KLCE: REGULARIZED IMBALANCE NODE-CLASSIFICATION VIA KL-DIVERGENCE AND CROSS-ENTROPY

Mohammad T. Teimuri
Sharif University of Technology
m.teimuri96@sharif.edu

Zahra Dehghanian
Sharif University of Technology
zahra.dehghanian97@sharif.edu

Gholamali Aminian
Alan Turing Institute
gaminian@turing.ac.uk

Hamid R. Rabiee
Sharif University of Technology
rabiee@sharif.edu

ABSTRACT

This paper introduces a novel regularization based on KL-divergence and cross-entropy for imbalance node classification via Graph neural networks. We evaluate the performance of our approach on several benchmark datasets and compare it with state-of-the-art methods. The experimental results demonstrate the effectiveness of our proposed method in addressing imbalance node classification tasks.

1 INTRODUCTION AND RELATED WORK

Graph neural networks (GNNs) have emerged as a powerful tool for learning from graph-structured data. They have been successfully applied to various tasks, such as node classification, link prediction, and graph classification. However, in many real-world graphs, nodes are inherently class-imbalanced. This imbalance can lead to the GNNs being biased towards major classes, Park et al. (2022); Song et al. (2022), and the traditional GNN models underperformed. In this work, we address imbalance node classification, proposing a straightforward approach that seamlessly complements existing methods. We introduce a regularization based on KL divergence and cross-entropy. This regularization strategy is inspired by the insight that the underrepresented class samples are more valuable and should be emphasized during the learning process.

In recent years, significant research has addressed the challenge of imbalance node classification in graph neural networks (GNNs). We review the existing data-level and algorithm-level methods and focus on loss function engineering techniques that have been proposed to tackle this problem.

Data-level (Augmentation methods) methods in imbalanced node classification with graph neural networks aim to address class imbalance by manipulating the training data. These methods focus on generating synthetic samples or modifying existing ones to balance the class distribution. DR-GCN Shi et al. (2020) introduces conditional GAN to generate virtual nodes that are similar to adjacent node features of source nodes. GraphSMOTE Zhao et al. (2021) synthesizes the features of minor nodes by interpolating two minor nodes as SMOTE Chawla et al. (2002) does and determines the edges of synthesized nodes with edge predictor. Our work differs from this line of research as we just apply the regularization to the available training dataset.

Algorithm-level methods have been developed to incorporate class-awareness into the GNN architecture. GraphENS Park et al. (2022) synthesizes the whole ego network for the minor class by combining two different ego networks based on their similarity. Another approach tries to minimize the generalization bound and adjusts the softmax by considering the relative quantity ratio between two classes, a.k.a. BalancedSoftmax Ren et al. (2020). PC-Softmax Hong et al. (2021) is a correction algorithm that rewards minor classes only in the inference.

The paramount role of loss function engineering in addressing class imbalances within imbalanced node classification is evident. One common technique is to assign higher weights to the minority class samples during the training process. This can be achieved by designing loss functions and

encouraging a wider separation between the minority and majority classes. Re-Weight Japkowicz & Stephen (2002) is an example of such a technique. Furthermore, TAM Song et al. (2022) compensates minor classes in the training phase and adapts the loss function to create a larger margin between classes, thereby mitigating misclassification.

2 METHODOLOGY

Problem Formulation: We consider $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ as node-feature and labels, respectively, where $|\mathcal{Y}| = k$. We denote the learned distribution over classes as $P(\hat{\mathbf{Y}}|\mathbf{X}) := \{P(\hat{Y} = j|\mathbf{X})\}_{j=1}^{k}$ where $P(\hat{Y} = j|\mathbf{X}) := \frac{1}{n}\sum_{i=1}^{n} P(\hat{Y} = j|X_i)$ is the prediction of model for j-th class for given node-features $\{X_i\}_{i=1}^{n}$. The KL divergence $\mathrm{KL}(P\|Q)$ is given by $\mathrm{KL}(P\|Q) := \int_{\mathcal{Z}} \log\big(\frac{dP}{dQ}\big)dP$. We also define the cross-entropy between P and Q, as $H(P,Q) = -\int_{\mathcal{Z}} \log\big(dP\big)dQ$. Let us define $\mathbf{P}_k := \{P_j\}_{j=1}^{k}$ as the target class distribution.

