Class-Imbalanced Graph Learning without Class Rebalancing

Zhining Liu [1] **Ruizhong Qiu** [1] **Zhichen Zeng** [1] **Hyunsik Yoo** [1] **David Zhou** [1] **Zhe Xu** [1] **Yada Zhu** [2]
Kommy Weldemariam [3] **Jingrui He** [1] **Hanghang Tong** [1]

Abstract

Class imbalance is prevalent in real-world node classification tasks and poses great challenges for graph learning models. Most existing studies are rooted in a *class-rebalancing* (CR) perspective and address class imbalance with class-wise reweighting or resampling. In this work, we approach the root cause of class-imbalance bias from an topological paradigm. Specifically, we theoretically reveal two fundamental phenomena in the graph topology that greatly exacerbate the predictive bias stemming from class imbalance. On this basis, we devise a lightweight topological augmentation framework BAT to mitigate the class-imbalance bias *without class rebalancing*. Being orthogonal to CR, BAT can function as an *efficient plug-and-play module* that can be seamlessly combined with and significantly boost existing CR techniques. Systematic experiments on real-world imbalanced graph learning tasks show that BAT can deliver up to 46.27% performance gain and up to 72.74% bias reduction over existing techniques. Code, examples, and documentations are available at https://github.com/ZhiningLiu1998/BAT.

1. Introduction

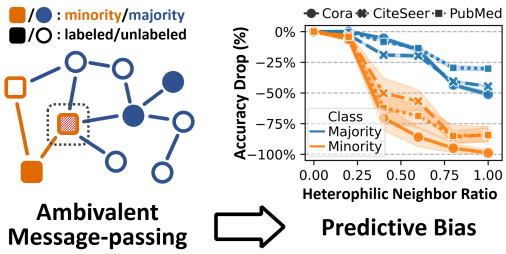
Node classification stands as one of the most fundamental tasks in graph machine learning, holding significant relevance in various real-world applications (Akoglu et al., 2015; Tang & Liu, 2010). Graph Neural Networks (GNNs) have demonstrated great success in tackling related tasks due to their robust representation learning capabilities (Song et al., 2022b; Fu & He, 2021). However, real-world graphs are often inherently class-imbalanced, i.e., the sizes of unique classes vary significantly, and a few majority classes have overwhelming numbers in the training set. In *Class-Imbalanced Graph Learning* (CIGL), GNNs are prone to suffer from severe performance degradation on minority class nodes (Park et al., 2022). This results in a pronounced predictive bias characterized by a large performance disparity between the majority and minority classes.

Traditional imbalance-handling techniques rely on *class rebalancing (CR)* such as class reweighting and resampling (Chawla et al., 2002; Cui et al., 2019), which works well for non-graph data. Recent studies propose more graph-specific CR strategies tailored for CIGL, e.g., neighborhood-aware reweighting (Li et al., 2022; Huang et al., 2022) and oversampling (Zhao et al., 2021b; Park et al., 2022). Nonetheless, these works are restricted to the class-rebalancing paradigm. Parallel to class imbalance, another emerging line of research studies topology imbalance, characterized by "the asymmetric topological properties of the *labeled nodes*" (Chen et al., 2021). It is considered an orthogonal problem to class imbalance, and hence, few work theoretically investigates how topological structure affects the learning on class imbalanced graphs. To fill this gap, we conduct an in-depth analysis of the role that topology plays in class-imbalanced graph learning. We theoretically show that topological differences between minority and majority classes significantly amplify the class imbalance bias, imposing great challenges to CIGL. This reveal an unexplored avenue that limits the performance of existing CIGL techniques: mitigating *class imbalance bias arising from imbalanced topology structure* through topological operations. Following this novel perspective, we devise a lightweight practical solution for CIGL that can be seamlessly combined with and further boost existing CR techniques.

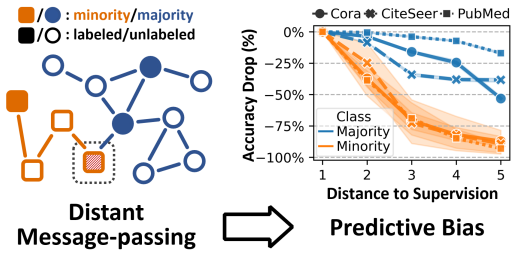
In this work, we formally define and theoretically investigate two fundamental *local topological phenomena* that greatly hinder CIGL: **(i)** *ambivalent message-passing* (AMP), i.e., high ratio of non-self-class neighbors in the node receptive field, and **(ii)** *distant message-passing* (DMP), i.e., poor connectivity with self-class labeled nodes. Intuitively, AMP leads to a higher influx of noisy information and DMP leads to poor reception of effective supervision signals in message-passing. Both result in lower signal-to-noise ratios and thus induce higher classification errors. Our theoretical finding reveals that the minority class is inherently more susceptible

[1]University of Illinois Urbana-Champaign [2]IBM Research [3]Amazon Science. Correspondence to: Hanghang Tong <htong@illinois.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.



(a) **L**: concept of AMP. **R**: relative performance loss with respect to the non-self-class neighbor ratio.



(b) **L**: concept of DMP. **R**: relative performance loss w.r.t. distance to the nearest same-class labeled node.

Figure 1. Concepts of *ambivalent message-passing* (AMP) and *distant message-passing* (DMP) and their impact in real-world imbalanced node classification tasks (Park et al., 2022). Both factors lead to a substantial increase in prediction errors, and further, **a larger performance disparity/bias** (i.e., the gap between the blue and orange curves) between the majority and minority classes.

to both AMP and DMP (Theorem 2.1 & 2.2), which leads to a more pronounced predictive bias. Such bias induced by the graph topology escalates as the level of class imbalance increases. We emphasize that AMP/DMP is defined for all nodes based on the local neighborhood, while influence conflict has no formal definition and influence insufficiency is defined only on labeled nodes with global PageRank score (Chen et al., 2021). To distinguish from them, we use the terminology AMP/DMP instead. Further discussions can be found in Section 5. Fig. 1 visually illustrates the concepts of AMP and DMP, highlighting their distinct impacts on the predictive performance of majority and minority classes.

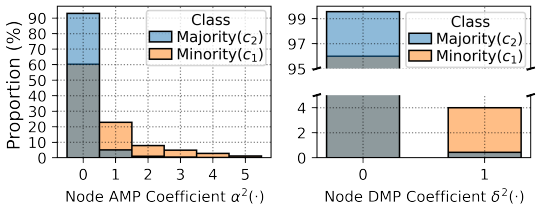
Following our theoretical and empirical findings, we devise BAT (BAlanced Topological augmentation), a model-agnostic and efficient technique to mitigate class imbalance bias in CIGL via topological augmentation. BAT dynamically locates and rectifies nodes critically influenced by AMP and DMP during learning, thereby effectively reducing the errors and biases in CIGL. Being orthogonal to class rebalancing, our solution is able to work hand-in-hand with existing techniques based on reweighting (Japkowicz & Stephen, 2002; Chen et al., 2021) and resampling (Zhao et al., 2021b; Park et al., 2022) and further boost their performance. Systematic experiments on real-world CIGL tasks show that BAT delivers significant performance boost (up to 46.27%) and bias reduction (up to 72.74%) over various CIGL baselines with diverse GNN architectures.

Our contributions: **(i) Novel Perspective.** We demonstrate the feasibility of taming class-imbalance bias *without* class rebalancing, which provides a new avenue that is orthogonal to the predominant class-rebalancing practice in CIGL. **(ii) Theoretical Insights.** We theoretically reveal the topological difference between minority and majority classes and its role in shaping predictive bias in CIGL, shedding light on future CIGL research. Empirical results validate our findings. **(iii) Practical Solution.** Motivated by theoretical and empirical finding, we devise a *lightweight and versatile* framework BAT to handle topological challenges in CIGL. Being complementary to class rebalancing, it can be seamlessly combined with and significantly boost existing CIGL techniques. **(iv) Empirical Study.** Systematic experiments and analysis across a diverse range of real-world tasks and GNN architectures show that BAT consistently demonstrates superior performance in both *promoting classification* and *mitigating predictive bias*.

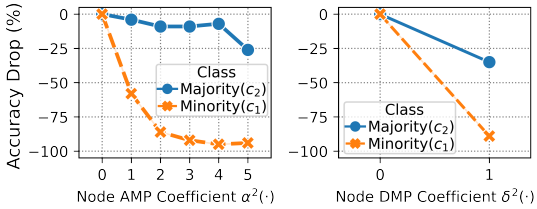
2. Class Imbalance and Local Topology

In this section, we delve into the impact of graph topology on the predictive bias in class-imbalanced node classification. We theoretically unveil that compared to the majority class, the minority class is inherently more susceptible to both Ambivalent Message Passing (AMP) and Distant Message Passing (DMP). This significantly worsens minority-class performance and leads to a more pronounced predictive bias stemming from class imbalance. After that, we present an empirical analysis to validate our theoretical findings, and to provide insights on how to mitigate the bias induced by AMP and DMP in practice. Detailed proofs can be found in Appendix A.

Theoretical analysis on local topology. Consider a graph $\mathcal{G} : (\mathcal{V}, \mathcal{E})$ from a stochastic block model (Holland et al., 1983) SBM(n, p, q), where n is the total number of nodes, p and q are the intra- and inter-class node connection probability. To facilitate analysis, we call node u *homo-connected* to node v if there is a path $[u, v_1, ..., v_k, v]$ where $v_1, ..., v_k, v$ are of the same class, and let $\mathcal{H}(u, k)$ denote the set of k-hop homo-connected neighbors of u. For binary node classification, we denote the number of nodes of class i as n_i ($n_1 + n_2 = n$); without loss of generality, let class 1/2 be the minority/majority class (thus $n_1 \ll n_2$). We denote class i's node set as $\mathcal{V}_i$ and labeled node set as $\mathcal{V}_i^{\text{L}}$ ($\subset \mathcal{V}_i$). For asymptotic analysis, we adopt conventional assumptions: $n_1 \cdot p = \beta + \mathcal{O}(\frac{1}{n})$ (i.e., β is the average intra-class



(a) Distribution of node AMP/DMP coefficients.



(b) Impact of AMP/DMP on predictive performance.

Figure 2. Node-level distribution of AMP and DMP coefficients and their impact on learning.

node degree of class 1); $p/q = \mathcal{O}(1)$ (Decelle et al., 2011).

We now give formal definitions of AMP and DMP. For a node u from class i, we define its **(i)** *k-hop AMP coefficient* $\alpha^k(u) \in [0, \infty)$ as the ratio of the expected number of non-self-class nodes to self-class nodes in its k-hop neighborhood $\mathcal{H}(u,k)$, i.e., $\alpha^k(u) := \frac{|\{v|v \notin \mathcal{V}_i, v \in \mathcal{H}(u,k)\}|}{|\{v|v \in \mathcal{V}_i, v \in \mathcal{H}(u,k)\}|}$; **(ii)** *$k$-hop DMP coefficient* $\delta^k(u) \in \{0,1\}$ as the indicator of whether all labeled nodes in its k-hop neighborhood are NON-self-class, i.e., $\delta^k(u) := \mathbb{1}(L_i^k(u) = 0, \Sigma_j L_j^k(u) > 0)$, where $L_j^k(u) = |\{v|v \in \mathcal{V}_j^{\mathrm{L}}, v \in \mathcal{H}(u,k)\}|$. For an intuitive example, the target node (marked by the dashed box) in Fig. 1(a) has $\alpha^1(u) = 3/1, \delta^1(u) = 0$ and node in Fig. 1(b) has $\alpha^1(u) = 1/1, \delta^1(u) = 1$. Further, to characterize the level of AMP/DMP for different *class*, for class i we define $\alpha_i^k := \frac{\mathbb{E}_{u \in \mathcal{V}_i}[|\{v|v \notin \mathcal{V}_i, v \in \mathcal{H}(u,k)\}|]}{\mathbb{E}_{u \in \mathcal{V}_i}[|\{v|v \in \mathcal{V}_i, v \in \mathcal{H}(u,k)\}|]}$ and $\delta_i^k := \mathbb{P}(\delta^k(u) = 1)$, where u is a node of class i. Intuitively, a higher α_i or δ_i indicates that class i is more susceptible to AMP or DMP. Building on these metrics, we analyze the disparities in α and δ between minority and majority classes, thereby providing insights into how the *graph topology induces additional class-imbalance bias*.

