A Other related work

Imbalanced Learning in Vision Domain. Most real-world data is naturally imbalanced, presenting a significant challenge in training fair models that are not biased towards majority classes. To address this problem, various approaches have been commonly utilized. Ensemble learning [7, 18, 49, 38, 20, 3] combines the outputs of multiple weak classifiers. Data re-sampling methods [5, 10, 29, 27, 14, 36] aim to adjust the label distribution in the training set by synthesizing or duplicating samples from the minority class. Another approach tackles the imbalance issue by modifying the loss function, assigning larger weights to minority classes or adjusting the margins between different classes [50, 32, 4, 31, 40, 26, 37]. Post-hoc correction methods compensate for the imbalanced classes during the inference step, after completing the model training [14, 33, 21, 12]. Although these techniques have been extensively applied to i.i.d. data, extending them to graph-structured data poses non-trivial challenges.

Graph Contrastive Learning. Contrastive methods, which have proven effective for unsupervised learning in vision, have also been adapted for graph data. One notable approach is DGI [35], which presents a framework for unsupervised node-level representation learning that maximizes global mutual information. Other approaches, such as GRACE [51], GCA [52], and GraphCL [45], utilize augmented graphs to optimize the similarity between positive node pairs and minimize negative pairs. CCA-SSG [46] introduces an efficient loss function based on canonical correlation analysis, eliminating the need for negative samples. Incorporating community information, gCooL [17] enhances node representations and downstream task performance. GGD [48] simplifies the mutual information loss function by directly discriminating between two sets of node samples, resulting in faster computation and lower memory usage. These contrastive methods exhibit potential for improving unsupervised learning on graph data. However, it is important to note that our model operates within the context of semi-supervised learning, which significantly differs from the mechanisms employed by these models.

B Proofs

B.1 Proofs of Theorem 1

Theorem 1 *Under the condition that $\sum_{i=1}^{c} n_i$ is a constant, the variance $\sum_{i=1}^{c} \mathbb{E}_x \left[\frac{1}{n_i} h^T(x)\Lambda^i h(x)\right]$ reach its minimum when all n_i equal.*

Proof The expression $\sum_{i=1}^{c} \mathbb{E}_x \left[\frac{1}{n_i} h^T(x)\Lambda^i h(x)\right]$ can be equivalently expressed as $\sum_{i=1}^{c} \frac{1}{n_i}\mathbb{E}_x \left[h^T(x)\Lambda^i h(x)\right]$. As previously assumed, the $\mathbb{E}_x[h^T(x)\Lambda^i h(x)]$ is the same for different i, which implies that our goal is to demonstrate that $\sum_{i=1}^{c} \frac{1}{n_i}$ is minimized when all n_i are equal.

Let m be $\sum_{i=1}^{c} n_i$. We wish to find the extremum of the sum of their reciprocals, which is given by

$$S = \frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_c}. \tag{11}$$

Using the inequality of arithmetic and harmonic means, we have

$$\frac{c}{\frac{1}{n_1} + \frac{1}{n_2} + \cdots + \frac{1}{n_c}} \leq \frac{n_1 + n_2 + \cdots + n_c}{c}, \tag{12}$$

with equality if and only if $n_1 = n_2 = \cdots = n_c$. Rearranging, we get

$$\frac{c^2}{n_1 + n_2 + \cdots + n_c} \leq S, \tag{13}$$

with equality if and only if $n_1 = n_2 = \cdots = n_c$. Since $m = n_1 + n_2 + \cdots + n_c$, we have

$$\frac{c^2}{m} \leq S, \tag{14}$$

with equality if and only if $n_1 = n_2 = \cdots = n_c = \frac{m}{c}$. Therefore, when all the c numbers are equal, the sum of their reciprocals is minimized and given by $S = \frac{c^2}{m}$.

We can also use the method of Lagrange multipliers. Let $f(n_1, n_2, \ldots, a_c) = \frac{1}{n_1} + \frac{1}{n_2} + \ldots + \frac{1}{n_c}$ be the function that we want to extremize. Then, the Lagrangian is:

$$\mathcal{L}(n_1, n_2, \ldots, n_c, \lambda) = f(n_1, n_2, \ldots, n_c) + \lambda(n_1 + n_2 + \ldots + n_c - m). \tag{15}$$

Taking the partial derivatives of $\mathcal{L}$ with respect to a_i and λ, we get:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial n_i} &= -\frac{1}{n_i^2} + \lambda \\ \frac{\partial \mathcal{L}}{\partial \lambda} &= n_1 + n_2 + \ldots + n_c - m. \end{aligned} \tag{16}$$

Setting these partial derivatives to zero, we get:

$$n_1 = n_2 = \ldots = n_c = \frac{m}{c}, \lambda = \frac{c^2}{m^2}. \tag{17}$$

Thus, when the c numbers are equal, their reciprocal sum is minimized and is equal to $\frac{c^2}{m}$. Moreover, since S is a continuously differentiable function of $n_1, n_2, \ldots, n_c$, this extremum is a minimum.

Therefore, we have shown that the reciprocal sum of c numbers with a sum of m is minimized and equal to $\frac{c^2}{m}$ when the c numbers are equal.

□

B.2 Proofs of Lemma 1 and More Details for Estimating the Expectation with Unlabeled Nodes (Section 3.2)

B.2.1 Proofs of Lemma 1

Lemma 1 *Under the above assumption for* $h^i \sim N(\mu^i, \Lambda^i)$, $C^i \sim N\left(\mu^i, \frac{1}{n_i}\Lambda^i\right)$, *minimizing the* $\sum_{i=1}^{c} \mathbb{E}_x\left[\mathrm{Var}(x)\right]$ *is equivalent to minimizing Equation 18:*

$$\frac{1}{N}\sum_{x\in G}\sum_{i=1}^{c}\left(h(x)^T\frac{1}{\sqrt{2n_i}}\left(h_1^i-h_2^i\right)\right)^2=\frac{1}{N}\sum_{k=1}^{n_j}\sum_{j=1}^{c}\sum_{i=1}^{c}\left((\mu_k^j+\epsilon_k^j)^T\frac{1}{\sqrt{2n_i}}\left(\epsilon_1^i-\epsilon_2^i\right)\right)^2. \quad (18)$$

Proof Let ϵ^i denote $h^i-\mu^i$, which follows the distribution $\epsilon^i \sim N(0,\Lambda^i)$. Similarly, we denote $e^i = C^i-\mu^i$ and it follows that $e^i \sim N\left(0, \frac{1}{n_i}\Lambda^i\right)$.