Regularization: In our regularization loss, denoted as (**KLCE**), we incorporate two loss terms within our GNN model,

$$\text{KLCE} \;:=\; \lambda_{\text{KLCE}}\mathrm{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) + \mathrm{KL}(P(\hat{\mathbf{Y}}|\mathbf{X})\|\mathbf{P}_k), \tag{1}$$

where

$$\mathrm{KL}(P(\hat{\mathbf{Y}}|\mathbf{X})\|\mathbf{P}_k) = \sum_{j=1}^{k} P(\hat{Y} = j|\mathbf{X}) \log(P(\hat{Y} = j|\mathbf{X})/P_j), \quad \text{and}$$

$$\mathrm{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) = -\sum_{j=1}^{k} P(\hat{Y} = j|\mathbf{X}) \log(P_j),$$

is the KL-divergence and cross-entropy between the learned distribution across classes and the target class distribution, and $\lambda_{\text{KLCE}} \in (-1, 1)$ is a hyper-parameter for tuning between the KL-divergence and cross-entropy.

Regarding the target class distribution, we can assign more probability mass points to minority classes while reducing them for the majority classes. We can also consider the uniform distribution if the test dataset is balanced. Integrating these specific regularization terms during training encourages the model to acquire a distribution that aligns more closely with this target class distribution over classes. This method effectively tackles the class imbalance issue present in graph node classification. To the best of our knowledge, applying these particular regularization terms to address imbalance classification, especially in the context of node classification, represents a novel contribution.

3 EXPERIMENTS AND DISCUSSIONS

We conducted some experiments to test our KLCE regularization loss, inspired by the baselines in Song et al. (2022). Details of experiments and more results are available in Appendix B. We compared our KLCE with other baselines under the GCN model, (Kipf & Welling, 2016), in Table 1. We report averaged balanced accuracy (bAcc) and F1-score with the standard errors for 20 repetitions on the GCN model. The performance of our KLCE regularization loss under more GNN architectures, e.g., GAT and GraphSage, is studied in Appendix B. Our approach enhances the performance (accuracy and F1 scores) of several imbalance algorithms baselines, e.g., BalancedSoftmax and GraphENS, when integrated with them. Furthermore, a notable benefit is the reduction in variance across most of the conducted experiments.

Table 1: Experimental results of our Regularization method (KLCE) and other baselines on three class-imbalanced node classification benchmark datasets: Cora, CiteSeer and PubMed. We report averaged balanced accuracy (bAcc.) and F1-score with the standard errors for 20 repetitions on three representative GNN architectures, i.e., GCN, GAT and GraphSage

	Dataset	Cora		CiteSeer		PubMed	
	Imbalance Ratio ($\rho = 10$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Re-Weight	65.52 ±0.84	65.54 ±1.20	44.52 ±1.22	38.85 ±1.62	70.17 ±1.25	66.37 ±1.73
	PC Softmax	67.79 ±0.92	67.39 ±1.08	49.81 ±1.12	45.55 ±1.26	70.20 ±0.60	68.83 ±0.73
	DR-GCN	60.17 ±0.83	59.31 ±0.97	42.64 ±0.75	38.22 ±1.22	65.51 ±0.81	64.95 ±0.53
	GraphSMOTE	62.66 ±0.83	61.76 ±0.96	34.26 ±0.89	28.31 ±1.48	68.94 ±0.89	64.17 ±1.43
	+ **KLCE**	66.18 ±0.82	64.83 ±0.97	40.37 ±1.72	39.51 ±1.73	72.73 ±0.54	72.40 ±0.58
	Log-loss	32.37 ±0.38	43.66 ±0.75	35.35 ±0.17	23.01 ±0.15	63.60 ±0.09	47.93 ±0.10
	+ **TAM**	52.00 ±0.40	42.81 ±0.75	33.16 ±0.18	22.13 ±0.20	64.20 ±0.24	49.11 ±0.43
	+ **KLCE**	59.07 ±1.22	53.74 ±1.91	54.46 ±1.06	51.41 ±1.38	70.04 ±0.48	71.39 ±0.48
	+ **TAM** + **KLCE**	61.41 ±1.35	57.28 ±1.93	42.32 ±1.54	36.00 ±2.20	63.00 ±0.59	50.03 ±1.83
	BalancedSoftmax	64.90 ±1.05	61.25 ±1.28	51.15 ±1.00	46.66 ±1.30	69.33 ±0.68	63.71 ±1.42
	+ **TAM**	63.09 ±1.10	59.48 ±1.34	38.89 ±2.01	31.96 ±2.72	67.83 ±1.24	59.61 ±2.29
	+ **KLCE**	67.22 ±0.73	64.25 ±0.88	55.90 ±0.87	53.28 ±1.14	72.98 ±0.52	72.89 ±0.76
	+ **TAM** + **KLCE**	61.59 ±1.22	59.42 ±1.23	44.88 ±2.06	40.03 ±2.78	67.70 ±1.04	61.69 ±2.06
	GraphENS	69.69 ±0.52	69.77 ±0.60	53.62 ±1.12	50.08 ±1.39	71.15 ±0.80	67.92 ±1.15
	+ **TAM**	69.95 ±0.70	70.16 ±0.68	56.01 ±0.80	54.83 ±0.98	71.35 ±0.77	68.81 ±1.31
	+ **KLCE**	**72.32** ±0.62	71.57 ±0.62	57.47 ±0.93	55.98 ±1.03	72.89 ±0.49	72.76 ±0.54
	+ **TAM** + **KLCE**	71.89 ±0.61	**71.58** ±0.65	**59.12** ±0.83	**58.27** ±0.89	**73.81** ±0.42	**73.24** ±0.50