To simplify notation, we define imbalance ratio $\rho := n_2/n_1$. The larger the ρ is, the more imbalanced the dataset is. Then for AMP, we have the following Theorem 2.1.

Theorem 2.1 (AMP-sourced bias)**.** *For a large n, the ratio of AMP coefficients α for the minority class to the majority class grows polynomially with the imbalance ratio ρ and exponentially with k:*

$$\frac{\alpha_1^k}{\alpha_2^k} = \left(\rho \cdot \frac{\sum_{t=1}^k (\rho\beta)^{t-1}}{\sum_{t=1}^k \beta^{t-1}}\right)^2 + \mathcal{O}\Big(\frac{1}{n}\Big). \tag{1}$$

Proof. Please see Appendix A.2. □

Theorem 2.1 shows that the same-class neighbor proportion of minority-class nodes is significantly smaller than that of majority-class nodes, i.e., the minority class is more susceptible to AMP. As the imbalance ratio ρ increases, this issue becomes even more pronounced and introduces a higher bias into the learning process. Moving on to DMP, we have the following theorem 2.2.

Theorem 2.2 (DMP-sourced bias)**.** *Let $r_i^{\mathrm{L}} := \frac{|\mathcal{V}_i^{\mathrm{L}}|}{|\mathcal{V}_i|}$ denote the label rate of class i. For a large n, the ratio of DMP coefficients δ of the minority class over the majority class grows exponentially with ρ:*

$$\frac{\delta_1^k}{\delta_2^k} \approx \frac{1 - r_1^{\mathrm{L}}}{1 - r_2^{\mathrm{L}}} \mathrm{e}^{(\rho-1)\beta} + \mathcal{O}\Big(\frac{1}{n}\Big). \tag{2}$$

Proof. Please see Appendix A.3. □

Similarly, the result shows that the minority class exhibits a significantly higher susceptibility to DMP than the majority class. Theorem 2.2 also has several interesting implications: **(i)** *The imbalance ratio greatly affects the bias induced by DMP,* as δ_1^k/δ_2^k grows exponentially with ρ. **(ii)** *Labeling more minority-class nodes can mitigate, but hardly solve the problem.* Enlarging the minority-class label rate r_1^{L} can linearly shrink δ_1^k/δ_2^k, but it can hardly eliminate the bias induced by DMP (i.e., to have $\delta_1^k \leq \delta_2^k$) as $\mathrm{e}^{(\rho-1)\beta}$ is usually very large in practice. Take the *Cora* dataset (Sen et al., 2008) as an example (let class 1/class 2 denote the smallest/largest class): eliminating the DMP bias requires the minority-class label rate $r_1^{\mathrm{L}} \geq 1 - \frac{1-r_2^{\mathrm{L}}}{\mathrm{e}^{(\rho-1)\beta}} > 1 - 5.05 \times 10^{-8}$, which is practically infeasible.

Our theoretical findings show that both AMP and DMP affect the minority and majority classes differently, and the difference is primarily determined by the imbalance ratio ρ. However, directly manipulating ρ is tricky in practice as it requires sampling new nodes and edges from an unknown underlying graph generation model, or at least, simulating the process by oversampling.

A closer look at AMP & DMP in practice. To verify the theoretical results, and to provide more insights on how to mitigate the bias brought about by AMP and DMP in practice, we conduct a fine-grained empirical analysis on a real-world task. Results are detailed in Fig. 2[1]. Starting from Fig. 2(a), we can first observe that the minority class 1 has a larger proportion of nodes with high α or δ than the majority class 2, i.e., minority class 1 has higher average α and δ (specifically, $\alpha_1/\alpha_2 = 1.357/0.179$,

[1]Results obtained by training a GCN on the *PubMed* dataset.

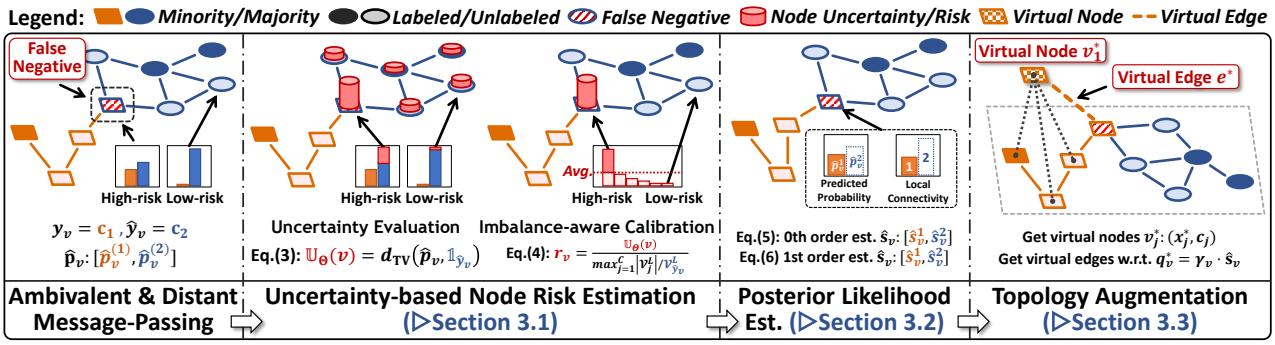


Figure 3. The proposed BAT (BAlanced Topological augmentation) framework, best viewed in color.

$\delta_1/\delta_2 = 0.040/0.004$), which is consistent with our theoretical findings. Further, Fig. 2(b) shows that both AMP and DMP significantly reduce the prediction accuracy, especially for the minority class that inherently has poorer data representation. This can be explained from a graph signal denoising perspective (Nt & Maehara, 2019): AMP introduces additional noise from dissimilar nodes, and DMP leads to less efficient label propagation/denoising, thus their impact is particularly significant on minority classes that are more susceptible to noise (Johnson & Khoshgoftaar, 2019) due to poor representation in the feature space. We further notice an intriguing fact that, at sample-level, the impact of AMP/DMP is concentrated on a small fraction of minority class nodes with large α or δ (e.g., the $\alpha \geq 1$ / $\delta = 1$ part in Fig. 2(a)). In other words, one can surrogate the tricky manipulation of ρ and directly mitigate AMP/DMP by *locating and rectifying a small number of critical nodes*, and this exactly motivates our subsequent studies.

3. Handling Class Imbalance from a Topological Perspective

Armed with the findings from Section 2, we now discuss how to devise a practical strategy to mitigate the error and bias induced by graph topology in CIGL. Earlier analyses have shown that this can be achieved by identifying and rectifying the critical nodes that are highly influenced by AMP/DMP. This naturally poses two challenging questions: **(i)** How can critical nodes be located as the direct calculation of α/δ using ground-truth labels is not possible? **(ii)** Subsequently, how can critical nodes be rectified and minimize the negative impact caused by AMP and DMP?

In answering the above questions, we devise a *lightweight* framework BAT (BAlanced Topological augmentation) for handling the topology-sourced errors and biases in CIGL. Specifically, for locating the misclassified nodes, BAT leverages model-based prediction uncertainty to assess the risk of potential misclassification caused by AMP/DMP for each node (§ 3.1). Then to rectify a misclassified node, we estimate a posterior likelihood of each node being in each class (§ 3.2) and dynamically augment the misclassified node's topological context based on our risk scores and posterior likelihoods (§ 3.3) thereby mitigating the impact of AMP and DMP. An overview of the proposed BAT framework is shown in Fig. 3.

3.1. Node Misclassification Risk Estimation

We now elaborate on the technical details of BAT. As discussed earlier, our first step is to locate the critical nodes that are highly influenced by AMP/DMP. Given the unavailability of ground-truth labels, direct computation of AMP/DMP coefficient is infeasible in practice. Fortunately, recent studies have shown that conflicting or lack of information from the neighborhood can disturb GNN learning and the associated graph-denoising process for affected nodes (Nt & Maehara, 2019; Wu et al., 2019; Ma et al., 2021). This further yields high vacuity or dissonance uncertainty (Stadler et al., 2021; Zhao et al., 2020) in the prediction. This motivates us to exploit the *model prediction uncertainty* to estimate nodes' risk of being misclassified due to AMP/DMP.

Uncertainty quantification. While there exist many techniques for uncertainty quantification (e.g., Bayesian-based (Zhang et al., 2019; Hasanzadeh et al., 2020), Jackknife sampling (Kang et al., 2022a)), they often either have to modify the model architecture, and/or impose additional computational overhead. In this study, we aim to streamline the design of BAT for optimal efficiency and adaptability. To this end, we employ an efficient and highly effective approach to uncertainty quantification. Formally, let C be the number of classes. For a node v, consider model $F(\cdot;\Theta)$'s predicted probability vector $\hat{\boldsymbol{p}}_v = F(\boldsymbol{A}, \boldsymbol{X}; \Theta)_v$, i.e., $\hat{p}_v^{(j)} = \mathbb{P}(y_v = j | \boldsymbol{A}, \boldsymbol{X}, \Theta)$. Let $\hat{y}_v$ be the predicted label. We measure the uncertainty score $\mathbb{U}_\Theta(v)$ by the total variation (TV) distance:

$$\mathbb{U}_\Theta(v) := d_{\text{TV}}(\hat{\boldsymbol{p}}_v, \mathbb{1}_{\hat{y}_v}) = \tfrac{1}{2}\sum_{j=1}^{C} |\hat{\boldsymbol{p}}_v^{(j)} - \mathbb{1}_{\hat{y}_v}^{(j)}| \in [0, 1]. \quad (3)$$

Intuitively, a node has higher uncertainty if the model is less confident about its current prediction. We remark that this

metric can be naturally replaced by other uncertainty measures (e.g., information entropy or more complex ones) with additional computation cost, yet the impact on performance is marginal. Please refer to the ablation study provided in Appendix C.1 for more details.

Imbalance-calibrated misclassification risk. Due to the lack of training instances, minority classes generally exhibit higher uncertainty. Therefore, using $\mathbb{U}_\Theta(\cdot)$ directly as the risk score would treat most minority-class nodes as high-risk, which is contrary to our intention of rectifying the false negatives (i.e., minority nodes wrongly predicted as majority-class) that cause bias in CIGL. To cope with this, we propose *imbalance-aware calibration* for misclassification risk scores. For each class i, let $\hat{\mathcal{V}}_i := \{u \in \mathcal{V} | \hat{y}_u = i\}$ and $\hat{\mathcal{V}}_i^{\text{L}} := \{u \in \mathcal{V}^{\text{L}} | y_u = i\}$. For node v with predicted label $\hat{y}_v$, we define its risk r_v as:

$$r_v := \frac{\mathbb{U}_\Theta(v)}{\max_{j=1}^{C} |\mathcal{V}_j^{\text{L}}|/|\mathcal{V}_{\hat{y}_v}^{\text{L}}|} \in [0, 1]. \tag{4}$$

Intuitively speaking, Eq. (4) calibrates v's prediction uncertainty by a *label imbalance score* $\max_{j=1}^{C} |\mathcal{V}_j^{\text{L}}|/|\mathcal{V}_{\hat{y}_v}^{\text{L}}|$. Minority classes with smaller labeled sets $\mathcal{V}_i^{\text{L}}$ will be discounted more.

Empirical validation. We validate the effectiveness of the proposed node risk assessment method, as shown in Fig. 4. The results indicate that our approach can accurately estimate node misclassification risk across various real-world CIGL tasks while enjoying computational efficiency.

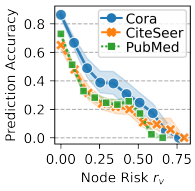


Figure 4. The negative correlation between the estimated node risk (x-axis) and the prediction accuracy (y-axis). We apply 10 sliding windows to compute the mean and deviation of the accuracy.