We know

$$\begin{aligned}\mathrm{Var}(x)&=\sum_{i=1}^{c}\mathbb{E}_D\left[\left(h^T(x)C^i-\mathbb{E}_D\left[h^T(x)C^i\right]\right)^2\right]\\&=\sum_{i=1}^{c}\mathbb{E}_{e^i}\left[\left(h^T(x)(\mu^i+e^i)-h^T(x)\mu^i\right)^2\right]\\&=\sum_{i=1}^{c}\mathbb{E}_{e^i}\left[\left(h^T(x)e^i\right)^2\right]=\sum_{i=1}^{c}\frac{1}{n_i}h^T(x)\Lambda^i h(x),\end{aligned} \quad (19)$$

so we get

$$\sum_{i=1}^{c}\mathbb{E}_x\left[\mathrm{Var}(x)\right]=\sum_{i=1}^{c}\mathbb{E}_x\left[\frac{1}{n_i}h^T(x)\Lambda^i h(x)\right]. \quad (20)$$

Motivated by stochastic gradient descent, we opt to sample an e^i on each occasion and compute, as opposed to directly calculating the expectation $\mathbb{E}_{D\subset G}$.

At present, we remain uncertain about how to sample the noise term e^i associated with the class center C_i. We put forth a proposition to estimate the class center noise e utilizing the feature noise ϵ. Under the supposition that when $v \in C_k$, the j_{th} element of the feature $f(v)$ adheres to a Gaussian distribution:

$$\epsilon^i \sim N\left(0,\Lambda^i\right),\ e^i\ \sim\ N\left(0,\frac{1}{n_i}\Lambda^i\right). \quad (21)$$

Consequently, multiplying the noise term ϵ^i in Equation 21 by $\frac{1}{\sqrt{n_i}}$ yields a random variable that exhibits an identical distribution to e^i:

$$\frac{1}{\sqrt{n_i}}\epsilon^i \sim N\left(0,\frac{1}{n_i}\Lambda^i\right) \quad (22)$$

Moreover, how might we compute ε_i? In practical scenarios, we merely possess the feature h. However, we are able to calculate the disparity between two features, h_1^i and h_2^i, originating from the same class i:

$$\frac{1}{\sqrt{2n_i}}\left(h_1^i-h_2^i\right)=\frac{1}{\sqrt{2n_i}}(\epsilon_1-\epsilon_2)\sim N\left(0,\frac{1}{n_i}\Lambda^i\right). \quad (23)$$

Incorporating this into Equation 20, the loss is expressed as:

$$\frac{1}{N}\sum_{x\in G}\sum_{i=1}^{c}\left(h(x)^T\frac{1}{\sqrt{2n_i}}\left(h_1^i-h_2^i\right)\right)^2. \quad (24)$$

□

B.2.2 More Details for Estimating the Expectation with Unlabeled Nodes (Section 3.2)

Lemma 1 suggests that $\sum_{i=1}^{c} \mathbb{E}_x \left[\mathrm{Var}(x)\right]$ can be estimated by sampling labeled node pairs (h_1^i, h_2^i) from the same class i. However, due to the scarcity of data in the minority class, this can often be challenging. In Section 3.2, we address this issue by utilizing random data augmentation and defining nodes from different views as members of the same class, enabling us to sample node pairs for Equation 3 and perform embedding subtraction.

However, as we mentioned, the pseudo node pairs lack information regarding their class membership, making it impossible to assign appropriate $\frac{1}{\sqrt{2n_i}}$ values for different i in Equation 3. These coefficients play a crucial role in compensating for the minority class, making it imperative to reintroduce information about the class number when constructing variance regularization. Therefore, in the second step, we replace h in Equation 3 with the class center C^i, as demonstrated in Equation 4. As $C^i = \mu^i + e^i$ and the variance of e^i is proportional to $\frac{1}{n_i}$, using Equation 4 to estimate Lemma 1 can provide similar compensatory effects for the minority class. We present the proof for the aforementioned propositions below.

Proof Firstly, the $\sum_{i=1}^{c} \mathbb{E}_x \left[\mathrm{Var}(x)\right]$ can be decomposed as follows:

$$
\begin{aligned}
\sum_{i=1}^{c} \mathbb{E}_x \left[\mathrm{Var}(x)\right] &= \frac{1}{N} \sum_{x \in G} \sum_{i=1}^{c} \left(h(x)^T \frac{1}{\sqrt{2n_i}} \left(h_1^i - h_2^i\right) \right)^2 \\
&= \frac{1}{N} \sum_{k=1}^{n_j} \sum_{j=1}^{c} \sum_{i=1}^{c} \left(\left(\mu_k^j + \epsilon_k^j\right)^T \frac{1}{\sqrt{2n_i}} \left(h_1^i - h_2^i\right) \right)^2 \\
&= \frac{1}{N} \sum_{k=1}^{n_j} \sum_{j=1}^{c} \sum_{i=1}^{c} \left((\mu_k^j)^T \frac{1}{\sqrt{2n_i}} \left(\epsilon_1^i - \epsilon_2^i\right) + (\epsilon_k^j)^T \frac{1}{\sqrt{2n_i}} \left(\epsilon_1^i - \epsilon_2^i\right) \right)^2 \\
&= \frac{1}{2N} \sum_{k=1}^{n_j} \sum_{j=1}^{c} \sum_{i=1}^{c} \left((\mu_k^j)^T \frac{1}{\sqrt{n_i}} \left(\epsilon_1^i - \epsilon_2^i\right) + (\epsilon_k^j)^T \frac{1}{\sqrt{n_i}} \left(\epsilon_1^i - \epsilon_2^i\right) \right)^2 \\
&= \underbrace{\frac{1}{2N} \sum_{k=1}^{n_j} \sum_{j=1}^{c} \sum_{i=1}^{c} \left(\frac{1}{n_i} \left[(\mu_k^j)^T \left(\epsilon_1^i - \epsilon_2^i\right) \right]^2 \right)}_{\mathcal{T}_1} + \underbrace{\frac{1}{2N} \sum_{k=1}^{n_j} \sum_{j=1}^{c} \sum_{i=1}^{c} \left(\frac{1}{n_i} \left[(\epsilon_k^j)^T \left(\epsilon_1^i - \epsilon_2^i\right) \right]^2 \right)}_{\mathcal{T}_2} \\
&+ \underbrace{\frac{1}{N} \sum_{k=1}^{n_j} \sum_{j=1}^{c} \sum_{i=1}^{c} \left(\frac{1}{n_i} \left[(\mu_k^j)^T \left(\epsilon_1^i - \epsilon_2^i\right) (\epsilon_k^j)^T \left(\epsilon_1^i - \epsilon_2^i\right) \right] \right)}_{\mathcal{T}_3}
\end{aligned}
\tag{25}
$$

