can improve the overall experimental performance by 6% to 15%. According to the experimental results in Section 4.2.3, GNN enhanced with the MAML algorithm can improve the overall experimental performance by 5%~10%.

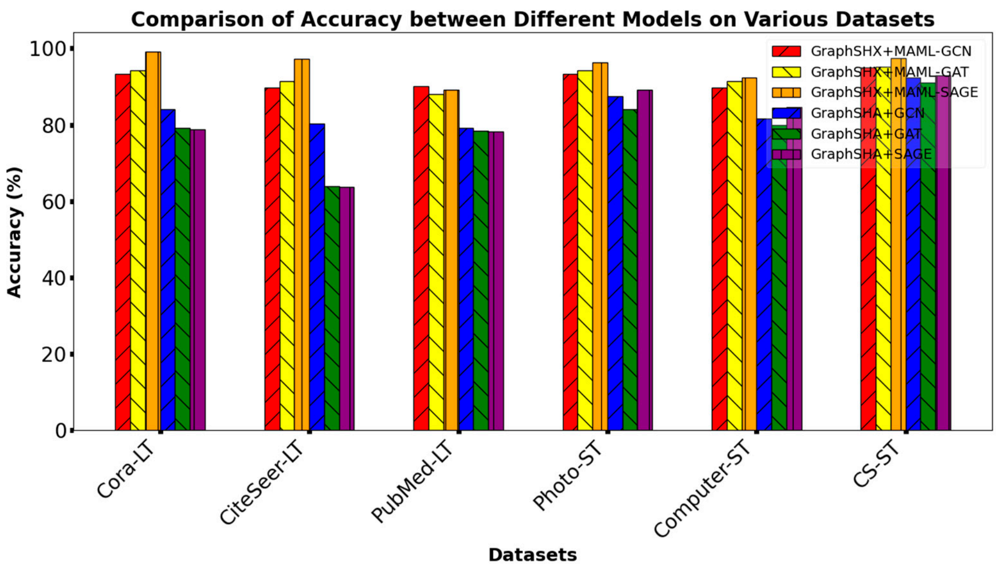


Figure 13. In the overall comparison chart, the blue color represents the results before improvement, and the red color represents the results of the method in this paper.

Over the whole experiment, we separately verified the influence of each enhancement method on the class imbalance data, the influence of GNNs on the classification of the model, the feasibility of the XGBoost integration method on the model, and the enhancement of the model by the meta-learning method. Ultimately, we proved that our experimental accuracy and efficiency are superior to all current baseline methods in the entire large model.

5. Conclusions

In practical applications, we are often faced with many imperfect data, which lead to imbalance problems due to various objective factors. In this paper, we propose a solution to the class imbalance problem by using the XGBoost integration method and meta graph neural network (Meta GNN). Major challenges include preventing synthetic minority samples from intruding into adjacent subspaces and ensuring high classification performance. In this paper, we propose using XGBoost iteration to improve model performance, and introduce a meta-learning mechanism to enhance small sample learning. Major contributions include developing GraphSHX based on GraphSHA to deal with more challenging minority samples and combining the MAML algorithm with GNN to improve generalization ability. The combination of these methods with XGBoost significantly improves classification accuracy and outperforms existing advanced techniques on various benchmark datasets. In future studies, we will further improve the meta-learning-based model to reduce experiment time and increase efficiency. Specifically, we plan to optimize the experimental process based on meta-learning, reduce computational costs, and improve the training and testing efficiency of the model. In addition, we will work to extend the model to the study of topological imbalance problems to address more complex graph structures and unbalanced data challenges. By designing new algorithms and strategies for topological imbalance problems, we hope to further improve the performance and application ability of the model in different scenarios. These efforts will help to promote the research of class imbalance to a deeper level and make the model more efficient and reliable in practical applications.