3.2. Posterior Likelihood Estimation

With the estimated risk scores of being affected by AMP/DMP, we move to the next question: how to rectify high-risk nodes with topological augmentation? As high-risk nodes are prone to misclassification, their true labels are more likely to be among the non-predicted classes $j \neq \hat{y}_v$. This motivates us to investigate schemes to harness information from these non-predicted classes. Since uniformly drawing from all classes probably introduces noise to learning, we propose to estimate the *posterior likelihood* $\hat{s}_v^{(j)}$ that a high-risk node v belongs to each class j after observing the current predictions. To estimate $\hat{s}_v^{(j)}$, we introduce a *zeroth-order* scheme and a *first-order* scheme with $\mathcal{O}(|\mathcal{V}|C)$ and $\mathcal{O}(|\mathcal{E}|C)$ time complexity, respectively. Please refer to §4 for a practical complexity analysis. We do not employ higher-order schemes due to the $\mathcal{O}(|\mathcal{V}|^{k-1}|\mathcal{E}|C)$ time complexity of the kth-order scheme.

Zeroth-order estimation. A natural approach is to utilize the predicted probabilities $\hat{p}_v^{(j)}$. As we have shown, the predicted label $\hat{y}_v$ of a high-risk node v is very likely to be wrong. Thus, we define the posterior likelihood $\hat{s}_v^{(j)}$ as the conditional probability given that the class is not $\hat{y}_v$, i.e.,

$$\begin{aligned} \hat{s}_v^{(j)} &:= \mathbb{P}_{y \sim \hat{\boldsymbol{p}}_v}[y = j | y \neq \hat{y}_v] \\ &= \begin{cases} \hat{p}_v^{(j)}/(1 - \hat{p}_v^{(\hat{y}_v)}), & \text{if } j \neq \hat{y}_v, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Intuitively, the zeroth-order posterior likelihoods $\hat{s}_v^{(j)}$ are consistent with the predicted probabilities $\hat{p}_v^{(j)}$ except for the wrongly predicted label $j = \hat{y}_v$. This can be computed efficiently on GPU in matrix form.

First-order estimation via random walk. We further explore the *local topology* for $\hat{s}_v^{(j)}$ estimation. Since neighboring nodes on a homophily graph tend to share labels, we consider a 1-step random walk starting from node v. Let $\mathcal{N}(v)$ be the neighboring node set of v, and let $v' \sim \mathcal{N}(v)$ denote the ending node of the random walk. We define $\hat{s}_v^{(j)}$ as the conditional probability that v' is predicted as class j given that v' is not predicted as class $\hat{y}_v$, i.e.,

$$\begin{aligned} \hat{s}_v^{(j)} &:= \mathbb{P}_{v' \sim \mathcal{N}(v)}[\hat{y}_{v'} = j | \hat{y}_{v'} \neq \hat{y}_v] \\ &= \begin{cases} \frac{|\{v' \in \mathcal{N}(v) | \hat{y}_{v'} = j\}|}{|\mathcal{N}(v)| - |\{v' \in \mathcal{N}(v) | \hat{y}_{v'} = \hat{y}_v\}|}, & \text{if } j \neq \hat{y}_v, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

With first-order estimation, $\hat{s}_v^{(j)}$ is proportional to the label frequency among adjacent nodes. Different from the zeroth-order scheme, this scheme relies on both node-level predictions and local connectivity patterns. The computation can be done via sparse matrix operation with $\mathcal{O}(|\mathcal{E}|C)$ time complexity. As a remark, although this scheme can extend to k-step random walks, we do not employ them due to the $\mathcal{O}(|\mathcal{V}|^{k-1}|\mathcal{E}|C)$ complexity of exact computation and the high variance of stochastic computation.

Empirical validation. Figure 5 compares the two schemes in practice. Results show that all high-risk ($r_v > 0$) minority nodes are misclassified, and both schemes can effectively find alternatives with significantly higher chances to be the ground truth class for high-risk nodes.

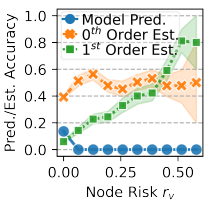


Figure 5. The minority-class accuracy of model prediction $\hat{y}_v = F(\boldsymbol{A}, \boldsymbol{X}; \Theta)$, and max-likelihood-based candidate selection $\hat{y}_v^s = \arg\max(\hat{\boldsymbol{s}}_v)$, on PubMed dataset. Note that this is just an illustrative example using $\arg\max(\hat{\boldsymbol{s}}_v)$. In practice, we consider the whole $\hat{\boldsymbol{s}}_v$ when sampling virtual edges, as described in Section 3.3.

3.3. Virtual Topology Augmentation

Finally, we discuss how to mitigate AMP and DMP via topology augmentation using our node risk scores and the posterior likelihoods. The general idea is to augment the local topology of high-risk nodes so as to integrate information from nodes that share similar patterns (mitigate AMP), even if they are not closely adjacent to each other in the graph topology (mitigate DMP), thus achieving less-biased CIGL. A straightforward way is to connect high-risk nodes to nodes from high-likelihood classes in the original graph. However, this can be problematic in practice as a massive number of possible edges could be generated, greatly disturbing the original topology structure.

To achieve efficient augmentation without disrupting the graph topology, we create virtual nodes (one per class) as "shortcuts" connecting to high-risk nodes according to posterior likelihoods. These shortcuts aggregate and pass class information to high-risk nodes from nodes that exhibit similar patterns (even if they are distant in the original graph), thus mitigating both AMP and DMP. Formally, for each class j, we build a virtual node v_j^* with feature $\boldsymbol{x}_{v_j^*} := \sum_{v\in\hat{\mathcal{V}}_j} \boldsymbol{x}_v/|\hat{\mathcal{V}}_j|$ and label $y_{v_j^*} := j$, and compute the average risk $\bar{r}_j := \sum_{v\in\hat{\mathcal{V}}_j} r_v/|\hat{\mathcal{V}}_j|$. Then for each node v, we connect a virtual edge between v and virtual node v_j^* with probability proportional to the posterior likelihood $\hat{s}_v^{(j)}$. However, if the connection probability is exactly $\hat{s}_v^{(j)}$, there will be many unnecessary virtual edges for low-risk nodes. Hence, we introduce a discount factor γ_v based on risk scores and connect the virtual edge with probability $q_v^{(j)} := \gamma_v \hat{s}_v^{(j)}$. To design the optimal γ_v, we propose to solve the following constrained quadratic program:

$$\min_{\boldsymbol{\gamma}\geq\mathbf{0}} \left(-\sum_{v\in\mathcal{V}} (r_v - \bar{r}_{\hat{y}_v})\gamma_v + \frac{1}{2}\|\boldsymbol{\gamma}\|_2^2 \right), \tag{7}$$

where the first term encourages virtual edges for high-risk nodes, and the second term is to minimize the number of virtual edges. The closed-form solution is $\gamma_v = \max(r_v - \bar{r}_{\hat{y}_v}, 0)$ (Antoniadis & Fan, 2001), which avoids virtual edges for low-risk nodes as we desire. We now summarize the procedure of BAT in Algorithm 1.

Complexity Analysis of BAT. BAT with 0th/1st-order estimation **scales linearly** with the number of nodes/edges (i.e., with $\mathcal{O}(|\mathcal{V}|C)$ or $\mathcal{O}(|\mathcal{E}|C)$ complexity). This makes BAT highly efficient and allows dynamically graph augmentation in each training step. Specifically, BAT introduces C (the number of class, usally small) virtual nodes with $\mathcal{O}(n)$ edges. Because of the long-tail distribution of node uncertainty and the discount factor used to solve Eq. (7), only a small portion of nodes have positive risks with relatively few (empirically around 1-3%) virtual edges introduced. We provide the scalability results of BAT later in the experiment section. In short, BAT takes milliseconds for a single topological augmentation. Please refer to Table 3 and the corresponding discussion for further details. We also discuss how to further speedup BAT in practice in C.2.

Algorithm 1 BAT: BAlanced Topological augmentation

Require: Class-imbalanced graph $\mathcal{G} : \{\boldsymbol{A}, \boldsymbol{X}\}$;
Initialize: node classifier $F(\cdot;\Theta)$;
while not converged **do**
  $\hat{\boldsymbol{P}} \leftarrow F(\boldsymbol{A}, \boldsymbol{X}; \Theta)$;
  $\hat{\boldsymbol{y}} \leftarrow \text{argmax}_{axis=1}(\hat{\boldsymbol{P}})$; ▷ get predictions $\hat{\boldsymbol{y}}$
  $\boldsymbol{r} \leftarrow \texttt{NodeRiskEst}(\hat{\boldsymbol{P}}, \hat{\boldsymbol{y}})$; ▷ Eq. (3) - (4)
  $\hat{\boldsymbol{S}} \leftarrow \texttt{PosteriorEst}(\boldsymbol{A}, \hat{\boldsymbol{P}}, \hat{\boldsymbol{y}})$; ▷ Eq. (5) - (6)
  for class $j = 1$ to C **do**
    $\boldsymbol{x}_{v_j^*} \leftarrow \sum_{v\in\hat{\mathcal{V}}_j} \boldsymbol{x}_v/|\hat{\mathcal{V}}_j|$;
    $v_j^* : (\boldsymbol{x}_{v_j^*}, j)$ ▷ virtual node v_j^* for class j
  end for
  $\mathcal{V}^* \leftarrow \{v_j^* | 1 \leq j \leq C\}$ ▷ virtual node set $\mathcal{V}^*$
  $\boldsymbol{Q}^* \leftarrow \hat{\boldsymbol{S}} \odot \boldsymbol{\gamma}$ ▷ virtual link prob. $\boldsymbol{Q}^*$ by Eq. (7)
  $\mathcal{E}^* \sim \boldsymbol{Q}^*$; ▷ sample virtual edges $\mathcal{E}^*$ w.r.t $\boldsymbol{Q}^*$
  Derive $\boldsymbol{X}^*, \boldsymbol{A}^*$ from $\mathcal{V}\cup\mathcal{V}^*, \mathcal{E}\cup\mathcal{E}^*$;
  Update Θ with augmented graph $\mathcal{G}^* : \{\boldsymbol{A}^*, \boldsymbol{X}^*\}$;
end while
Return: a balanced node classifier $F(\boldsymbol{A}, \boldsymbol{X}; \Theta)$;

4. Experiments

We carry out systematic experiments and analysis to validate BAT in the following aspects: **(i) Effectiveness** in both *promoting imbalanced node classification* and *mitigating the prediction bias* between different classes. **(ii) Versatility** in cooperating with and further boosting various CIGL techniques and GNN backbones. **(iii) Robustness** to extreme class imbalance. **(iv) Efficiency** in real-world applications.

Experiment Protocol. We validate BAT on five benchmark datasets for semi-supervised node classification, including the *Cora, CiteSeer, PubMed* from Plantoid graphs (Sen et al., 2008), and larger-scale *CS, Physics* from co-author networks (Shchur et al., 2018) with high-dimensional features. Following the same setting as prior studies (Park et al., 2022; Song et al., 2022a; Zhao et al., 2021b), we select half of the classes as minority. The imbalance ratio $\rho = n_{max}/n_{min} \geq 1$ is the ratio between the size of the largest class to the smallest class, i.e., more imbalance $\Leftrightarrow$ higher IR. Detailed data statistics and class distributions can be found in Appendix B.1. We test BAT with six CIGL techniques (Park et al., 2022; Chen et al., 2021; Zhao et al., 2021b; Chawla et al., 2002; Japkowicz & Stephen, 2002) and three GNN backbones (Veličković et al., 2018; Hamilton et al., 2017; Welling & Kipf, 2016) under all possible combinations to fully validate BAT's effectiveness and versatility in practice. Note that although there are other techniques

Table 1. BAT significantly boosts existing CIGL techniques, achieving better classification performance (Balanced Acc./Marco-F1) with reduced bias (PerfStd). For each CIGL baseline, we report its performance before and after collaborating with BAT. The average and the best score of all CIGL methods under three settings (base, +BAT_0, +BAT_1) are also provided, with **performance gain of BAT highlighted by** Δ. Due to space limitation, we report the key results and omit the error bar, full results can be found in Appendix D.