The above Equation can be decomposed into three parts, namely $\mathcal{T}_1$, $\mathcal{T}_2$, and $\mathcal{T}_3$. Notably, each of these parts is associated with weight $\frac{1}{n_i}$. This observation supports Theorem 1 and Lemma 1, which suggest that the variance of a model is highly dependent on the distribution of the dataset's sample size, and that the extent of sample imbalance can significantly increase the model's variance.

Note that $\epsilon^i \sim N(0, \Lambda^i)$, $e^i \sim N\left(0, \frac{1}{n_i}\Lambda^i\right)$, and so $\frac{1}{\sqrt{n_i}}\epsilon^i \sim N\left(0, \frac{1}{n_i}\Lambda^i\right)$. So, if we use the class center C^i to replace h in Equation 3, then we can get the following expression,

$$
\begin{aligned}
&\frac{1}{N}\sum_{x\in G}\sum_{i=1}^{c}\Big((C^i)^T h(x)-(C^{i'})^T h'(x)\Big)^2\\
=&\frac{1}{N}\sum_{k=1}^{n_j}\sum_{j=1}^{c}\sum_{i=1}^{c}\Big((\mu^i)^T(\epsilon_k^j-\epsilon_k^{j'})+(e^i)^T\epsilon_k^j-(e^{i'})^T\epsilon_k^{j'}\Big)^2\\
=&\frac{1}{N}\sum_{k=1}^{n_j}\sum_{j=1}^{c}\sum_{i=1}^{c}\Big((\mu^i)^T(\epsilon_k^j-\epsilon_k^{j'})+(\frac{1}{\sqrt{n_i}}\epsilon^i)^T\epsilon_k^j-(\frac{1}{\sqrt{n_i}}\epsilon^i)^T\epsilon_k^{j'}\Big)^2\\
=&\frac{1}{N}\sum_{k=1}^{n_i}\sum_{i=1}^{c}\sum_{j=1}^{c}\Big((\mu^j)^T(\epsilon_k^i-\epsilon_k^{i'})+(\frac{1}{\sqrt{n_j}}\epsilon^j)^T\epsilon_k^i-(\frac{1}{\sqrt{n_j}}\epsilon^j)^T\epsilon_k^{i'}\Big)^2\\
=&\frac{1}{N}\sum_{k=1}^{n_i}\sum_{i=1}^{c}\sum_{j=1}^{c}\Big((\mu^j)^T(\epsilon_k^i-\epsilon_k^{i'})+(\frac{1}{\sqrt{n_j}}\epsilon^j)^T(\epsilon_k^i-\epsilon_k^{i'})\Big)^2\\
=&\underbrace{\frac{1}{N}\sum_{k=1}^{n_i}\sum_{i=1}^{c}\sum_{j=1}^{c}\Big(\Big[(\mu^j)^T(\epsilon_k^i-\epsilon_k^{i'})\Big]^2\Big)}_{\mathcal{L}_1}+\underbrace{\frac{1}{N}\sum_{k=1}^{n_i}\sum_{i=1}^{c}\sum_{j=1}^{c}\Big(\frac{1}{n_j}\Big[(\epsilon^j)^T(\epsilon_k^i-\epsilon_k^{i'})\Big]^2\Big)}_{\mathcal{L}_2}\\
&+\underbrace{\frac{2}{N}\sum_{k=1}^{n_i}\sum_{i=1}^{c}\sum_{j=1}^{c}\Big(\frac{1}{\sqrt{n_j}}\Big[(\mu^j)^T(\epsilon_k^i-\epsilon_k^{i'})(\epsilon^j)^T(\epsilon_k^i-\epsilon_k^{i'})\Big]\Big)}_{\mathcal{L}_3}
\end{aligned}
\tag{26}
$$

Like the previous Equation 25, Equation 26 can be decomposed into three parts: $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$. If we substitute the class center C^i for h, we can observe that even though $\mathcal{L}_1$ is insensitive to class imbalance, variables $\mathcal{L}_2$ and $\mathcal{L}_3$, which respectively incorporate weights $\frac{1}{n_j}$ and $\frac{1}{\sqrt{n_j}}$, can still provide the following support: when optimizing the variance of the model, more attention can be given to the variance introduced by the minority classes, which also provides an innovative perspective for understanding class imbalance on graphs. □

C More Results

C.1 More results Under Traditional Semi-supervised Settings

In Table 3, we present the results of RVGNN and other algorithms on Cora-Semi (ρ=10). The experimental results demonstrate that TAM has a significant gain effect on all models (BalancedSoftmax, Renode, GraphENS), while GraphENS achieved the best performance among all current baselines on various models. By comparing with GraphSMOTE, we can conclude that addressing the node classification imbalance issue should focus on the topological characteristics of the graph. Finally, and most importantly, RVGNN achieved state-of-the-art results under all basic models, which also verifies the superiority of our model in traditional imbalanced segmentation.