URM STATEMENT

The authors acknowledge that at least one key author of this work meets the URM criteria of the ICLR 2024 Tiny Papers Track.

REFERENCES

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Youngkyu Hong, Seungju Han, Kwanghee Choi, Seokjun Seo, Beomsu Kim, and Buru Chang. Disentangling label distribution for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6626–6636, 2021.

Nathalie Japkowicz and Shaju Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5):429–449, 2002.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

Joonhyung Park, Jaeyun Song, and Eunho Yang. Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *International Conference on Learning Representations, ICLR*. International Conference on Learning Representations (ICLR), 2022.

Jiawei Ren, Cunjun Yu, Xiao Ma, Haiyu Zhao, Shuai Yi, et al. Balanced meta-softmax for long-tailed visual recognition. *Advances in neural information processing systems*, 33:4175–4186, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Min Shi, Yufei Tang, Xingquan Zhu, David Wilson, and Jianxun Liu. Multi-class imbalanced graph convolutional network learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2020.

Jaeyun Song, Joonhyung Park, and Eunho Yang. Tam: topology-aware margin loss for class-imbalanced node classification. In *International Conference on Machine Learning*, pp. 20369–20383. PMLR, 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Tianxiang Zhao, Xiang Zhang, and Suhang Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 833–841, 2021.

A OTHER RELATED WORKS

Numerous Graph Neural Networks (GNNs) have emerged for non-euclidean graph-based tasks, spanning node, edge, and graph levels. GCNs, introduced in Kipf & Welling (2016), simplify the Cheby -Filter from Defferrard et al. (2016) for one-hop neighbors. GAT Veličković et al. (2017) computes the coefficient implicitly using a learnable attention mechanism. GraphSAGE Hamilton et al. (2017) stands for Graph Sampling and Aggregation, which generates node embeddings based on local network neighborhoods using neural networks. MPGNNs, as proposed in Gilmer et al. (2017), outline a general GNN framework, treating graph convolutions as message-passing among nodes and edges. For graph-level tasks, like graph classification, a typical practice involves applying a graph readout (pooling) layer after graph filtering layers composed of graph filters. We consider GCN, GAT, and GraphSage for comparability as our GNN model.

B EXPERIMENT

B.1 EXPERIMENT DETAILS

Datasets: We summarized the dataset statistics in Table 2. For our experiments, we consider citation network datasets—Citeseer, Cora and Pubmed (Sen et al., 2008)— where nodes are documents and edges are citation links. Chameleon and Squirrel Rozemberczki et al. (2021)- where nodes are Wikipedia's pages and edges are links between them, and Wisconson [1] is a graph of webpages crawled from the Internet by the Carnegie Mellon University.

Table 2: Dataset statistics.

Dataset	**Nodes**	**Edges**	**Classes**	**Features**
Citeseer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
Pubmed	19,717	44,338	3	500
Chameleon	2277	36101	5	2325
Squirrel	5201	217073	5	2089
Wisconsin	251	515	5	1703

Hyperparameters: We utilize two hyper-parameters λ and λ_{KLCE} for our regularization method when added to a baseline,

$$\begin{aligned} &\text{Baseline} + \lambda\text{KLCE} \\ &= \text{Baseline} + \lambda\big(\lambda_{\text{KLCE}}\text{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) + \text{KL}(P(\hat{\mathbf{Y}}|\mathbf{X})\|\mathbf{P}_k)\big) \end{aligned} \tag{2}$$

where KLCE as the regularization loss is added directly to the baseline, e.g., BalancedSoftmax (+TAM) or GraphENS(+TAM). We have used Optuna to find the best hyperparameter for each model and dataset. We chose λ in the range of $(0, 1)$ and λ_{KLCE} from the range of $(-1, 1)$. We have used Optuna under multiple settings and have found that the default searching setting works best in terms of the trade-off between optimal results and time complexity. We have used the Tree-structured Parzen Estimator, which is used by Optuna by default, and for the target function of Optuna, we have aimed to maximize the average of validation F1 score and accuracy.