Metric			Balanced Acc.↑									Macro-F1↑		PerfStd↓	
CIGL Baseline			ERM	RW	RN	RS	SM	GS	GE	Avg (Δ)	Best (Δ)	Avg (Δ)	Best (Δ)	Avg (Δ)	Best (Δ)
Cora	GCN	BASE	61.6	67.7	66.6	59.5	58.3	68.0	70.1	64.5	70.1	63.7	70.0	25.7	20.0
		+BAT_0	65.5	71.0	71.4	72.5	72.2	68.5	72.2	**70.5 (+5.9)**	**72.5 (+2.4)**	**69.2 (+5.5)**	**71.6 (+1.7)**	**16.7 (-9.0)**	**14.4 (-5.6)**
		+BAT_1	69.8	72.1	71.8	74.2	73.9	71.6	72.6	**72.3 (+7.8)**	**74.2 (+4.1)**	**71.1 (+7.5)**	**72.8 (+2.9)**	**17.6 (-8.0)**	**15.2 (-4.8)**
	GAT	BASE	61.5	66.9	66.8	57.8	58.8	64.7	69.8	63.8	69.8	63.1	70.0	26.0	20.1
		+BAT_0	66.3	71.8	72.1	71.9	70.5	69.3	70.6	**70.4 (+6.6)**	**72.1 (+2.4)**	**69.0 (+6.0)**	**70.9 (+0.9)**	**17.2 (-8.9)**	**15.1 (-5.0)**
		+BAT_1	70.1	71.6	70.3	73.3	72.2	71.1	71.0	**71.4 (+7.6)**	**73.3 (+3.5)**	**70.2 (+7.1)**	**72.3 (+2.4)**	**18.0 (-8.0)**	**17.3 (-2.8)**
	SAGE	BASE	59.2	63.8	65.3	57.8	58.8	61.6	68.8	62.2	68.8	60.9	68.2	27.1	19.8
		+BAT_0	66.2	70.1	71.3	71.2	70.3	69.9	69.8	**69.8 (+7.7)**	**71.3 (+2.5)**	**68.9 (+8.0)**	**70.4 (+2.2)**	**16.4 (-10.7)**	**13.3 (-6.5)**
		+BAT_1	66.5	71.1	71.5	73.0	73.0	72.3	71.9	**71.4 (+9.2)**	**73.0 (+4.2)**	**70.1 (+9.2)**	**71.7 (+3.5)**	**16.6 (-10.5)**	**14.9 (-4.9)**
CiteSeer	GCN	BASE	37.6	42.5	42.6	39.2	39.3	45.1	56.0	43.2	56.0	36.1	54.5	27.0	16.9
		+BAT_0	52.7	57.9	57.5	57.9	60.1	57.7	60.6	**57.8 (+14.6)**	**60.6 (+4.6)**	**56.9 (+20.8)**	**59.9 (+5.4)**	**17.6 (-9.4)**	**13.8 (-3.1)**
		+BAT_1	55.4	58.4	59.3	58.8	62.0	57.6	62.7	**59.2 (+16.0)**	**62.7 (+6.7)**	**58.4 (+22.3)**	**62.5 (+8.0)**	**19.3 (-7.7)**	**13.9 (-3.0)**
	GAT	BASE	39.2	41.3	43.2	36.0	37.0	41.8	51.5	41.4	51.5	34.1	48.3	29.0	25.2
		+BAT_0	55.7	59.3	58.3	60.1	60.6	56.1	60.9	**58.7 (+17.3)**	**60.9 (+9.4)**	**58.1 (+23.9)**	**60.0 (+11.7)**	**16.3 (-12.6)**	**10.7 (-14.5)**
		+BAT_1	60.3	61.2	59.1	60.3	62.4	57.7	63.5	**60.6 (+19.2)**	**63.5 (+12.0)**	**59.9 (+25.8)**	**62.5 (+14.2)**	**17.7 (-11.3)**	**13.2 (-12.0)**
	SAGE	BASE	43.0	45.9	48.6	39.4	38.4	42.2	52.6	44.3	52.6	37.9	51.0	27.1	19.8
		+BAT_0	55.0	58.0	56.3	61.4	64.1	60.9	64.4	**60.0 (+15.7)**	**64.4 (+11.8)**	**59.4 (+21.6)**	**63.9 (+12.8)**	**17.5 (-9.7)**	**13.2 (-6.6)**
		+BAT_1	53.2	55.9	56.5	61.9	66.3	62.3	63.8	**60.0 (+15.7)**	**66.3 (+13.8)**	**59.3 (+21.4)**	**65.9 (+14.9)**	**18.6 (-8.6)**	**12.8 (-7.0)**
PubMed	GCN	BASE	64.2	71.2	71.5	65.0	64.4	74.0	73.7	69.1	74.0	63.5	71.3	23.2	11.9
		+BAT_0	68.6	74.2	73.2	72.5	73.2	73.1	76.1	**73.0 (+3.8)**	**76.1 (+2.1)**	**72.0 (+8.5)**	**75.8 (+4.5)**	**9.1 (-14.1)**	**3.4 (-8.6)**
		+BAT_1	67.6	73.4	72.5	72.9	73.1	76.6	76.9	**73.3 (+4.1)**	**76.9 (+2.9)**	**72.6 (+9.1)**	**76.9 (+5.6)**	**9.9 (-13.4)**	**5.1 (-6.8)**
	GAT	BASE	65.5	68.4	71.2	65.1	64.8	68.7	73.1	68.1	73.1	62.4	71.8	24.8	10.3
		+BAT_0	73.2	75.3	75.6	73.3	73.9	74.7	74.3	**74.3 (+6.2)**	**75.6 (+2.4)**	**73.4 (+11.1)**	**75.1 (+3.3)**	**6.5 (-18.2)**	**3.0 (-7.3)**
		+BAT_1	74.8	74.5	75.2	73.9	74.1	74.4	75.7	**74.6 (+6.5)**	**75.7 (+2.5)**	**73.9 (+11.5)**	**75.0 (+3.2)**	**6.9 (-17.8)**	**4.6 (-5.7)**
	SAGE	BASE	67.6	68.0	69.1	69.2	65.0	71.5	71.4	68.8	71.5	64.5	70.1	22.1	11.8
		+BAT_0	75.3	74.6	74.2	74.9	74.6	74.7	75.9	**74.9 (+6.1)**	**75.9 (+4.3)**	**74.2 (+9.7)**	**75.3 (+5.3)**	**7.6 (-14.4)**	**3.4 (-8.4)**
		+BAT_1	77.4	75.4	75.3	75.8	77.3	76.1	76.5	**76.3 (+7.4)**	**77.4 (+5.8)**	**75.8 (+11.3)**	**76.9 (+6.9)**	**6.8 (-15.3)**	**4.1 (-7.7)**

*BAT_0/BAT_1: BAT with 0^{th}/1^{st}-order posterior likelihood estimation. ERM: Empirical Risk Minimization (standard training), RW: Reweight (Japkowicz & Stephen, 2002), RN: ReNode (Chen et al., 2021), RS: Resampling (Japkowicz & Stephen, 2002), SM: SMOTE (Chawla et al., 2002), GS: GraphSMOTE (Zhao et al., 2021b), GE: GraphENS (Park et al., 2022).

available for CIGL (Hong et al., 2021; Kang et al., 2019; Shi et al., 2020), previous studies (Park et al., 2022; Song et al., 2022a) have shown they are generally outperformed by the baselines we use. Detailed settings can be found in Appendix B.2. To ensure a comprehensive evaluation, we employ **three** metrics to assess both the classification performance (Balanced Accuracy, Macro-F1) and the model predictive bias (PerfStd, i.e., the standard deviation of accuracy scores across all classes). Lower PerfStd indicates smaller performance gap between all majority and minority classes, and thus smaller predictive bias. For clarity, we use ↑/↓ to denote larger/smaller is better for each metric.

BAT significantly boosts various CIGL techniques. We report the main results in Table 1 (IR=10). In all settings (3 datasets×3 backbones×7 baselines×3 metrics), BAT achieves significant and consistent performance improvements over other CIGL techniques, which also yields new state-of-the-art performance. Specifically: **(1)** By mitigating AMP and DMP, BAT further boosts the best CIGL baseline by a large margin, e.g., it boosts the best balanced accuracy score by 4.1/13.8/5.8 on Cora/CiteSeer/PubMed datasets. **(2)** In addition to better classification performance, BAT also greatly reduces the predictive bias in CIGL, with up to 10.7/14.5/18.2 average performance deviation reduction on Cora/CiteSeer/PubMed. **(3)** Compared with BAT_0, BAT_1 achieves better classification performance with first-order posterior likelihood estimation. But we also note that BAT_0 performs better in terms of reducing predictive bias and is more computationally efficient, as we will discuss later in the scalability experiments (Table 3).

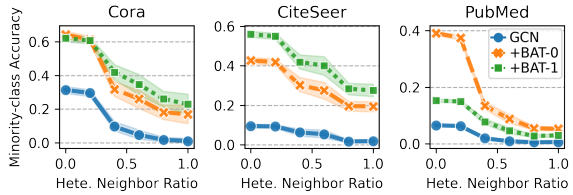
BAT is robust even under extreme class imbalance. We further extend Table 1 and test BAT's robustness to varying types and levels of imbalance, as reported in Table 2. In this experiment, we extend the step imbalance ratio from 10 (used in Table 1) to 20 to test BAT under even more challenging class imbalance scenarios. In addition, we consider the natural (long-tail) class imbalance (Park et al., 2022) that is commonly observed in real-world graphs with IR of 50 and 100. Datasets from (Shchur et al., 2018) (*CS, Physics*) are also included to test BAT on large-scale tasks. Results show that: **(1)** BAT is robust to extreme class imbalance, and it consistently boosts the CIGL performance by a significant margin under varying types and levels of imbalance. **(2)** The performance drop from increasing IR is significantly lowered by BAT, i.e., applying BAT improves model's robustness to extreme class imbalance. **(3)** BAT*'s advantage is even more prominent under higher class imbalance*, e.g., on Cora with step IR, the performance gain of applying BAT on Base raised from 8.2 to 18.5 when IR increased from 10 to 20, and similar patterns can be observed in other settings.

Table 2. BAT deliver consistent and significant performance gain to CIGL methods under varying types and levels of class imbalance. The numbers in brackets are the performance gain brought about by BAT over Base/BestCIGL method. We report the Balanced Accuracy here, full results with other metrics can be found in Appendix D.3.