Table 3: Experimental results of our method RVGNN and other baselines on ***Cora-Semi***. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures. Highlighted are the top **first** and **second**. **Δ** is the margin by which our method leads state-of-the-art method.

Dataset(*Cora-Semi*)	GCN		GAT		SAGE	
Imbalance Ratio ($\rho = 10$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
Vanilla	62.82 ± 1.43	61.67 ± 1.59	62.33 ± 1.56	61.82 ± 1.84	61.82 ± 0.97	60.97 ± 1.07
Re-Weight	65.36 ± 1.15	64.97 ± 1.39	66.87 ± 0.97	66.62 ± 1.13	63.94 ± 1.07	63.82 ± 1.30
PC Softmax	68.04 ± 0.82	67.84 ± 0.81	66.69 ± 0.79	66.04 ± 1.10	65.79 ± 0.70	66.04 ± 0.92
GraphSMOTE	66.39 ± 0.56	65.49 ± 0.93	66.71 ± 0.32	65.01 ± 1.21	61.65 ± 0.34	60.97 ± 0.98
BalancedSoftmax	69.98 ± 0.58	68.68 ± 0.55	67.89 ± 0.36	67.96 ± 0.41	67.43 ± 0.61	67.66 ± 0.69
+ TAM	69.94 ± 0.45	69.54 ± 0.47	69.16 ± 0.27	69.39 ± 0.37	69.03 ± 0.92	**69.03** ± 0.97
Renode	67.03 ± 1.41	67.16 ± 1.67	67.33 ± 0.79	68.08 ± 1.16	66.84 ± 1.78	67.08 ± 1.75
+ TAM	68.26 ± 1.84	68.11 ± 1.97	67.50 ± 0.67	68.06 ± 0.96	67.28 ± 1.11	67.15 ± 1.11
GraphENS	70.89 ± 0.71	70.90 ± 0.81	**70.45** ± 1.25	69.87 ± 1.32	68.74 ± 0.46	68.34 ± 0.33
+ TAM	**71.69** ± 0.36	**72.14** ± 0.51	70.15 ± 0.18	**70.00** ± 0.40	**70.45** ± 0.74	**70.40** ± 0.75
RVGNN	**72.92** ± 2.27	**72.60** ± 2.26	**74.56** ± 0.96	**74.61** ± 0.96	**73.32** ± 3.02	68.91 ± 3.13
Δ	**+ 1.23**	**+ 0.46**	**+ 4.11**	**+ 4.61**	**+ 2.87**	**- 1.49**

Table 4: Experimental results of our method RVGNN and other baselines on ***Computers-Random***. We report averaged balanced accuracy (bAcc.,%) and F1-score (%) with the standard errors over 5 repetitions on three representative GNN architectures. Highlighted are the top **first** and **second**. **Δ** is the margin by which our method leads state-of-the-art method.

Dataset(*Computers-Random*)	GCN		GAT		SAGE	
Imbalance Ratio ($\rho = 25.50$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
Vanilla	78.43 ± 0.41	77.14 ± 0.39	71.35 ±1.18	69.60 ± 1.11	65.30 ± 1.07	64.77 ± 1.19
Re-Weight	80.49 ± 0.44	75.07 ± 0.58	71.95 ± 0.80	70.67 ± 0.51	66.50 ± 1.47	66.10 ± 1.46
PC Softmax	81.34 ± 0.55	75.17 ± 0.57	70.56 ± 1.46	67.26 ± 1.48	69.73 ± 0.53	67.03 ± 0.6
GraphSMOTE	80.50 ± 1.11	73.79 ± 0.14	71.98 ± 0.21	67.98 ± 0.31	72.69 ± 0.82	68.73 ± 1.01
BalancedSoftmax	81.39 ± 0.25	74.54 ± 0.64	72.09 ± 0.31	68.38 ± 0.69	73.80 ± 1.06	69.74 ± 0.60
+ TAM	81.64 ± 0.48	75.59 ± 0.83	74.00 ± 0.77	70.72 ± 0.50	73.77 ± 1.26	71.03 ± 0.69
Renode	81.64 ± 0.34	**76.87** ± 0.32	72.80 ± 0.94	71.40 ± 0.97	70.94 ± 1.50	70.04 ± 1.16
+ TAM	80.50 ± 1.11	75.79 ± 0.14	71.98 ± 0.21	70.98 ± 0.31	72.69 ± 0.82	70.73 ± 1.01
GraphENS	82.66 ± 0.61	76.55 ± 0.17	75.25 ± 0.85	71.49 ± 0.54	77.64 ± 0.52	72.65 ± 0.53
+ TAM	**82.83** ± 0.68	76.76 ± 0.39	**75.81** ± 0.72	**72.62** ± 0.57	**78.98** ± 0.60	**73.59** ± 0.55
RVGNN	**85.00** ± 0.07	**82.35** ± 0.08	**81.94** ± 0.54	**80.94** ± 0.25	**80.61** ± 0.11	**77.49** ± 0.09
Δ	**+ 2.17**	**+ 5.48**	**+ 6.13**	**+ 8.32**	**+ 1.63**	**+ 3.90**

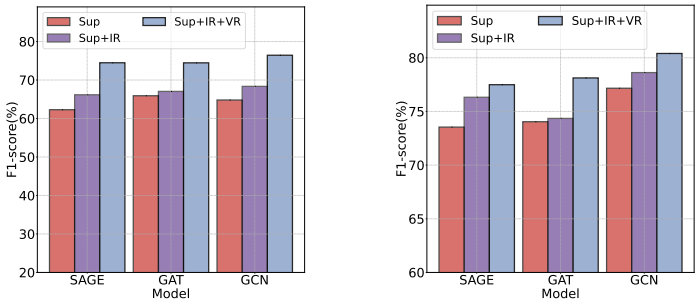
C.2 More results On Naturally Imbalanced Datasets

One of our major highlights is testing our algorithms on naturally imbalanced datasets, which is more representative of real-world scenarios. This means that in a semi-supervised setting, our training set and unlabeled data are both imbalanced, which poses a significant challenge for oversampling methods. In Table 4, we present the results of RVGNN and all baselines on ***Computers-Random***,

demonstrating that our model still achieves state-of-the-art performance. In contrast, oversampling methods such as GraphSMOTE perform poorly on this dataset due to overfitting limitations. We believe that our model's ability to regularize variance from a bottom-up perspective contributes to its superior performance on naturally imbalanced graphs, while also demonstrating its powerful generalization capabilities.