Imbalance ratio: The distribution of train sets for each dataset can be seen in Table 3. All datasets follow a 10-imbalance ratio except Wisconsin, which has a ρ equal to 11.63. All the training

Target class distribution: For each dataset, the computation of P_k for the regularization term involves two steps. Firstly, we calculate the ratio of each class within all six datasets, as detailed in Table 3. Subsequently, we derive a normalized inverse ratio for each specific dataset for each class. The inversion is intended to emphasize the significance of the minority class, and normalization is applied to the output to transform it into a probability distribution.

[1] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

Table 3: Number of training nodes per class

Dataset ($\rho = 10$)	$\mathbf{L}_0$	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$	$\mathbf{L}_5$	$\mathbf{L}_6$
Cora	20	20	20	20	2	2	2
CiteSeer	20	20	20	2	2	2	-
PubMed	20	20	2	-	-	-	-
Chameleon	225	220	218	22	22	-	-
Squirrel	487	494	501	50	50	-	-
Wisconsin ($\rho = 11.63$)	4	38	50	5	5	-	-

Hardware and setup: All of our Experiments were conducted on a single server containing an Nvidia GTX 1080ti GPU, AMD Ryzen5 3600x CPU @ 3.80GHz and 32GB RAM. With this setup, all experiments were completed within one week.

Training: To choose the training mask, we use the training masks provided by Pytorch Geometric. We train each model for 200 epochs while finding the best hyper-parameters based on the Validation f1 score and balanced accuracy. After 30 trials, the best hyper-parameters found by Optuna are used for an accurate run with 2000 epochs.

Optimizer: Considering the regularization terms, we use Adaptive Moment Estimation (ADAM) from PyTorch as the optimizer. Also, ℓ_2 regularization with weight decay of $5e-4$ and dropout in some layers are used to prevent over-fitting. We use a learning rate of 0.01.

Baselines: For baselines, inspired by Song et al. (2022), we consider Re-weight Japkowicz & Stephen (2002), PC-Softmax Hong et al. (2021), DR-GCN (Shi et al., 2020), GraphSMOTE (Zhao et al., 2021), BalancedSoftmax Ren et al. (2020), GraphENS (Park et al., 2022), TAM (Song et al., 2022) and traditional Log-loss function.

B.2 More Experiments

The results for six datasets are presented in Tables 4 and 5. We can observe the many cases, and our KLCE approach outperforms other baselines when combined with them.

Table 4: Experimental results of our regularization methods and other baselines on three class-imbalanced node classification benchmark datasets: Chameleon, Squirrel and Wisconsin. We report averaged balanced accuracy (bAcc.) and F1-score with the standard errors for 20 repetitions on three representative GNN architectures, i.e., GCN, GAT and GraphSage