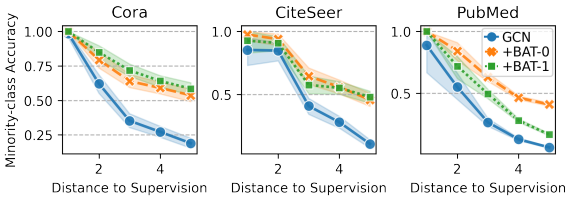
Dataset	**Cora**		**CiteSeer**		**PubMed**		**CS**		**Physics**	
Step IR	10	20	10	20	10	20	10	20	10	20
Base	61.6	52.7	37.6	34.2	64.2	60.8	75.4	65.3	80.1	67.7
+ BAT	**69.8** (+8.2)	**71.3** (+18.5)	**55.4** (+17.7)	**51.3** (+17.1)	**68.6** (+4.4)	**63.3** (+2.5)	**82.6** (+7.2)	**79.9** (+14.5)	**87.6** (+7.5)	**88.0** (+20.2)
BestCIGL	70.1	66.5	56.0	47.2	74.0	71.1	84.1	81.3	89.4	85.7
+ BAT	**74.2** (+4.1)	**71.6** (+5.1)	**62.7** (+6.7)	**62.5** (+15.3)	**76.9** (+2.9)	**75.7** (+4.6)	**86.3** (+2.2)	**85.6** (+4.3)	**91.2** (+1.9)	**90.9** (+5.2)
Natural IR	50	100	50	100	50	100	50	100	50	100
Base	60.1	47.0	28.1	21.9	55.1	46.4	72.7	59.2	80.7	64.7
+ BAT	**68.7** (+8.6)	**69.6** (+22.6)	**54.9** (+26.9)	**48.9** (+27.0)	**67.2** (+12.1)	**60.7** (+14.3)	**78.6** (+5.9)	**74.7** (+15.5)	**88.8** (+8.1)	**87.8** (+23.2)
BestCIGL	70.0	66.2	54.5	45.0	71.3	68.9	83.9	80.9	89.5	86.2
+ BAT	**72.8** (+2.9)	**70.2** (+4.0)	**62.5** (+8.0)	**62.1** (+17.1)	**76.9** (+5.6)	**74.9** (+6.0)	**85.4** (+1.6)	**84.6** (+3.7)	**90.7** (+1.2)	**90.0** (+3.8)

*__Base__: vanilla GCN model; __Base+BAT__: applying BAT without any other CIGL method; __BestCIGL__: best CIGL baseline w/o BAT; __BestCIGL+BAT__: best CIGL baseline w/ BAT;

BAT effectively alleviates both AMP and DMP. We further design experiments to verify to what extent BAT can effectively handle the topological challenges identified in this paper, i.e., ambivalent and distant message-passing. Specifically, we investigate whether BAT can improve the prediction accuracy of minority class nodes that are highly influenced by AMP/DMP, i.e., with high heterophilic neighbor ratio/long distance to supervision. Results are shown in Fig. 6 (5 independent runs with GCN classifier, IR=10). As can be observed, BAT effectively alleviates the negative impact of AMP and DMP and helps node classifiers to achieve better performance in minority classes.



(a) BAT mitigates AMP in multiple CIGL tasks.



(b) BAT mitigates DMP in multiple CIGL tasks.

Figure 6. BAT effectively alleviates both AMP and DMP.

BAT is computationally efficient. As previously discussed in the complexity analysis, BAT with 0th/1st-order estimation **scales linearly** with the number of nodes/edges (i.e., with $\mathcal{O}(|\mathcal{V}|C)$ or $\mathcal{O}(|\mathcal{E}|C)$ complexity). Since all the operations can be executed in parallel in matrix form, BAT_0/BAT_1 has $\mathcal{O}(\frac{|\mathcal{V}|C}{D})$/$\mathcal{O}(\frac{|\mathcal{E}|C}{D})$ time complexity, where D is the number of available computational units and is usually large for modern GPUs. Table 3 reports the ratio of virtual nodes/edges to the original graph introduced and the running time of BAT. It can be observed that BAT only introduces a small number of virtual nodes/edges, and is highly efficient (taking milliseconds for augmentation) in practice.

Table 3. Efficiency results of BAT_0/BAT_1.

Dataset	**Δ Nodes (%)**	**Δ Edges (%)**	**Δ Time (ms)**
Cora	0.258%	2.842%/1.509%	4.50/4.65ms
CiteSeer	0.180%	3.715%/1.081%	4.72/4.97ms
PubMed	0.015%	3.175%/1.464%	6.23/6.64ms
CS	0.082%	1.395%/1.053%	16.97/18.61ms
Physics	0.014%	0.797%/0.527%	30.68/31.91ms

* Results obtained on an NVIDIA® Tesla V100 32GB GPU.

Further discussions. We refer the readers to Appendix for reproducibility details (§B), ablation study and extended discussions (§C), and additional empirical results (§D).

5. Related Works

Imbalanced graph learning. Class imbalance is ubiquitous in many machine-learning tasks (Krawczyk, 2016; Liu et al., 2020a;b). However, most of the existing works focus on i.i.d. scenarios (LemaÃŽtre et al., 2017; Liu et al., 2023b), which may not be tailored to the unique characteristics of graph data. To handle imbalanced graph learning, several techniques have been proposed in recent studies (e.g., by adversarial training (Qu et al., 2021), designing new GNN architectures (Wang et al., 2020; Liu et al., 2021) or loss functions (Song et al., 2022a)), we review the most closely related model-agnostic CR methods here. One of the early works GraphSMOTE (Zhao et al., 2021b) adopts SMOTE (Chawla et al., 2002) oversampling in the node embedding space to synthesize minority nodes and complements the topology with a learnable edge predictor. A more recent work GraphENS (Park et al., 2022) synthesizes the ego network through saliency-based ego network mixing to handle the neighbor-overfitting problem. Most studies are rooted in a *class-rebalancing* perspective and address the imbalance by node/class-wise reweighting or resampling.

Topology-imbalance in graphs. Topology imbalance is firstly discussed in Chen et al. (2021). They found that "the unequal structure role of labeled nodes" can cause influence conflict, and propose to re-weight the labeled nodes based on a conflict detection measure. Other works further discussed how to better address the issue via position-aware structure learning (Sun et al., 2022), and handle topology-imbalance in fake news detection (Gao et al., 2022) and bankruptcy prediction (Liu et al., 2023a). These studies discussed concepts related to "influence conflict/insufficiency" (Chen et al., 2021), which motivated us to investigate AMP/DMP in this work. It is worth noting that AMP/DMP coefficients are defined on all nodes based on the k-hop local neighborhood, while the influence conflict measure in Chen et al. (2021) is defined only on labeled nodes by global personalized PageRank score, and influence insufficient has no formal definition. Therefore, we did not inherit their naming and employ AMP/DMP to distinguish and avoid confusion with the existing concepts. We theoretically investigate the role of AMP/DMP in shaping class-imbalance bias, and propose an augmentation technique BAT tailored for handling AMP and DMP in CIGL. Results show that the topology imbalance algorithm can also significantly boosted by BAT.

Heterophilic graph/long-distance propagation. Numerous studies exist in the literature concerning learning from heterophilic graphs (Yan et al., 2024; Xu et al., 2023; Guo et al., 2023) and employing multi-hop propagation (Zhao et al., 2021a; Fu et al., 2024). In particular, heterophilic GNNs often combine intermediate representations to derive more refined structure-aware features (Zhu et al., 2020). The GPRGNN (Chien et al., 2020) takes a step further by introducing learnable weights to adaptively combine representations from each layer. Meanwhile, in multi-hop propagation, APPNP (Gasteiger et al., 2018) stands as a representative technique that leverages personalized PageRank for extracting information from a broader neighborhood. Nevertheless, these works focus on addressing *global* graph heterophily and long-distance propagation by modifying the GNN architecture or aggregation operators. They are not tailored to address class imbalance and cannot readily handle the AMP and DMP. We refer the readers to Appendix D where we show that BAT can also significantly boost the performance of such GNNs (Chien et al., 2020; Gasteiger et al., 2018) in various CIGL tasks.

6. Conclusion

In this paper, we study class-imbalanced graph learning from a novel topological perspective. We theoretically reveal that two fundamental topological phenomena, i.e., ambivalent and distant message-passing, can greatly exacerbate the predictive bias stemming from class imbalance. Our findings reveal an unexplored avenue that limits the performance of existing class-rebalancing-based CIGL techniques. In light of this, we propose BAT to handle the topological challenges in CIGL by dynamic topological augmentation. BAT is a swift and model-agnostic framework that can seamlessly complement other CIGL techniques, augmenting their performance and mitigating predictive bias. Systematic experiments validate BAT's superior effectiveness, versatility, robustness, and efficiency across various CIGL tasks.

Impact Statement

This paper presents work whose goal is to advance the field of Graph Data Mining. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

This work is supported by NSF (1947135), the NSF Program on Fairness in AI in collaboration with Amazon (1939725), NIFA (2020-67021-32799), DHS (17STQAC00001-07-00), the C3.ai Digital Transformation Institute, MIT-IBM Watson AI Lab, and IBM-Illinois Discovery Accelerator Institute. The content of the information in this document does not necessarily reflect the position or the policy of the Government or Amazon, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Akoglu, L., Tong, H., and Koutra, D. Graph based anomaly detection and description: a survey. *Data mining and knowledge discovery*, 29(3):626–688, 2015.

Antoniadis, A. and Fan, J. Regularization of wavelet approximations. *Journal of the American Statistical Association*, 96(455):939–967, 2001.

Bojchevski, A. and Günnemann, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*, 2017.

Cai, L., Li, J., Wang, J., and Ji, S. Line graph neural networks for link prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5103–5113, 2021.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Chen, D., Lin, Y., Zhao, G., Ren, X., Li, P., Zhou, J., and Sun, X. Topology-imbalance learning for semi-

supervised node classification. *Advances in Neural Information Processing Systems*, 34:29885–29897, 2021.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

Decelle, A., Krzakala, F., Moore, C., and Zdeborová, L. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

Fey, M. and Lenssen, J. E. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Fu, D. and He, J. SDG: A simplified and dynamic graph neural network. In Diaz, F., Shah, C., Suel, T., Castells, P., Jones, R., and Sakai, T. (eds.), *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pp. 2273–2277. ACM, 2021. doi: 10.1145/3404835.3463059. URL https://doi.org/10.1145/3404835.3463059.

Fu, D., Zhou, D., Maciejewski, R., Croitoru, A., Boyd, M., and He, J. Fairness-aware clique-preserving spectral clustering of temporal graphs. In Ding, Y., Tang, J., Sequeda, J. F., Aroyo, L., Castillo, C., and Houben, G. (eds.), *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023*, pp. 3755–3765. ACM, 2023. doi: 10.1145/3543507.3583423. URL https://doi.org/10.1145/3543507.3583423.

Fu, D., Hua, Z., Xie, Y., Fang, J., Zhang, S., Sancak, K., Wu, H., Malevich, A., He, J., and Long, B. Vcr-graphormer: A mini-batch graph transformer via virtual connections. *CoRR*, abs/2403.16030, 2024. doi: 10.48550/ARXIV.2403.16030. URL https://doi.org/10.48550/arXiv.2403.16030.

Gao, L., Song, L., Liu, J., Chen, B., and Shang, X. Topology imbalance and relation inauthenticity aware hierarchical graph attention networks for fake news detection. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 4687–4696, 2022.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Guo, K., Cao, X., Liu, Z., and Chang, Y. Taming over-smoothing representation on heterophilic graphs. *Information Sciences*, 647:119463, 2023.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., and Qian, X. Bayesian graph neural networks with adaptive connection sampling. In *International conference on machine learning*, pp. 4094–4104. PMLR, 2020.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Hong, Y., Han, S., Choi, K., Seo, S., Kim, B., and Chang, B. 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.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Huang, Z., Tang, Y., and Chen, Y. A graph neural network-based node classification model on class-imbalanced graph data. *Knowledge-Based Systems*, 244:108538, 2022.

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

Johnson, J. M. and Khoshgoftaar, T. M. Survey on deep learning with class imbalance. *Journal of Big Data*, 6(1):1–54, 2019.

Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.

Kang, J., He, J., Maciejewski, R., and Tong, H. Inform: Individual fairness on graph mining. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 379–389, 2020.

Kang, J., Zhou, Q., and Tong, H. Jurygcn: quantifying jackknife uncertainty on graph convolutional networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 742–752, 2022a.

Kang, J., Zhu, Y., Xia, Y., Luo, J., and Tong, H. Rawlsgcn: Towards rawlsian difference principle on graph convolutional network. In *Proceedings of the ACM Web Conference 2022*, pp. 1214–1225, 2022b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krawczyk, B. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

LemaÃŽtre, G., Nogueira, F., and Aridas, C. K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of machine learning research*, 18(17):1–5, 2017.

Li, X., Wen, L., Deng, Y., Feng, F., Hu, X., Wang, L., and Fan, Z. Graph neural network with curriculum learning for imbalanced node classification. *arXiv preprint arXiv:2202.02529*, 2022.

Liu, Y., Ao, X., Qin, Z., Chi, J., Feng, J., Yang, H., and He, Q. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pp. 3168–3177, 2021.