D More Analysis

D.1 More Ablation Analysis for the Loss Function



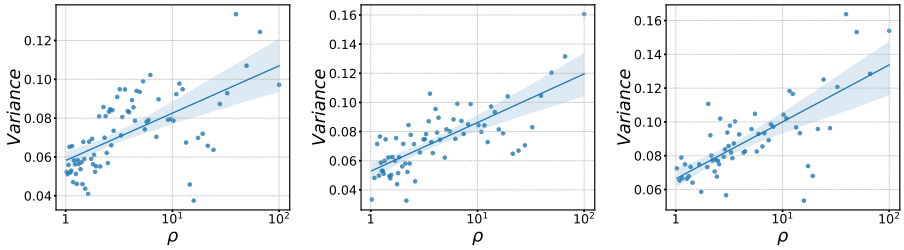
(a) *PubMed-Semi* (b) *Computers-Semi*

Figure 4: More Ablation Analysis for the Loss Function.

Details of Experimental Setup. We conduct more experiments on imbalance ratio 10 ($\rho = 10$) for the datasets PubMed and Amazon-Computers. For each GNN network, we add one loss regularization at a time, i.e. for GCN, we gradually add $\mathcal{L}_{\mathrm{IR}}$, $\mathcal{L}_{\mathrm{VR}}$ from the initial $\mathcal{L}_{\mathrm{sup}}$. The architecture we employed consisted of a 2-layers graph neural network (GNN) with 128 hidden dimensions, using GCN [16], GAT [34], and GraphSAGE [9]. The models were trained for 2000 epochs.

Analysis. Figure 4 reports more experiments for each component in the loss function. As we have seen, each component of the loss function can bring an improvement in the training effect. It is worth noting that in most cases $\mathcal{L}_{\mathrm{VR}}$ has a larger effect boost than $\mathcal{L}_{\mathrm{IR}}$. We believe that $\mathcal{L}_{\mathrm{VR}}$ plays a more important role in the loss function.

D.2 More Experiments for Variance and Imbalance Ratio Correlation in Theorem 1



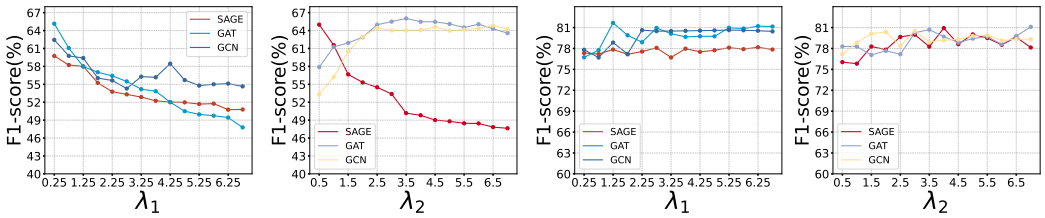
(a) *PubMed-GCN* (b) *PubMed-GAT* (c) *PubMed-SAGE*

Figure 5: More experiments for variance and imbalance ratio correlation in Theorem 1.

Details of Experimental Setup. In this experiment, the classifier is not MLP, which means that the probability of a node v being classified into class i depends on the distance of this node from the center of class i. We classify half of the classes as majority classes and the other half as minority classes. Initially, each class in majority classes and minority classes has 200 labeled nodes. To generate different imbalance scenarios (ρ), we reduce the number of labeled nodes in the minority class by 1 and add 1 to the number of labeled nodes in the majority class each time, so as to keep the number of samples in the training set consistent and eliminate the effect of the sample size variance in the training set. To calculate the variance of the model under a certain imbalance ratio (ρ), we repeat 20 times to randomly select different but the same number of training sets, and train 2000 epochs for each fixed training set. The architecture we employed consisted of a 2-layers graph neural network (GNN) with 128 hidden dimensions, using GCN [16], GAT [34], and GraphSAGE [9].

Analysis. Figure 5 reveals an intriguing pattern, as the majority of data points closely align along a regression curve exhibiting a positive slope. This observation provides substantial evidence that establishes a strong and direct linear relationship between the imbalance ratio (ρ) and the associated variance. Consequently, our findings provide compelling support for the hypothesis postulated in Theorem 1.

D.3 More Experiments for Hyperparameter Sensitivity Analysis of RVGNN



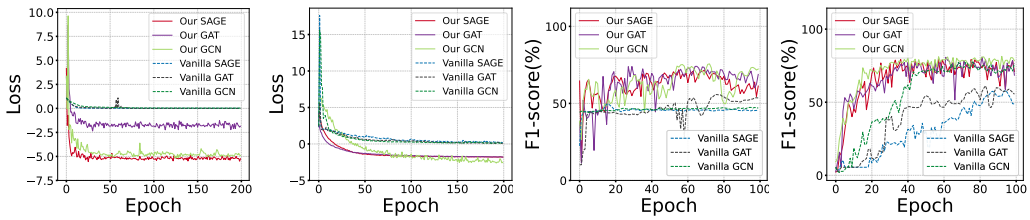
(a) *CiteSeer*-λ_1 (b) *CiteSeer*-λ_2 (c) *Computers*-λ_1 (d) *Computers*-λ_2

Figure 6: More Experiments of Hyperparameter Sensitivity Analysis for RVGNN.

Details of Experimental Setup. We implemented experiments on the two datasets, CiteSeer and Amazon-Computers. In this experiment, we evaluate the F1 score when we fix one λ while the other changes. The architecture we employed consisted of a 2-layer graph neural network (GNN) with 128 hidden dimensions, using GCN [16], GAT [34], and GraphSAGE [9]. The models were trained for 2000 epochs.