	Dataset	Chameleon		Squirrel		Wisconsin ($\rho = 11.63$)	
	Imbalance Ratio ($\rho = 10$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Re-Weight	36.07 $\pm$0.87	35.61 $\pm$0.81	26.92 $\pm$0.53	25.04 $\pm$0.59	44.13 $\pm$3.08	40.74 $\pm$3.27
	PC Softmax	36.86 $\pm$1.04	36.24 $\pm$1.01	26.49 $\pm$0.59	25.73 $\pm$0.49	30.90 $\pm$3.10	28.15 $\pm$2.16
	DR-GCN	33.34 $\pm$0.81	29.60 $\pm$0.79	23.34 $\pm$0.43	18.20 $\pm$0.49	29.44 $\pm$1.36	27.08 $\pm$1.37
	GraphSMOTE	41.50 $\pm$0.82	40.80 $\pm$0.79	27.14 $\pm$0.49	26.67 $\pm$0.53	45.36 $\pm$4.21	40.91 $\pm$4.39
	Log-loss	44.99 $\pm$0.87	37.34 $\pm$1.33	31.55 $\pm$0.24	23.76 $\pm$0.21	37.27 $\pm$1.45	33.14 $\pm$0.97
	+ **TAM**	33.02 $\pm$0.32	25.23 $\pm$0.31	27.23 $\pm$0.28	19.59 $\pm$0.33	39.75 $\pm$2.05	34.12 $\pm$1.43
	+ **KLCE**	53.16 $\pm$0.51	52.36 $\pm$0.56	37.21 $\pm$0.42	35.20 $\pm$0.50	48.75 $\pm$2.23	43.04 $\pm$1.49
	+ **TAM** + **KLCE**	33.56 $\pm$0.47	27.22 $\pm$0.81	26.52 $\pm$0.37	20.37 $\pm$0.34	42.83 $\pm$1.89	36.60 $\pm$1.13
	BalancedSoftmax	55.21 $\pm$1.99	55.07 $\pm$2.10	39.17 $\pm$0.35	37.35 $\pm$0.49	47.93 $\pm$1.89	41.77 $\pm$1.19
	+ **TAM**	33.28 $\pm$0.41	27.28 $\pm$0.56	27.46 $\pm$0.33	20.83 $\pm$0.37	**48.83** $\pm$0.33	40.74 $\pm$1.29
	+ **KLCE**	55.16 $\pm$0.54	55.34 $\pm$0.56	**40.03** $\pm$0.21	**38.19** $\pm$0.33	48.67 $\pm$2.03	**41.75** $\pm$1.49
	+ **TAM** + **KLCE**	34.10 $\pm$0.51	27.90 $\pm$0.59	26.94 $\pm$0.38	21.07 $\pm$0.40	47.83 $\pm$2.13	39.53 $\pm$1.36
	GraphENS	**56.89** $\pm$0.41	**56.87** $\pm$0.43	36.78 $\pm$0.46	35.95 $\pm$0.48	44.17 $\pm$2.13	39.38 $\pm$1.65
	+ **TAM**	41.33 $\pm$0.62	38.86 $\pm$0.78	26.94 $\pm$0.47	22.41 $\pm$0.69	42.96 $\pm$2.49	38.43 $\pm$1.66
	+ **KLCE**	**56.89** $\pm$0.45	56.84 $\pm$0.42	36.79 $\pm$0.36	35.84 $\pm$0.35	45.46 $\pm$2.53	40.97 $\pm$1.90
	+ **TAM** + **KLCE**	42.41 $\pm$0.54	40.27 $\pm$0.75	27.96 $\pm$0.50	23.31 $\pm$0.66	44.26 $\pm$2.47	39.73 $\pm$1.76
GAT	Re-Weight	35.72 $\pm$0.65	34.19 $\pm$0.74	25.79 $\pm$0.52	24.32 $\pm$0.62	42.15 $\pm$2.33	37.66 $\pm$2.27
	PC Softmax	38.32 $\pm$0.88	37.46 $\pm$0.84	26.52 $\pm$0.31	25.71 $\pm$0.44	41.89 $\pm$3.95	38.03 $\pm$3.35
	DR-GCN	34.84 $\pm$0.72	31.53 $\pm$0.86	24.69 $\pm$0.46	21.81 $\pm$0.42	33.93 $\pm$2.34	31.75 $\pm$2.50
	GraphSMOTE	40.18 $\pm$0.67	39.43 $\pm$0.76	27.10 $\pm$0.49	26.63 $\pm$0.63	40.77 $\pm$2.24	38.96 $\pm$2.48
	Log-loss	49.96 $\pm$0.69	46.79 $\pm$0.92	30.84 $\pm$0.33	23.30 $\pm$0.29	37.12 $\pm$1.47	32.77 $\pm$1.42
	+ **TAM**	42.33 $\pm$0.59	37.02 $\pm$0.92	27.62 $\pm$0.28	21.13 $\pm$0.35	35.94 $\pm$1.94	32.04 $\pm$1.39
	+ **KLCE**	53.52 $\pm$0.60	52.94 $\pm$0.65	34.80 $\pm$0.81	34.08 $\pm$0.85	**46.99** $\pm$2.04	**41.19** $\pm$1.24
	+ **TAM** + **KLCE**	44.17 $\pm$0.73	42.67 $\pm$1.03	27.65 $\pm$0.27	25.06 $\pm$0.40	41.03 $\pm$2.28	35.93 $\pm$1.43