Liu, Y., Gao, Z., Liu, X., Luo, P., Yang, Y., and Xiong, H. Qtiah-gnn: Quantity and topology imbalance-aware heterogeneous graph neural network for bankruptcy prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1572–1582, 2023a.

Liu, Z., Cao, W., Gao, Z., Bian, J., Chen, H., Chang, Y., and Liu, T.-Y. Self-paced ensemble for highly imbalanced massive data classification. In *2020 IEEE 36th international conference on data engineering (ICDE)*, pp. 841–852. IEEE, 2020a.

Liu, Z., Wei, P., Jiang, J., Cao, W., Bian, J., and Chang, Y. Mesa: boost ensemble imbalanced learning with meta-sampler. *Advances in Neural Information Processing Systems*, 33:14463–14474, 2020b.

Liu, Z., Kang, J., Tong, H., and Chang, Y. Imbens: Ensemble class-imbalanced learning in python. *arXiv preprint arXiv:2111.12776*, 2023b.

Liu, Z.-Y., Li, S.-Y., Chen, S., Hu, Y., and Huang, S.-J. Uncertainty aware graph gaussian process for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4957–4964, 2020c.

Ma, Y., Liu, X., Zhao, T., Liu, Y., Tang, J., and Shah, N. A unified view on graph neural networks as graph signal denoising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1202–1211, 2021.

Nt, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Pandey, B., Bhanodia, P. K., Khamparia, A., and Pandey, D. K. A comprehensive survey of edge prediction in social networks: Techniques, parameters and challenges. *Expert Systems with Applications*, 124:164–181, 2019.

Park, J., Song, J., and Yang, E. Graphens: Neighbor-aware ego network synthesis for class-imbalanced node classification. In *International Conference on Learning Representations*, 2022.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Qu, L., Zhu, H., Zheng, R., Shi, Y., and Yin, H. Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1390–1398, 2021.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Shi, M., Tang, Y., Zhu, X., Wilson, D., and Liu, J. Multi-class imbalanced graph convolutional network learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 2020.

Song, J., Park, J., and Yang, E. Tam: Topology-aware margin loss for class-imbalanced node classification. In *International Conference on Machine Learning*, pp. 20369–20383. PMLR, 2022a.

Song, Z., Yang, X., Xu, Z., and King, I. Graph-based semi-supervised learning: A comprehensive review. *IEEE Transactions on Neural Networks and Learning Systems*, 2022b.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Stadler, M., Charpentier, B., Geisler, S., Zügner, D., and Günnemann, S. Graph posterior network: Bayesian predictive uncertainty for node classification. *Advances in Neural Information Processing Systems*, 34:18033–18048, 2021.

Sun, Q., Li, J., Yuan, H., Fu, X., Peng, H., Ji, C., Li, Q., and Yu, P. S. Position-aware structure learning for graph topology-imbalance by relieving under-reaching and over-squashing. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 1848–1857, 2022.

Tang, L. and Liu, H. Community detection and mining in social media. *Synthesis lectures on data mining and knowledge discovery*, 2(1):1–137, 2010.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wang, Z., Ye, X., Wang, C., Cui, J., and Philip, S. Y. Network embedding with completely-imbalanced labels. *IEEE Transactions on Knowledge and Data Engineering*, 33(11):3634–3647, 2020.

Welling, M. and Kipf, T. N. Semi-supervised classification with graph convolutional networks. In *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

Wu, L., Xia, J., Gao, Z., Lin, H., Tan, C., and Li, S. Z. Graphmixup: Improving class-imbalanced node classification by reinforcement mixup and self-supervised context prediction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 519–535. Springer, 2022.

Xu, Z., Chen, Y., Zhou, Q., Wu, Y., Pan, M., Yang, H., and Tong, H. Node classification beyond homophily: Towards a general solution. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 2862–2873, 2023.

Yan, Y., Liu, L., Ban, Y., Jing, B., and Tong, H. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 4564–4572, 2021a.

Yan, Y., Zhang, S., and Tong, H. Bright: A bridging algorithm for network alignment. In *Proceedings of the web conference 2021*, pp. 3907–3917, 2021b.

Yan, Y., Chen, Y., Chen, H., Xu, M., Das, M., Yang, H., and Tong, H. From trainable negative depth to edge heterophily in graphs. *Advances in Neural Information Processing Systems*, 36, 2024.

Yun, S., Kim, K., Yoon, K., and Park, C. Lte4g: Long-tail experts for graph neural networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 2434–2443, 2022.

Zeng, Z., Zhang, S., Xia, Y., and Tong, H. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM Web Conference 2023*, pp. 372–382, 2023a.

Zeng, Z., Zhu, R., Xia, Y., Zeng, H., and Tong, H. Generative graph dictionary learning. In *International Conference on Machine Learning*, pp. 40749–40769. PMLR, 2023b.

Zeng, Z., Du, B., Zhang, S., Xia, Y., Liu, Z., and Tong, H. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 16660–16668, 2024.

Zhang, Y., Pal, S., Coates, M., and Ustebay, D. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 5829–5836, 2019.

Zhao, J., Dong, Y., Ding, M., Kharlamov, E., and Tang, J. Adaptive diffusion in graph neural networks. *Advances in Neural Information Processing Systems*, 34:23321–23333, 2021a.

Zhao, T., Zhang, X., and Wang, S. 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, 2021b.

Zhao, T., Luo, D., Zhang, X., and Wang, S. Topoimb: Toward topology-level imbalance in learning from graphs. In *Learning on Graphs Conference*, pp. 37–1. PMLR, 2022.

Zhao, X., Chen, F., Hu, S., and Cho, J.-H. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836, 2020.

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., and Koutra, D. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

Appendix

A. Proofs of Theoretical Results

Define random variables $H_{ij}^k := |\mathcal{V}_j \cap \mathcal{H}(u,k)|$ denote the number of class-j k-hop homo-connected neighbors of a node $u \in \mathcal{V}_i$. Note that the results of both Theorems 2.1 & 2.2 depend only on the distributions of H_{ij}^k. Thus, we will first derive the limiting distributions of H_{ij}^k as a technical lemma, and then give the proofs of Theorems 2.1 & 2.2.

A.1. Limiting Distributions of H_{ij}^k

To count the number of homo-connected neighbors, consider the breadth-first search (BFS) tree rooted at node $u \in \mathcal{V}_1$. By enumerating the numbers of $1,\dots,k$-hop homo-connected neighbors in the BFS tree respectively, we can calculate the exact joint distribution of (H_{11}^k, H_{12}^k):

$$\begin{aligned}\mathbb{P}\{H_{11}^k = s, H_{12}^k = s'\} = \sum_{\substack{a_1+\cdots+a_k=s\\ b_1+\cdots+b_k=s'}} &\binom{n_1-1}{a_1,\dots,a_k,n_1-1-s}\binom{n_2}{b_1,\dots,b_k,n_2-s'}\\ &p^{a_1}\left(\prod_{t=2}^{k}(1-p)^{a_t(1+a_1+\cdots+a_{t-2})}(1-(1-p)^{a_{t-1}})^{a_t}\right)(1-p)^{(n_1-1-s)(1+s-a_k)}\\ &q^{b_1}\left(\prod_{t=2}^{k}(1-q)^{b_t}(1-p)^{b_t(b_1+\cdots+b_{t-2})}(1-(1-p)^{b_{t-1}})^{b_t}\right)(1-q)^{n_2-s'}(1-p)^{(n_2-s')(s'-b_k)}.\end{aligned}$$

Thus, H_{11}^k and H_{12}^k are independent, and their marginal distributions are:

$$\begin{aligned}\mathbb{P}\{H_{11}^k = s\} = \sum_{a_1+\cdots+a_k=s} &\binom{n_1-1}{a_1,\dots,a_k,n_1-1-s}\\ &p^{a_1}\left(\prod_{t=2}^{k}(1-p)^{a_t(1+a_1+\cdots+a_{t-2})}(1-(1-p)^{a_{t-1}})^{a_t}\right)(1-p)^{(n_1-1-s)(1+s-a_k)},\\ \mathbb{P}\{H_{12}^k = s\} = \sum_{b_1+\cdots+b_k=s} &\binom{n_2}{b_1,\dots,b_k,n_2-s}\\ &q^{b_1}\left(\prod_{t=2}^{k}(1-q)^{b_t}(1-p)^{b_t(b_1+\cdots+b_{t-2})}(1-(1-p)^{b_{t-1}})^{b_t}\right)(1-q)^{n_2-s}(1-p)^{(n_2-s)(s-b_k)}.\end{aligned}$$

Now consider $n \to \infty$. Let

$$\begin{aligned}
\beta_{11} &:= \lim_{n\to\infty} n_1 \cdot p = \beta, \\
\beta_{22} &:= \lim_{n\to\infty} n_2 \cdot p = \rho\beta, \\
\beta_{21} &:= \lim_{n\to\infty} n_1 \cdot q = \beta\frac{q}{p}, \\
\beta_{12} &:= \lim_{n\to\infty} n_2 \cdot q = \rho\beta\frac{q}{p}.
\end{aligned}$$

Then, the limiting distributions of H_{11}^k and H_{12}^k are:

$$\begin{aligned}
\mathbb{P}\{H_{11}^k = s\} &\sim \sum_{a_1+\cdots+a_k=s} \frac{n_1^s}{a_1!\cdots a_k!} p^{a_1} \left(\prod_{t=2}^{k} (a_{t-1}p)^{a_t}\right) (1-p)^{n_1(1+s-a_k)} \\
&\to \mathrm{e}^{-\beta_{11}} \sum_{a_1+\cdots+a_k=s} \frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_1}}{a_1!} \left(\prod_{t=2}^{k-1} \frac{(a_{t-1}\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_t}}{a_t!}\right) \frac{(a_{k-1}\beta_{11})^{a_k}}{a_k!}, \\
\mathbb{P}\{H_{12}^k = s\} &\sim \sum_{b_1+\cdots+b_k=s} \frac{n_2^s}{b_1!\cdots b_k!} q^{b_1} \left(\prod_{t=2}^{k} (b_{t-1}p)^{b_t}\right) (1-q)^{n_2} (1-p)^{n_2(s-b_k)} \\
&\to \mathrm{e}^{-\beta_{12}} \sum_{b_1+\cdots+b_k=s} \frac{(\beta_{12}\mathrm{e}^{-\beta_{22}})^{b_1}}{b_1!} \left(\prod_{t=2}^{k-1} \frac{(b_{t-1}\beta_{22}\mathrm{e}^{-\beta_{22}})^{b_t}}{b_t!}\right) \frac{(b_{k-1}\beta_{22})^{b_k}}{b_k!}.
\end{aligned}$$

Figure 7 shows that the limiting distributions are good approximation for finite n.