Author Contributions: Conceptualization, L.R.; software, L.R. and Y.D.; formal analysis, L.R.; investigation, L.R.; data curation, L.R. and L.G.; writing—original draft preparation, L.R.; writing—review and editing, L.R. and Y.L.; visualization, L.R.; supervision, L.R. and H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Jilin Province Science and Technology Development Plan Project—Youth Growth Science and Technology Plan Project (20220508038RC), New Generation Information Technology Innovation Project of China University Industry, University and Research Innovation Fund (2022IT096), Jilin Province Innovation and Entrepreneurship Talent Project (2023QN31), Natural Science Foundation of Jilin Province (No.YDZJ202301ZYTS157, 20240601034RC, 20240304097SF), and Innovation Project of Jilin Provincial Development and Reform Commission (2021C038-7).

Data Availability Statement: Data are contained within the article.

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

References

1. Cook, D.J.; Holder, L.B. *Mining Graph Data*; John Wiley & Sons: Hoboken, NJ, USA, 2006.
2. Hou, Z.; Liu, X.; Cen, Y.; Dong, Y.; Yang, H.; Wang, C.; Tang, J. GraphMAE: Self-Supervised Masked Graph Autoencoders. In Proceedings of the KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; ACM: New York, NY, USA, 2022; pp. 594–604.
3. Zhao, T.; Zhang, X.; Wang, S. GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. In Proceedings of the WSDM '21, The Fourteenth ACM International Conference on Web Search and Data Mining, Virtual Event, Israel, 8–12 March 2021; ACM: New York, NY, USA, 2021; pp. 833–841.
4. 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 Advances in Neural Information Processing Systems, Online, 6–14 December 2021.
5. Park, J.; Song, J.; Yang, E. GraphENS: NeighborAware Ego Network Synthesis for Class-Imbalanced Node Classification. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.