	BalancedSoftmax	54.78 $\pm$0.35	54.36 $\pm$0.37	35.84 $\pm$0.91	34.90 $\pm$0.97	43.30 $\pm$1.68	39.01 $\pm$1.25
	+ **TAM**	45.92 $\pm$0.54	43.97 $\pm$0.89	27.90 $\pm$0.29	21.72 $\pm$0.46	44.10 $\pm$1.91	37.62 $\pm$1.24
	+ **KLCE**	54.73 $\pm$0.44	54.54 $\pm$0.47	35.75 $\pm$1.00	35.19 $\pm$0.99	42.96 $\pm$2.18	38.54 $\pm$1.62
	+ **TAM** + **KLCE**	46.63 $\pm$0.71	45.06 $\pm$1.04	28.25 $\pm$0.33	23.59 $\pm$0.56	40.74 $\pm$2.49	35.22 $\pm$1.68
	GraphENS	57.79 $\pm$0.61	57.80 $\pm$0.66	38.97 $\pm$0.65	38.56 $\pm$0.63	41.15 $\pm$1.94	39.27 $\pm$1.65
	+ **TAM**	48.74 $\pm$0.74	48.03 $\pm$0.85	28.60 $\pm$0.35	27.44 $\pm$0.70	42.30 $\pm$2.38	38.19 $\pm$1.77
	+ **KLCE**	**57.94** $\pm$0.58	**57.95** $\pm$0.62	**40.06** $\pm$0.47	**39.52** $\pm$0.46	40.65 $\pm$1.99	38.58 $\pm$1.78
	+ **TAM** + **KLCE**	50.54 $\pm$0.76	50.16 $\pm$0.80	30.31 $\pm$0.33	29.60 $\pm$0.36	42.05 $\pm$2.21	37.77 $\pm$1.62
GraphSage	Re-Weight	36.49 $\pm$1.21	34.84 $\pm$1.30	29.83 $\pm$0.59	25.88 $\pm$0.42	68.13 $\pm$3.19	63.45 $\pm$2.27
	PC Softmax	40.71 $\pm$0.82	39.95 $\pm$0.98	29.23 $\pm$0.50	28.19 $\pm$0.54	**70.57** $\pm$3.34	**67.13** $\pm$2.91
	DR-GCN	39.58 $\pm$0.58	38.37 $\pm$0.72	28.78 $\pm$0.50	25.01 $\pm$0.70	69.30 $\pm$1.99	64.60 $\pm$2.00
	GraphSMOTE	33.31 $\pm$0.63	30.83 $\pm$0.67	25.51 $\pm$0.43	19.79 $\pm$0.49	65.14 $\pm$3.84	62.53 $\pm$3.40
	Log-loss	44.55 $\pm$0.50	39.64 $\pm$0.67	27.72 $\pm$0.22	21.55 $\pm$0.18	43.36 $\pm$1.85	38.56 $\pm$1.50
	+ **TAM**	39.44 $\pm$0.49	33.42 $\pm$0.64	26.06 $\pm$0.28	20.15 $\pm$0.28	42.24 $\pm$1.74	38.73 $\pm$1.44
	+ **KLCE**	50.64 $\pm$0.47	50.77 $\pm$0.46	31.23 $\pm$0.27	30.94 $\pm$0.28	54.29 $\pm$2.12	52.03 $\pm$1.89
	+ **TAM** + **KLCE**	40.84 $\pm$0.70	37.97 $\pm$0.86	27.49 $\pm$0.41	22.82 $\pm$0.68	51.46 $\pm$1.87	50.23 $\pm$1.74
	BalancedSoftmax	51.57 $\pm$0.47	50.98 $\pm$0.51	32.05 $\pm$0.22	31.15 $\pm$0.24	59.86 $\pm$2.01	57.20 $\pm$1.58
	+ **TAM**	42.19 $\pm$0.53	39.33 $\pm$0.74	26.99 $\pm$0.45	22.43 $\pm$0.50	53.44 $\pm$2.01	50.82 $\pm$1.34
	+ **KLCE**	52.05 $\pm$0.46	52.11 $\pm$0.48	32.50 $\pm$0.26	32.25 $\pm$0.26	62.63 $\pm$2.18	58.13 $\pm$1.95
	+ **TAM** + **KLCE**	42.65 $\pm$0.57	40.49 $\pm$0.75	26.03 $\pm$0.30	22.07 $\pm$0.31	54.20 $\pm$1.95	50.89 $\pm$1.61
	GraphENS	53.27 $\pm$0.46	53.20 $\pm$0.47	31.98 $\pm$0.51	31.59 $\pm$0.53	64.98 $\pm$2.42	61.26 $\pm$2.15
	+ **TAM**	48.73 $\pm$0.60	47.90 $\pm$0.63	30.46 $\pm$0.39	30.14 $\pm$0.47	53.73 $\pm$2.99	48.67 $\pm$1.94
	+ **KLCE**	**53.79** $\pm$0.48	**53.74** $\pm$0.50	**32.58** $\pm$0.47	**32.34** $\pm$0.48	66.03 $\pm$2.32	61.40 $\pm$2.02
	+ **TAM** + **KLCE**	49.51 $\pm$0.52	48.84 $\pm$0.53	31.23 $\pm$0.43	30.77 $\pm$0.47	54.80 $\pm$3.01	49.29 $\pm$1.95