Analysis. In Figure 6, we present the sensitivity analysis of RVGNN to two weight hyperparameters (λ_1, λ_2) in the loss function on the CiteSeer and Amazon-Computers. It is evident that RVGNN exhibits varying degrees of sensitivity to hyperparameters on different datasets. Specifically, the model demonstrates lower sensitivity to a and b on CiteSeer, while the opposite is observed on Amazon-Computers. We postulate that the number of nodes (i.e., dataset size) may be a crucial factor in this discrepancy. Nevertheless, the optimal range of hyperparameter selection appears to be relatively narrow, indicating that significant efforts are not be required for model fine-tuning.

D.4 More Results of loss curve and F1 score



(a) *PubMed-Loss* (b) *Computers-Loss* (c) *PubMed-F1 Score* (d) *Computers-F1 score*

Figure 7: More analysis of loss curve and F1 score.

Details of Experimental Setup. We implemented experiments on three datasets PubMed, CiteSeer and Amazon-Computers. We compare our method RVGNN and other vanilla models (only using cross-entropy and not using graph augmentation). The architecture we employed for RVGNN and vanilla model consisted of a 2-layer graph neural network (GNN) with 128 hidden dimensions, using GCN [16], GAT [34], and GraphSAGE [9]. The models were trained for 2000 epochs.

Analysis. We plot the loss curve and F1 score with epoch on the PubMed and Computers datasets. By analyzing the plotted loss curves, we observe that our model demonstrates notable advantages, including faster convergence of the loss curves and improved performance across multiple datasets. For the F1 score, our model consistently outperforms alternative approaches, showcasing its superior capability in effectively capturing the complex relationships within the data and making accurate predictions.

E Elaboration of the experimental setup

In this section, we present our approach for constructing imbalanced datasets, describe our evaluation protocol, and provide comprehensive details on our algorithm as well as the baseline methods utilized in our study. To achieve this, we leverage sophisticated techniques and utilize advanced metrics to ensure the reliability and relevance of our results.

E.1 Construction for Imbalanced datasets

The detailed descriptions of the datasets are shown in Table 5. The details of label distribution in the training set of the five imbalanced benchmark datasets are in Table 7, and the label distribution of the full graph is provided in Table 7.

Table 5: Summary of the datasets used in this work.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Amazon-Computers	13,752	491,722	767	10
Coauthor-CS	18,333	163,788	6,805	15

Imbalanced datasets construction for Traditional Semi-supervised Settings. For each citation dataset, we adopt the "public" split and apply a random undersampling technique to make the class distribution imbalanced until the target imbalance ratio ρ is achieved. Specifically, we convert the minority class nodes to unlabeled nodes in a random manner. Regarding the co-purchased networks Amazon-Computers, we conduct replicated experiments by randomly selecting nodes as the training set in each trial. We create a random validation set that contains 30 nodes in each class, and the remaining nodes are used as the testing set.

Imbalanced datasets construction for Naturally Imbalanced Datasets. For ***Computers-Random*** and ***CS-Random***, we constructed a training set with equal proportions based on the label distribution of the complete graph (Amazon-Computers). The label distributions for the training sets of ***Computers-Random*** and ***CS-Random*** are presented in Table 7. To achieve this, we employed a stratified sampling approach that ensured an unbiased representation of all labels in the training set.

Table 6: Label distributions in the training sets

Dataset	$\mathcal{C}_0$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_9$	$\mathcal{C}_{10}$	$\mathcal{C}_{11}$	$\mathcal{C}_{12}$	$\mathcal{C}_{13}$	$\mathcal{C}_{14}$
Cora-Semi ($\rho = 10$)	**23.26%**	**23.26%**	**23.26%**	**23.26%**	**2.32%**	**2.32%**	**2.32%**	-	-	-	-	-	-	-	-
CiteSeer-Semi ($\rho = 10$)	**30.30%**	**30.30%**	**30.30%**	**3.03%**	**3.03%**	**3.03%**	-	-	-	-	-	-	-	-	-
PubMed-Semi ($\rho = 10$)	**47.62%**	**47.62%**	**4.76%**	-	-	-	-	-	-	-	-	-	-	-	-
Computers-Semi ($\rho = 10$)	**18.18%**	**18.18%**	**18.18%**	**18.18%**	**18.18%**	**1.82%**	**1.82%**	**1.82%**	**1.82%**	**1.82%**	-	-	-	-	-
Computers-Random ($\rho = 25.50$)	**3.01%**	**15.79%**	**10.53%**	**3.76%**	**38.35%**	**2.26%**	**3.01%**	**6.02%**	**15.79%**	**1.50%**	-	-	-	-	-
CS-Random ($\rho = 41.00$)	**3.98%**	**2.27%**	**11.36%**	**2.27%**	**7.39%**	**11.93%**	**1.70%**	**5.11%**	**3.98%**	**0.57%**	**7.95%**	**11.36%**	**2.27%**	**23.30%**	**4.55%**

Table 7: Label distributions on the whole graphs

Dataset	$\mathcal{C}_0$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$	$\mathcal{C}_8$	$\mathcal{C}_9$	$\mathcal{C}_{10}$	$\mathcal{C}_{11}$	$\mathcal{C}_{12}$	$\mathcal{C}_{13}$	$\mathcal{C}_{14}$
Cora ($\rho \approx 4.54$)	**351**	**217**	**418**	**818**	**426**	**298**	**180**	-	-	-	-	-	-	-	-
CiteSeer ($\rho \approx 2.66$)	**264**	**590**	**668**	**701**	**696**	**508**	-	-	-	-	-	-	-	-	-
PubMed ($\rho \approx 1.91$)	**4103**	**7739**	**7835**	-	-	-	-	-	-	-	-	-	-	-	-
Amazon-Computers ($\rho \approx 17.73$)	**436**	**2142**	**1414**	**542**	**5158**	**308**	**487**	**818**	**2156**	**291**	-	-	-	-	-
Coauthor-CS ($\rho \approx 35.05$)	**708**	**462**	**2050**	**429**	**1394**	**2193**	**371**	**924**	**775**	**118**	**1444**	**2033**	**420**	**4136**	**876**

E.2 Architecture of GNNs

We conducted evaluations using three classic GNN architectures, namely GCN [16], GAT [34], and GraphSAGE [9]. The GNN models were constructed with different numbers of layers, namely $L = 1, 2, 3$. To enhance the learning process, each GNN layer was accompanied by a BatchNorm layer with a momentum of 0.99, followed by a PRelu activation function [11]. For the GAT architecture, we employed multi-head attention with 8 heads. We search for the best model on the validation set. The available choices for the hidden unit sizes were 64, 128, and 256.