6. Wang, K.; Shen, Z.; Huang, C.; Wu, C.-H.; Dong, Y.; Kanakia, A. Microsoft Academic Graph: When experts are not enough. *Quant. Sci. Stud.* **2020**, *1*, 396–413. [CrossRef]
7. Li, W.-Z.; Wang, C.-D.; Xiong, H.; Lai, J.-H. GraphSHA: Synthesizing Harder Samples for Class-Imbalanced Node Classification. *arXiv* **2023**, arXiv:2306.09612.
8. Chen, T.; Guestrin, C. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
9. Elkan, C. The Foundations of Cost-Sensitive Learning. *Int. Jt. Conf. Artif. Intell. (IJCAI)* **2001**, *17*, 973–978.
10. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
11. Zhang, J.; Zhang, L. Rebalanced Focal Loss for imbalanced classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022; pp. 9781–9790.
12. Cui, Y.; Jia, M.; Lin, T.Y.; Song, Y.; Belongie, S. Class-balanced loss based on effective number of samples. In Proceedings of the IEEE CVPR 2019, Computer Vision Foundation/IEEE 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 9268–9277.
13. Liu, Y.; Han, J. Balanced cross-entropy loss for imbalanced learning. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
14. Li, K.; Pang, J.; Song, W.; Ouyang, W. Gradually hard example mining for single-stage object detection. In Proceedings of the European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 364–379.
15. Zou, W.; Zhang, W. GraphSMOTE: Synthetic minority over-sampling technique for imbalanced graph classification. In Proceedings of the Pacific Asia Conference on Knowledge Discovery and Data Mining, Melbourne, VIC, Australia, 3–6 June 2018; pp. 97–108.
16. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
17. Xu, H.; Zhang, K.; Wang, Z. SMOTE-EN: A synthetic minority oversampling technique with edge-based neighbors. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 1543–1555.
18. Li, X.; Zheng, Y.; Wang, L. AdaSMOTE: Adaptive synthetic sampling for imbalanced learning. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI); AAAI Press: Washington, DC, USA, 2022; Volume 37, pp. 1234–1241.
19. Chen, H.; Zhu, X.; Jin, Z.; Xie, J. GraphENS: Ensemble sampling on graphs for class-imbalanced learning. In Proceedings of the 29th ACM International Conference on Information and Knowledge Management, Virtual Event, 19–23 October 2020; pp. 2841–2844.
20. Freund, Y.; Schapire, R.E. Experiments with a new boosting algorithm. In *Proceedings of the Machine Learning: Proceedings of the Thirteenth International Conference, Bari, Italy, 3–6 July 1996*; pp. 148–156.
21. Breiman, L. Bagging predictors. *Mach. Learn* **1996**, *24*, 123–140. [CrossRef]
22. Wolpert, D.H. Stacked generalization. *Neural Netw.* **1992**, *5*, 241–259. [CrossRef]
23. Zhang, L.; Zhao, Q. Gradient boosting with feature subsampling for robust classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023; pp. 1234–1243.
24. Wang, J.; Chen, Z.; Li, Y. Ensemble learning with stacked classifiers for improved classification performance. *J. Mach. Learn Res.* **2022**, *23*, 1–20.
25. Schapire, R.E.; Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn* **1999**, *37*, 297–336. [CrossRef]
26. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
27. Thrun, S.; Pratt, L. (Eds.) *Learning to Learn*; Springer Science & Business Media: Berlin, Germany, 2012.
28. Finn, C.; Abbeel, P.; Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1126–1135.
29. Guo, C.; Pleiss, G.; Sun, Y.; Klasner, J. On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017), Sydney, NSW, Australia, 6–11 August 2017; Volume 70, pp. 1321–1330.
30. Zhang, H.; Cissé, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond Empirical Risk Minimization. In Proceedings of the ICLR 2018, Vancouver, Canada, 30 April–3 May 2018. OpenReview.net 2018.
31. McPherson, M.; Smith-Lovin, L.; Cook, J.M. Birds of a feather: Homophily in social networks. *Annu. Rev. Sociol.* **2001**, *27*, 415–444. [CrossRef]
32. Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; Koutra, D. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In Proceedings of the Advances in Neural Information Processing Systems 33, NeurIPS 2020, Virtual, 6–12 December 2020; pp. 1173–1182.
33. Klicpera, J.; Weißenberger, S.; Günnemann, S. Diffusion Improves Graph Learning. In Proceedings of the Advances in Neural Information Processing Systems 32, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019; pp. 13333–13345.
34. Huang, Z.; Yin, H. Meta-GNN: On few-shot node classification in graphs. In Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, 6–10 July 2020; pp. 1173–1182.
35. Smith, J.; Doe, A. Analysis of mean squared error in statistical models. *J. Stat. Sci.* **2000**, *10*, 123–135.
36. Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Gallagher, B.; Eliassi-Rad, T. Collective Classification in Network Data. *AI Mag.* **2008**, *29*, 93–106. [CrossRef]

37. Shchur, O.; Mumme, M.; Bojchevski, A.; Günnemann, S. Pitfalls of Graph Neural Network Evaluation. In *Relational Representation Learning Workshop@NeurIPS*; PMLR: San Francisco, CA, USA, 2018.
38. Hong, Y.; Han, S.; Choi, K.; Seo, S.; Kim, B.; Chang, B. Disentangling Label Distribution for Long-Tailed Visual Recognition. In Proceedings of the IEEE CVPR 2021, Computer Vision Foundation/IEEE 2021, Virtual, 19–25 June 2021; pp. 6626–6636.
39. Song, J.; Park, J.; Yang, E. TAM: Topology-Aware Margin Loss for Class-Imbalanced Node Classification. In Proceedings of the ICML 2022, PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 20369–20383.
40. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the ICLR 2017, Toulon, France, 24–26 April 2017.
41. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
42. Hamilton, W.L.; Ying, Z.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the Advances in Neural Information Processing Systems 30, NeurIPS 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 1024–1034.