Table 5: Experimental results of our Regularization method (KLCE) and other baselines on three class-imbalanced node classification benchmark datasets: Cora, CiteSeer and PubMed. We report averaged balanced accuracy (bAcc.) and F1-score with the standard errors for 20 repetitions on three representative GNN architectures, i.e., GCN, GAT and GraphSage

	Dataset	Cora		CiteSeer		PubMed	
	Imbalance Ratio ($\rho = 10$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Re-Weight	65.52 ±0.84	65.54 ±1.20	44.52 ±1.22	38.85 ±1.62	70.17 ±1.25	66.37 ±1.73
	PC Softmax	67.79 ±0.92	67.39 ±1.08	49.81 ±1.12	45.55 ±1.26	70.20 ±0.60	68.83 ±0.73
	DR-GCN	60.17 ±0.83	59.31 ±0.97	42.64 ±0.75	38.22 ±1.22	65.51 ±0.81	64.95 ±0.53
	GraphSMOTE	62.66 ±0.83	61.76 ±0.96	34.26 ±0.89	28.31 ±1.48	68.94 ±0.89	64.17 ±1.43
	+ **KLCE**	66.18 ±0.82	64.83 ±0.97	40.37 ±1.72	39.51 ±1.73	72.73 ±0.54	72.40 ±0.58
	Log-loss	52.57 ±0.38	43.66 ±0.75	35.33 ±0.17	23.01 ±0.15	63.60 ±0.09	47.93 ±0.10
	+ **TAM**	52.00 ±0.40	42.81 ±0.75	33.16 ±0.18	22.13 ±0.20	64.20 ±0.24	49.11 ±0.43
	+ **KLCE**	59.07 ±1.22	53.74 ±1.91	54.46 ±1.06	51.41 ±1.38	70.04 ±0.48	71.39 ±0.48
	+ **TAM** + **KLCE**	61.41 ±1.35	57.28 ±1.93	42.32 ±1.54	36.00 ±2.20	63.00 ±0.59	50.03 ±1.83
	BalancedSoftmax	64.90 ±1.05	61.23 ±1.28	51.13 ±1.00	46.66 ±1.30	69.33 ±0.68	63.71 ±1.42
	+ **TAM**	63.09 ±1.10	59.48 ±1.34	38.89 ±2.01	31.96 ±2.72	67.83 ±1.24	59.61 ±2.29
	+ **KLCE**	67.22 ±0.73	64.25 ±0.88	55.90 ±0.87	53.28 ±1.14	72.98 ±0.52	72.89 ±0.76
	+ **TAM** + **KLCE**	61.59 ±1.22	59.42 ±1.23	44.88 ±2.06	40.03 ±2.78	67.70 ±1.04	61.69 ±2.06
	GraphENS	69.69 ±0.52	69.77 ±0.60	53.62 ±1.12	50.08 ±1.39	71.15 ±0.80	67.92 ±1.15
	+ **TAM**	69.95 ±0.70	70.16 ±0.68	56.01 ±0.80	54.83 ±0.98	71.35 ±0.77	68.81 ±1.31
	+ **KLCE**	**72.32** ±0.62	71.57 ±0.62	57.47 ±0.93	55.98 ±1.03	72.89 ±0.49	72.76 ±0.54
	+ **TAM** + **KLCE**	71.89 ±0.61	**71.58** ±0.65	**59.12** ±0.83	**58.27** ±0.89	**73.81** ±0.42	**73.24** ±0.50
GAT	Re-Weight	66.72 ±0.80	66.52 ±1.06	45.59 ±1.73	39.43 ±2.03	69.13 ±1.25	64.81 ±1.70
	PC Softmax	67.02 ±0.65	66.57 ±0.89	50.70 ±1.73	47.14 ±1.85	72.20 ±0.49	70.95 ±0.82
	DR-GCN	59.30 ±0.76	57.79 ±1.03	44.04 ±1.26	39.44 ±1.76	69.56 ±1.01	68.49 ±0.71
	GraphSMOTE	56.50 ±0.71	54.27 ±1.09	44.94 ±1.36	41.63 ±1.78	62.86 ±0.53	53.00 ±1.17
	+ **KLCE**	60.83 ±1.21	59.21 ±1.22	55.33 ±1.90	54.64 ±2.24	66.24 ±0.71	63.46 ±1.22
	Log-loss	50.69 ±0.36	43.45 ±0.57	34.49 ±0.21	22.58 ±0.15	61.15 ±0.34	45.98 ±0.26