E.3 Evaluation Protocol

We utilized the Adam optimizer [15] with an initial learning rate of 0.01 or 0.005. To manage the learning rate, we employed a scheduler based on the approach outlined in [30], which reduced the learning rate by half when there was no decrease in validation loss for 100 consecutive epochs. Weight decay with a rate of 0.0005 was applied to all learnable parameters in the model. In the initial training iteration, we trained the model for 200 epochs using the original training set for Cora, CiteSeer, PubMed, or Amazon-Computers. However, for Flickr, the training was extended to 2000 epochs in the first iteration. Subsequently, in the remaining iterations, we trained the models for 2000 epochs using the aforementioned optimizer and scheduler. The best models were selected based on validation accuracy, and we employed early stopping with a patience of 300 epochs.

E.4 Technical Details of RVGNN

For all datasets ***Cora-Semi***, ***CiteSeer-Semi***, ***PubMed-Semi***, ***Computers-Semi***, ***Computers-Random*** and ***CS-Random***, the learning rate η is chosen from $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1\}$. The temperature hyperparameter τ is chosen from $\{0.05, 0.08, 0.13, 0.16, 0.21, 0.23, 0.26\}$. The threshold v is chosen from $\{0.6, 0.63, 0.66, 0.7, 0.8, 0.83, 0.9, 0.93, 0.96, 0.99\}$. The factor λ_1 of $\mathcal{L}_{\mathrm{VR}}$ is chosen from $\{0.25, 0.35, 0.5, 0.85, 1, 1.5, 2, 2.15, 2.65, 3\}$. The factor λ_2 of $\mathcal{L}_{\mathrm{IR}}$ is chosen from $\{0.35, 0.5, 1, 1.25, 1.5, 2.85, 3\}$. The factors of mask node properties and edges in $\tilde{G}$ are chosen from $\{0.4, 0.45, 0.5, 0.6, 0.65, 0.7\}$ and $\{0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7\}$. The factors of mask node properties and edges in $\tilde{G}'$ are chosen from $\{0.1, 0.15, 0.2, 0.3, 0.4, 0.45\}$ and $\{0.1, 0.15, 0.2, 0.3, 0.35, 0.4, 0.45\}$.

E.5 Technical Details of Baselines

For the GraphSMOTE method [47], we employed the branched algorithms whose edge predictions are discrete-valued, which have achieved superior performance over other variants in most experiments. Regarding the ReNode method [6], we search hyperparameters within the lower bound of cosine annealing, $w_{\min} \in 0.25, 0.5, 0.75$, and the upper bound range, $w_{\max} \in 1.25, 1.5, 1.75$, as suggested in [6]. The PageRank teleport probability was fixed at $\alpha = 0.15$, which is the default setting in the released codes. As for TAM [30], we performed a hyperparameter search for the coefficient of the ACM term, $\alpha \in 1.25, 1.5, 1.75$, the coefficient of the ADM term, $\beta \in 0.125, 0.25, 0.5$, and the minimum temperature of the class-wise temperature, $\phi \in 0.8, 1.2$, following the approach described in [30]. The sensitivity to the imbalance ratio of the class-wise temperature, δ, was fixed at 0.4 for all the main experiments. Additionally, consistent with [30], we implemented a warmup phase for 5 iterations, as we utilized model predictions for unlabeled nodes.

E.6 Configuration

All the algorithms and models are implemented in Python and PyTorch Geometric. Experiments are conducted on a server with an NVIDIA 3090 GPU (24 GB memory) and an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz.

F Algorithm

Algorithm 1 *RVGNN*

Input: Imbalanced Graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{V_L})$, feature matrix $\mathbf{X}$, adjacency matrix $\mathbf{A}$, unlabeled set $\mathbf{V_U} = \mathbf{V} - \mathbf{V_L}$, label matrix $\mathbf{Y}$, feature matrix of labeled nodes and unlabeled nodes $\mathbf{X_L}$ and $\mathbf{X_U}$, the number of classes $\mathbf{k}$, the threshold v for determining whether a node has confident prediction, temperature hyperparameter τ, GNN model f_θ, the classifier h_θ, learning rate η, The total number T of epochs for model training. The $\mathrm{sim}(\cdot,\cdot)$ computes the cosine similarity between two vectors, *CE* represents the cross-entropy loss function, and $\mathbb{I}(\cdot)$ is an indicator function.