A.2. Proof of Theorem 2.1

Note that for any $t' = 1, \ldots, k$,

$$\mathrm{e}^{-\beta_{11}} \sum_{a_1=0}^{\infty} \cdots \sum_{a_k=0}^{\infty} a_{t'} \cdot \frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_1}}{a_1!} \left(\prod_{t=2}^{k-1} \frac{(a_{t-1}\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_t}}{a_t!}\right) \frac{(a_{k-1}\beta_{11})^{a_k}}{a_k!} = \beta_{11}^{t'}.$$

Thus,

$$\begin{aligned}
&\lim_{n\to\infty} \mathbb{E}[H_{11}^k] = \sum_{s=0}^{\infty} s \cdot \lim_{n\to\infty} \mathbb{P}\{H_{11}^k = s\} \\
&= \sum_{s=0}^{\infty} s \cdot \mathrm{e}^{-\beta_{11}} \sum_{a_1+\cdots+a_k=s} \frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_1}}{a_1!} \left(\prod_{t=2}^{k-1} \frac{(a_{t-1}\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_t}}{a_t!}\right) \frac{(a_{k-1}\beta_{11})^{a_k}}{a_k!} \\
&= \sum_{s=0}^{\infty} \mathrm{e}^{-\beta_{11}} \sum_{a_1+\cdots+a_k=s} s \cdot \frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_1}}{a_1!} \left(\prod_{t=2}^{k-1} \frac{(a_{t-1}\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_t}}{a_t!}\right) \frac{(a_{k-1}\beta_{11})^{a_k}}{a_k!} \\
&= \mathrm{e}^{-\beta_{11}} \sum_{a_1=0}^{\infty} \cdots \sum_{a_k=0}^{\infty} (a_1 + \cdots + a_k) \cdot \frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_1}}{a_1!} \left(\prod_{t=2}^{k-1} \frac{(a_{t-1}\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_t}}{a_t!}\right) \frac{(a_{k-1}\beta_{11})^{a_k}}{a_k!} \\
&= \sum_{t'=1}^{k} \mathrm{e}^{-\beta_{11}} \sum_{a_1=0}^{\infty} \cdots \sum_{a_k=0}^{\infty} a_{t'} \cdot \frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_1}}{a_1!} \left(\prod_{t=2}^{k-1} \frac{(a_{t-1}\beta_{11}\mathrm{e}^{-\beta_{11}})^{a_t}}{a_t!}\right) \frac{(a_{k-1}\beta_{11})^{a_k}}{a_k!} \\
&= \sum_{t'=1}^{k} \beta_{11}^{t'}.
\end{aligned}$$

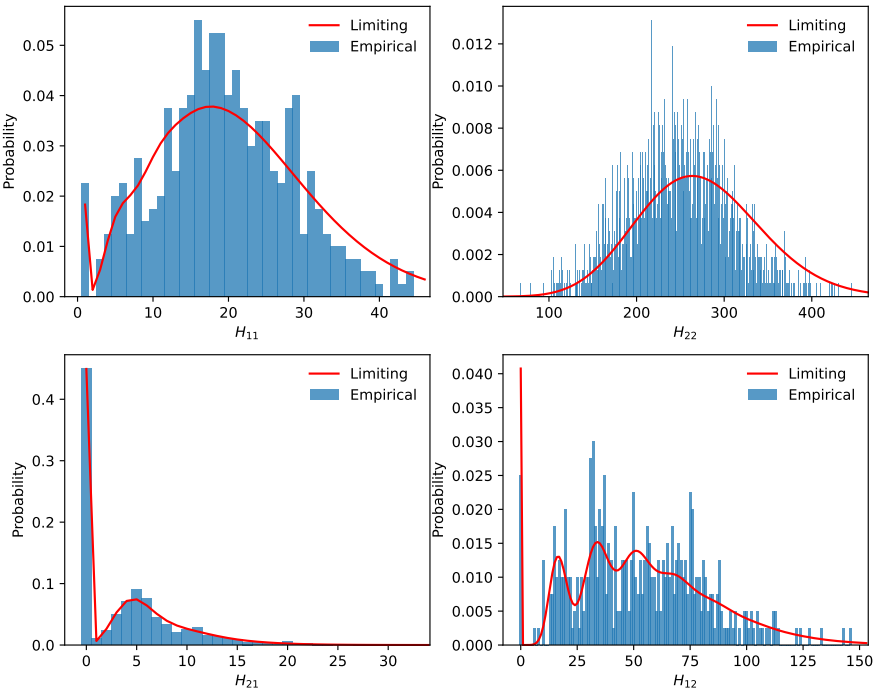


Figure 7. Distributions of H_{ij}^k. Simulated with $n = 2000$, $\rho = 4$, $p = 0.01$, $q = 0.002$, $k = 2$.

Similarly,

$$\lim_{n\to\infty} \mathbb{E}[H_{22}^k] = \sum_{t=1}^{k} \beta_{22}^t,$$
$$\lim_{n\to\infty} \mathbb{E}[H_{12}^k] = \sum_{t=1}^{k} \beta_{12}\beta_{22}^{t-1},$$
$$\lim_{n\to\infty} \mathbb{E}[H_{21}^k] = \sum_{t=1}^{k} \beta_{21}\beta_{11}^{t-1}.$$

It follows that

$$\begin{aligned}
\lim_{n\to\infty} \frac{\alpha_1^k}{\alpha_2^k} &= \lim_{n\to\infty} \frac{\mathbb{E}[H_{12}^k]/\mathbb{E}[H_{11}^k]}{\mathbb{E}[H_{21}^k]/\mathbb{E}[H_{22}^k]} \\
&= \frac{\sum_{t=1}^k \beta_{12}\beta_{22}^{t-1} \big/ \sum_{t=1}^k \beta_{11}^t}{\sum_{t=1}^k \beta_{21}\beta_{11}^{t-1} \big/ \sum_{t=1}^k \beta_{22}^t} \\
&= \frac{\beta_{12}\beta_{22}}{\beta_{21}\beta_{11}} \cdot \frac{\left(\sum_{t=1}^k \beta_{22}^{t-1}\right)^2}{\left(\sum_{t=1}^k \beta_{11}^{t-1}\right)^2} \\
&= \left(\rho \cdot \frac{\sum_{t=1}^k (\rho\beta)^{t-1}}{\sum_{t=1}^k \beta^{t-1}}\right)^2.
\end{aligned}$$

□

A.3. Proof of Theorem 2.2

For $k = 2$, note the identity:

$$\sum_{s=0}^{\infty}\sum_{a=0}^{s}\frac{\lambda^a(\mu a)^{s-a}}{a!(s-a)!} = \sum_{a=0}^{\infty}\frac{\lambda^a}{a!}\sum_{b=0}^{\infty}\frac{(\mu a)^b}{b!} = \sum_{a=0}^{\infty}\frac{\lambda^a \mathrm{e}^{\mu a}}{a!} = \mathrm{e}^{\lambda \mathrm{e}^{\mu}}.$$

It follows that (with $\lambda = (1-r_1^{\mathrm{L}})\beta_{11}\mathrm{e}^{-\beta_{11}}$ and $\mu = (1-r_1^{\mathrm{L}})\beta_{11}$)

$$\begin{aligned}
\lim_{n\to\infty}\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{11}^2}] &= \sum_{s=0}^{\infty}(1-r_1^{\mathrm{L}})^s \cdot \lim_{n\to\infty}\mathbb{P}\{H_{11}^2 = s\} \\
&= \sum_{s=0}^{\infty}(1-r_1^{\mathrm{L}})^s \cdot \mathrm{e}^{-\beta_{11}}\sum_{a=0}^{s}\frac{(\beta_{11}\mathrm{e}^{-\beta_{11}})^a(\beta_{11}a)^{s-a}}{a!(s-a)!} \\
&= \mathrm{e}^{-\beta_{11}}\sum_{s=0}^{\infty}\sum_{a=0}^{s}\frac{((1-r_1^{\mathrm{L}})\beta_{11}\mathrm{e}^{-\beta_{11}})^a((1-r_1^{\mathrm{L}})\beta_{11}a)^{s-a}}{a!(s-a)!} \\
&= \mathrm{e}^{-\beta_{11}}\mathrm{e}^{(1-r_1^{\mathrm{L}})\beta_{11}\mathrm{e}^{-\beta_{11}}\mathrm{e}^{(1-r_1^{\mathrm{L}})\beta_{11}}} \\
&= \mathrm{e}^{-(1-(1-r_1^{\mathrm{L}})\mathrm{e}^{-r_1^{\mathrm{L}}\beta_{11}})\beta_{11}} \\
&\approx \mathrm{e}^{-\beta_{11}}.
\end{aligned}$$

For $k = 3$, similarly,

$$\lim_{n\to\infty}\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{11}^3}] = \mathrm{e}^{-\left(1-(1-r_1^{\mathrm{L}})\beta_{11}\mathrm{e}^{-\left(1-(1-r_1^{\mathrm{L}})\beta_{11}\mathrm{e}^{-r_1^{\mathrm{L}}\beta_{11}}\right)\beta_{11}}\right)\beta_{11}} \approx \mathrm{e}^{-\beta_{11}}.$$

In general, the result for k has k nested exponentiations, but we still have:

$$\lim_{n\to\infty}\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{11}^k}] \approx \mathrm{e}^{-\beta_{11}}.$$

Similarly,

$$\begin{aligned}
\lim_{n\to\infty}\mathbb{E}[(1-r_2^{\mathrm{L}})^{H_{12}^k}] &\approx \mathrm{e}^{-\beta_{12}}, \\
\lim_{n\to\infty}\mathbb{E}[(1-r_2^{\mathrm{L}})^{H_{22}^k}] &\approx \mathrm{e}^{-\beta_{22}}, \\
\lim_{n\to\infty}\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{21}^k}] &\approx \mathrm{e}^{-\beta_{21}}.
\end{aligned}$$

By the law of total probability and the independence of H_{i1}^k and H_{i2}^k,

$$\begin{aligned}
\frac{\delta_1^k}{\delta_2^k} &= \frac{\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{11}^k+1}(1-(1-r_2^{\mathrm{L}})^{H_{12}^k})]}{\mathbb{E}[(1-r_2^{\mathrm{L}})^{H_{22}^k+1}(1-(1-r_1^{\mathrm{L}})^{H_{21}^k})]} \\
&= \frac{(1-r_1^{\mathrm{L}})\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{11}^k}](1-\mathbb{E}[(1-r_2^{\mathrm{L}})^{H_{12}^k}])}{(1-r_2^{\mathrm{L}})\mathbb{E}[(1-r_2^{\mathrm{L}})^{H_{22}^k}](1-\mathbb{E}[(1-r_1^{\mathrm{L}})^{H_{21}^k}])}.
\end{aligned}$$

It follows that

$$\begin{aligned}
\lim_{n\to\infty}\frac{\delta_1^k}{\delta_2^k} &\approx \frac{(1-r_1^{\mathrm{L}})\mathrm{e}^{-\beta_{11}}(1-\mathrm{e}^{-\beta_{12}})}{(1-r_2^{\mathrm{L}})\mathrm{e}^{-\beta_{22}}(1-\mathrm{e}^{-\beta_{21}})} \\
&\approx \frac{1-r_1^{\mathrm{L}}}{1-r_2^{\mathrm{L}}}\mathrm{e}^{\beta_{22}-\beta_{11}} = \frac{1-r_1^{\mathrm{L}}}{1-r_2^{\mathrm{L}}}\mathrm{e}^{(\rho-1)\beta}.
\end{aligned}$$

□

B. Reproducibility

In this section, we describe the detailed experimental settings including (**§B.1**) data statistics, (**§B.2**) baseline settings, and (**§B.3**) evaluation protocols. The source code for implementing and evaluating BAT and all the CIGL baseline methods will be released after the paper is published.

B.1. Data Statistics

As previously described, we adopt 5 benchmark graph datasets: the Cora, CiteSeer, and PubMed citation networks (Sen et al., 2008), and the CS and Physics coauthor networks (Shchur et al., 2018) to test BAT on large graphs with more nodes and high-dimensional features. All datasets are publicly available[2]. Table 4 summarizes the dataset statistics.

Table 4. Statistics of datasets.

Dataset	#nodes	#edges	#features	#classes
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3
CS	18,333	163,788	6,805	15
Physics	34,493	495,924	8,415	5

We follow previous works (Zhao et al., 2021b; Park et al., 2022; Song et al., 2022a) to construct and adjust the class imbalanced node classification tasks. For step imbalance, we select half of the classes ($\lfloor m/2 \rfloor$) as minority classes and the rest as majority classes. We follow the public split (Sen et al., 2008) for semi-supervised node classification where each class has 20 training nodes, then randomly remove minority class training nodes until the given imbalance ratio (IR) is met. The imbalance ratio is defined as $\text{IR} = \frac{\#\text{majority training nodes}}{\#\text{minority training nodes}} \in [1, \infty)$, i.e., more imbalanced data has higher IR. For natural imbalance, we simulate the long-tail class imbalance present in real-world data by utilizing a power-law distribution. Specifically, for a given IR, the largest head class have $n_{\text{head}} = \text{IR}$ training nodes, and the smallest tail class have 1 training node. The number of training nodes of the k-th class is determined by $n_k = \lfloor n_{\text{head}}^{\lambda_k} \rfloor, \lambda_k = \frac{m-k}{m-1}$. We set the IR (largest class to smallest class) to 50/100 to test BAT's robustness under natural and extreme class imbalance. We show the training data distribution under step and natural imbalance in Fig. 8.