	+ **TAM**	49.05 ±0.60	43.24 ±0.86	33.17 ±0.33	22.07 ±0.24	61.86 ±0.29	47.75 ±0.34
	+ **KLCE**	65.47 ±0.73	63.36 ±0.76	51.83 ±1.08	49.54 ±1.31	68.66 ±0.71	68.42 ±0.74
	+ **TAM** + **KLCE**	61.03 ±1.60	55.09 ±2.26	42.58 ±1.52	35.35 ±2.38	62.91 ±0.90	56.44 ±1.64
	BalancedSoftmax	62.27 ±0.87	58.42 ±1.21	43.50 ±1.68	37.58 ±2.14	69.30 ±0.75	64.75 ±1.40
	+ **TAM**	62.66 ±1.14	59.66 ±1.52	44.37 ±1.29	38.13 ±1.75	65.48 ±1.02	57.13 ±1.73
	+ **KLCE**	63.79 ±1.15	61.96 ±1.08	51.01 ±1.20	48.40 ±1.45	70.49 ±0.77	69.90 ±0.93
	+ **TAM** + **KLCE**	62.30 ±1.52	59.87 ±1.58	46.05 ±1.32	43.54 ±1.56	67.17 ±1.22	66.43 ±1.56
	GraphENS	70.07 ±0.62	68.79 ±0.77	52.29 ±0.91	48.84 ±1.23	71.86 ±0.80	69.17 ±1.13
	+ **TAM**	70.06 ±0.77	68.71 ±0.90	56.02 ±0.87	54.47 ±1.10	72.23 ±0.67	69.86 ±1.05
	+ **KLCE**	70.92 ±0.64	**70.06** ±0.64	**57.24** ±0.81	**56.55** ±0.97	72.91 ±0.47	72.30 ±0.52
	+ **TAM** + **KLCE**	**71.12** ±0.63	69.96 ±0.74	55.98 ±0.83	54.79 ±0.96	**72.93** ±0.43	**72.58** ±0.50
GraphSage	Re-Weight	63.76 ±0.98	63.46 ±1.22	46.64 ±1.92	41.38 ±2.76	69.03 ±1.17	64.01 ±2.18
	PC Softmax	64.03 ±0.81	63.73 ±0.99	50.14 ±1.89	47.38 ±2.13	71.39 ±0.84	70.25 ±1.02
	DR-GCN	61.05 ±1.17	60.17 ±1.23	46.00 ±0.93	47.73 ±1.12	69.23 ±0.68	67.35 ±0.90
	GraphSMOTE	65.37 ±0.55	65.13 ±0.68	38.94 ±1.05	33.60 ±1.68	64.15 ±0.55	54.00 ±1.14
	+ **KLCE**	**71.51** ±0.81	**71.01** ±0.89	48.49 ±1.16	48.54 ±1.26	70.00 ±0.61	69.20 ±0.75
	Log-loss	49.54 ±0.16	39.45 ±0.34	34.12 ±0.10	21.73 ±0.12	61.36 ±0.07	45.89 ±0.08
	+ **TAM**	50.94 ±0.29	41.94 ±0.60	34.91 ±0.21	22.63 ±0.17	62.67 ±0.34	47.68 ±0.51
	+ **KLCE**	55.87 ±1.86	55.05 ±1.78	34.67 ±0.18	22.21 ±0.23	62.18 ±0.33	48.98 ±1.31
	+ **TAM** + **KLCE**	54.43 ±1.31	46.27 ±2.03	36.73 ±0.94	26.09 ±1.62	62.38 ±0.52	47.22 ±1.58
	BalancedSoftmax	53.34 ±0.88	46.81 ±1.40	35.90 ±1.04	26.76 ±1.39	63.58 ±0.82	51.48 ±1.91
	+ **TAM**	53.34 ±0.88	46.81 ±1.40	34.16 ±1.04	24.98 ±1.32	63.97 ±0.82	52.36 ±1.95
	+ **KLCE**	51.97 ±1.22	49.58 ±1.27	41.52 ±1.45	37.08 ±1.62	66.72 ±0.67	60.14 ±1.49
	+ **TAM** + **KLCE**	51.68 ±1.63	49.45 ±1.63	39.73 ±1.39	35.87 ±1.63	67.39 ±0.62	62.64 ±1.21
	GraphENS	65.85 ±0.72	65.96 ±0.77	53.32 ±0.89	51.05 ±1.15	70.55 ±0.53	69.55 ±0.66
	+ **TAM**	67.04 ±0.63	66.93 ±0.69	51.87 ±1.09	49.11 ±1.37	70.46 ±0.66	67.23 ±0.93
	+ **KLCE**	68.86 ±0.61	68.58 ±0.68	55.60 ±0.70	55.20 ±0.76	71.06 ±0.58	71.00 ±0.63
	+ **TAM** + **KLCE**	69.29 ±0.56	68.96 ±0.55	**56.96** ±0.75	**55.86** ±0.87	**71.80** ±0.45	**71.74** ±0.50