for $t = 0, 1, \ldots, T$ **do**
  Generate two differently augmented views $\tilde{G} = (\tilde{\mathrm{A}}, \tilde{\mathrm{X}})$ and $\tilde{G}' = (\tilde{\mathrm{A}}', \tilde{\mathrm{X}}')$ from original graph $\mathbf{G}$.
  $\tilde{\mathbf{O}}_L \leftarrow f_\theta(\tilde{\mathrm{A}}, \tilde{\mathrm{X}}_L)$, $\tilde{\mathbf{O}}_U \leftarrow f_\theta(\tilde{\mathrm{A}}, \tilde{\mathrm{X}}_U)$
  $\tilde{\mathbf{O}}'_L \leftarrow f_\theta(\tilde{\mathrm{A}}', \tilde{\mathrm{X}}'_L)$, $\tilde{\mathbf{O}}'_U \leftarrow f_\theta(\tilde{\mathrm{A}}', \tilde{\mathrm{X}}'_U)$
  % Component 1: Supervised Loss
  $\mathcal{L}_{sup} \leftarrow \frac{1}{2} CE(h_\theta(\tilde{\mathbf{O}}_L), \mathrm{Y}) + \frac{1}{2} CE(h_\theta(\tilde{\mathbf{O}}'_U), \mathrm{Y})$
  % Component 2: Intra-Class Aggregation Regularization
  $\mathcal{L}_{\mathrm{IR}} \leftarrow -\frac{1}{N_U} \sum_{i=1}^{N_U} \mathrm{sim}(\tilde{\mathbf{O}}_i, \tilde{\mathbf{O}}'_i) - \frac{1}{N_{all}} (\sum_{i=1}^{N_L} \sum_{\substack{j=1 \\ (i,j)\in same}}^{N_L} \mathrm{sim}(\tilde{\mathbf{O}}_i, \tilde{\mathbf{O}}'_j) + \sum_{i=1}^{N_L} \sum_{\substack{j=1, j\neq i \\ (i,j)\in same}}^{N_L} \mathrm{sim}(\tilde{\mathbf{O}}_i, \tilde{\mathbf{O}}_j)$
  % Component 3: Variance-constrained Optimization with Adaptive Regularization
  for i = 1, 2, ..., k **do**
    Compute the class centers $\tilde{\mathbf{C}}_i$ and $\tilde{\mathbf{C}}'_i$ for class i
  end for
  ▷ Obtain the Label Probability for Each Node
  for $i = 1, 2, \ldots, |\mathbf{V}|$ **do**
    $\tilde{\mathbf{p}}_i \leftarrow \mathrm{Softmax}(\tilde{\mathbf{O}}_i \cdot \left[\tilde{\mathbf{C}}_1, \ldots, \tilde{\mathbf{C}}_k\right]^T)$
    $\tilde{\mathbf{p}}'_i \leftarrow \mathrm{Softmax}\left(\tilde{\mathbf{O}}'_i \cdot [\tilde{\mathbf{C}}'_1, \ldots, \tilde{\mathbf{C}}'_k\right]^T)$
  end for
  ▷ Eliminate Nodes with Low Confidence.
  $\mathbf{V}_{\text{conf}} \leftarrow \{i \mid \mathbb{I}(\max(\tilde{\mathbf{p}}'_i) > v) = 1, \forall i \in V_U\}$
  ▷ Replace Labeled Node Prediction
  for $i = 1, 2, \ldots, |\mathbf{V_L}|$ **do**
    $\tilde{\mathbf{p}}'_i \leftarrow \mathbf{Y}_i$
  end for
  $\mathcal{L}_{\mathbf{VR}} \leftarrow \frac{1}{|V_{\mathbf{conf}}|} \sum_{i\in V_{\mathbf{conf}}} CE(\tilde{\mathbf{p}}'_i, \tilde{\mathbf{p}}_i) + \frac{1}{|V_L|} \sum_{v_i \in V_L} CE(\mathbf{Y}_i, \tilde{\mathbf{p}}_i)$
  $\mathcal{L}_{Training} \leftarrow \mathcal{L}_{sup} + \lambda_1 \mathcal{L}_{VR} + \lambda_2 \mathcal{L}_{IR}$
  $\theta \leftarrow \theta - \eta(\nabla \mathcal{L}_{Training})$
end for

G Notations

Table 8: Notation table.

Indices	
n	The number of nodes, $n = \lvert\mathbf{V}\rvert$.
f	The node feature dimension.
k	The number of different classes.
D	The dimension of the embedding space, or the dimension of the last layer of GNNs.
Parameters	
$\mathbf{G}$	An undirected and unweighted graph.
$\mathbf{V}$	The node set of $\mathbf{G}$.
$\mathbf{E}$	The edge set of $\mathbf{G}$.
$\mathbf{X}$	The feature matrix of $\mathbf{G}$, $\mathbf{X} \in \mathbb{R}^{n \times f}$.
$\mathbf{V_L}$	The set of labeled nodes of $\mathbf{G}$.
$\mathbf{A}$	The adjacency matrix of $\mathbf{G}$, $\mathbf{A} \in \{0,1\}^{n \times n}$.
$\mathbf{N}(v)$	The set of 1-hop neighbors for node v.
$\mathbf{V_U}$	The set of unlabeled nodes, $\mathbf{V_U} := \mathbf{V} \setminus \mathbf{V_L}$.
$\mathbf{C}_i$	The labeled set for class i.
ρ	Imbalance ratio of a dataset, $\rho = \max_i \lvert\mathbf{C}_i\rvert \,/\, \min_i \lvert\mathbf{C}_i\rvert$.
$\tilde{\mathbf{G}}$	A view from the original graph $\mathbf{G}$, $\tilde{\mathbf{G}} = \left(\tilde{\mathbf{A}}, \tilde{\mathbf{X}}\right)$.
$\tilde{\mathbf{G}}'$	Another view from the original graph $\mathbf{G}$, $\tilde{\mathbf{G}}' = \left(\tilde{\mathbf{A}}', \tilde{\mathbf{X}}'\right)$.
h	Output of GNN network, $\mathrm{h} = f_\theta(\mathbf{A}, \mathbf{X})$.
$\mathcal{S}$	The probability distribution reference matrix, $\mathcal{S} := [C_1; C_2; \ldots; C_k]$.
C	The collection of class centers, $C := (C_1; C_2; \ldots; C_k)$.
p_i	The probability distribution for node i.
y_i	The one-hot label vector for node i.
$\mathcal{L}_{\text{sup}}$	Supervised loss.
$\mathcal{L}_{\text{IR}}$	Intra-class aggregation regularization.
$\mathcal{L}_{\text{VR}}$	Variance-constrained optimization with adaptive regularization.
$\mathcal{L}_{\text{final}}$	The sum of all losses.
V_{conf}	The set of nodes with confident predictions.
λ_1	The weight for $\mathcal{L}_{\text{VR}}$.
λ_2	The weight for $\mathcal{L}_{\text{IR}}$.
v	The threshold for determining whether a node has confident prediction.
τ	The temperature hyperparameter.
α	The size threshold of nodes being added in each class per round.
η	Learning rate of GNN model.
Functions	
$\mathbb{I}(\cdot)$	The indicater function.
$\mathrm{sim}(\cdot)$	The function that computes the cosine similarity between two vectors.
$CE(\cdot)$	The cross-entropy function.
f_θ	GNN model.