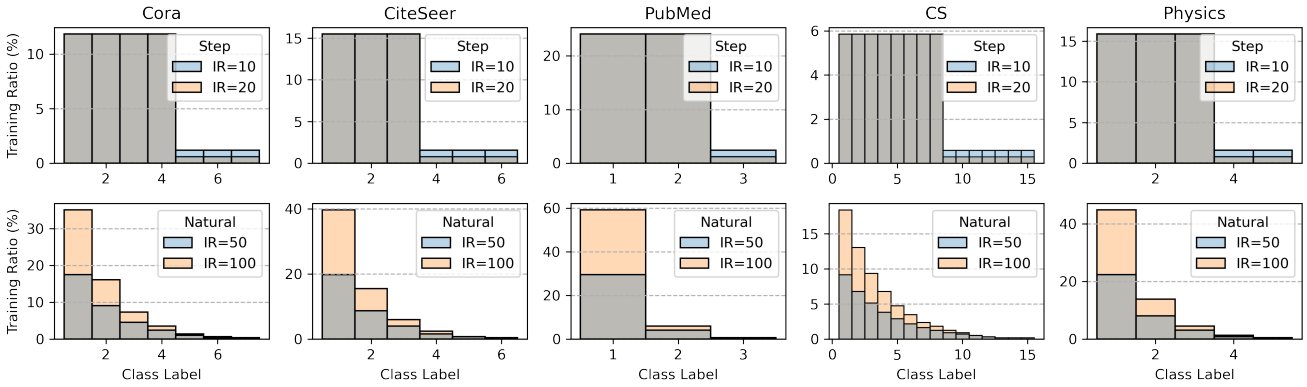


Figure 8. Class distribution of training datasets under step and natural imbalance.

B.2. Baseline Settings

To fully validate BAT's performance and compatibility with existing CIGL techniques and GNN backbones, we include six baseline methods with five popular GNN backbones in our experiments, and combine BAT with them under all possible combinations. The included CIGL baselines can be generally divided into two categories: reweighting-based (i.e., Reweight (Japkowicz & Stephen, 2002), ReNode (Chen et al., 2021)) and augmentation-based (i.e., Oversampling (Japkowicz & Stephen, 2002), SMOTE (Chawla et al., 2002), GraphSMOTE (Zhao et al., 2021b), and GraphENS (Park et al., 2022)).

- Reweight (Japkowicz & Stephen, 2002) assigns minority classes with higher misclassification costs (i.e., weights in the loss function) by the inverse of the class frequency in the training set.
- ReNode (Chen et al., 2021) measures the influence conflict of training nodes, and perform instance-wise node reweighting to alleviate the topology imbalance.

[2] https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html.

- Oversample (Japkowicz & Stephen, 2002) augments minority classes with additional synthetic nodes by replication-base oversampling.
- SMOTE (Chawla et al., 2002) synthesizes minority nodes by 1) randomly selecting a seed node, 2) finding its k-nearest neighbors in the feature space, and 3) performing linear interpolation between the seed and one of its k-nearest neighbors.
- GraphSMOTE (Zhao et al., 2021b) extends SMOTE (Chawla et al., 2002) to graph-structured data by 1) performing SMOTE in the low-dimensional embedding space of GNN and 2) utilizing a learnable edge predictor to generate better topology connections for synthetic nodes.
- GraphENS (Park et al., 2022) directly synthesize the whole ego network (node with its 1-hop neighbors) for minority classes by similarity-based ego network combining and saliency-based node mixing to prevent neighbor memorization.

Baseline implementation details. We use the public implementations[3][4][5] of the baseline methods for a fair comparison. For ReNode (Chen et al., 2021), we use its transductive version and search hyperparameters among the lower bound of cosine annealing $w_{min} \in \{0.25, 0.5, 0.75\}$ and upper bound of the cosine annealing $w_{max} \in \{1.25, 1.5, 1.75\}$ following the original paper. We set the teleport probability of PageRank $\alpha = 0.15$ as given by the default setting in the released implementation. As Oversample (Cui et al., 2019) and SMOTE (Chawla et al., 2002) were not proposed to handle graph data, we adopt their enhanced versions provided by GraphSMOTE (Zhao et al., 2021b), which also duplicate the edges from the seed nodes to the synthesized nodes in order to connect them to the graph. For GraphSMOTE (Zhao et al., 2021b), we use the version that predicts edges with binary values as it performs better than the variant with continuous edge predictions in many datasets. For GraphENS (Park et al., 2022), we follow the settings in the paper: Beta$(2, 2)$ distribution is used to sample λ, the feature masking hyperparameter k and temperature τ are tuned among $\{1, 5, 10\}$ and $\{1, 2\}$, and the number of warmup epochs is set to 5.

Combining BAT and baseline CIGL techniques. Since BAT only manipulates the graph data and remains independent of the model architecture, it seamlessly integrates with the aforementioned CIGL techniques. During each training epoch, BAT enhances the original graph $\mathcal{G}$ using the current model f and yields the augmented graph $\mathcal{G}^*$. Subsequently, other CIGL methods operate on the augmented graph $\mathcal{G}^*$. Specifically, loss function engineering methods (Reweight and ReNode) perform loss computation and backpropagation based on $\mathcal{G}^*$, and data augmentation methods (Resampling, SMOTE, GSMOTE, GENS) carry out additional class-balancing operations on $\mathcal{G}^*$, generating new minority nodes based on its structure.

GNN backbone implementation details. We use `pytorch` (Paszke et al., 2019) and `torch_geometric` (Fey & Lenssen, 2019) to implement all five GNN backbones used in this paper, i.e., GCN (Welling & Kipf, 2016), GAT (Veličković et al., 2018), GraphSAGE (Hamilton et al., 2017), APPNP (Gasteiger et al., 2018), and GPRGNN (Chien et al., 2020). Most of our settings are aligned with prevailing works (Park et al., 2022; Chen et al., 2021; Song et al., 2022a) to obtain fair and comparable results. Specifically, we implement all GNNs' convolution layer with ReLU activation and dropout (Srivastava et al., 2014) with a dropping rate of 0.5 before the last layer. For GAT, we set the number of attention heads to 4. For APPNP and GPRGNN, we follow the best setting in the original paper and use 2 `APPNP`/`GPR_prop` convolution layers with 64 hidden units. Note that GraphENS's official implementation requires modifying the graph convolution for resampling and thus cannot be directly combined with APPNP and GPRGNN. The teleport probability = 0.1 and the number of power iteration steps K = 10. We search for the best architecture for other backbones from #layers $l \in \{1, 2, 3\}$ and hidden dimension $d \in \{64, 128, 256\}$ based on the average of validation accuracy and F1 score. We train each GNN for 2,000 epochs using Adam optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.01. To achieve better convergence, we follow (Park et al., 2022) to use 5e-4 weight decay and adopt the `ReduceLROnPlateau` scheduler in Pytorch, which reduces the learning rate by half if the validation loss does not improve for 100 epochs.

B.3. Evaluation Protocol

To evaluate the predictive performance on class-imbalanced data, we use two balanced metrics, i.e., balanced accuracy (*BAcc.*) and macro-averaged F1 score (*Macro-F1*). They compute accuracy/F1-score for each class independently and use the unweighted average mean as the final score, i.e., *BAcc.* $= \frac{1}{m}\sum_{i=1}^{m} Acc(c_i)$, *Macro-F1* $= \frac{1}{m}\sum_{i=1}^{m} F1(c_i)$. Additionally, we use performance standard deviation (PerfStd) to evaluate the level of model predictive bias. Formally, let $Acc(c_i)$ be the

[3] https://github.com/victorchen96/renode
[4] https://github.com/TianxiangZhao/GraphSmote
[5] https://github.com/JoonHyung-Park/GraphENS

classification accuracy of class c_i, the PerfStd is defined as the standard deviation of the accuracy scores of all classes, i.e., $\sqrt{\frac{1}{m}\sum_{i=1}^{m}(Acc(c_i) - BAcc.)^2}$. All the experiments are conducted on a Linux server with Intel® Xeon® Gold 6240R CPU and NVIDIA® Tesla V100 32GB GPU.

C. Extended Discussions

In this section, we present an ablation study **(§C.1)** validate the effectiveness and efficiency of the key modules, then we discuss how to further speed up BAT in practice **(§C.2)**; how to choose between BAT_0 and BAT_1 in practice **(§C.3)**; and finally, the limitation and future works **(§C.4)**.

C.1. Ablation Study

We present an ablation study to validate the effectiveness and efficiency of the key modules in BAT. Specifically, for node risk estimation, we compare our total-variation-distance-based uncertainty with (i) the naïve random assignment that drawn uncertainty score from a uniform distribution $U(0, 1)$ and (ii) the information entropy $\mathbf{H}(Y) = -\sum_{y\in\mathcal{Y}} p(y)\log_2(p(y))$. We substitute the original uncertainty metric with these aforementioned methods in BAT_0, and assess their impact on performance as well as the computational time required for uncertainty estimation. It is worth noting that in practical implementation, the computation is parallelized on GPU (assuming sufficient GPU memory). Therefore, the computational time of a given uncertainty measure remains consistent across the three datasets we employed (Cora, CiteSeer, PubMed with IR=10). The detailed results are presented in Table 5, revealing that: (i) Randomly assigned uncertainty scores significantly impede the performance of BAT, resulting in a large drop in both balanced accuracy and Marco-f1. (ii) In comparison to our approach, employing information entropy as the node uncertainty score necessitates ∼2.3x computation time, yet the influence on performance remains marginal.

Table 5. Ablation study on node risk estimation of BAT.

Uncertainty	Cora		CiteSeer		PubMed		Computation Time(ms)
	BAcc	Macro-F1	BAcc	Macro-F1	BAcc	Macro-F1	
Random	61.64±1.89	59.44±1.71	46.59±2.29	44.37±3.23	61.60±1.69	58.13±1.81	0.0249
Information Entropy	65.18±1.68	63.11±1.91	51.87±2.96	50.36±3.43	67.72±1.27	67.19±1.57	0.1257
TVDistance (ours)	65.54±1.25	63.28±1.07	52.65±1.08	51.55±1.28	68.62±0.77	67.16±1.53	0.0543

Further, we conduct an ablation study for our posterior likelihood estimation strategy by comparing our 0^{th}-order (BAT_0) and 1^{st}-order (BAT_1) likelihood estimation methods with the random method that assigns (unnormalized) node-class likelihood by drawing from a uniform distribution $U(0, 1)$. Results are shown in Table 6. We can observe that the random method significantly worsens the predictive performance on all CIGL tasks. Altogether, the ablation study results confirm the effectiveness and efficiency of the design of BAT, showcasing its ability to deliver strong performance with minimal computational overhead.

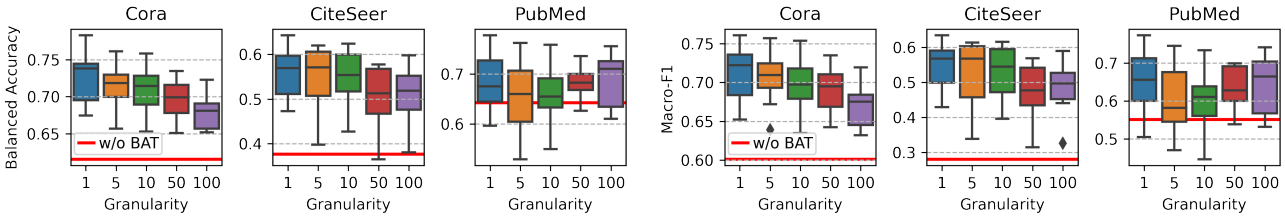
Table 6. Ablation study on posterior likelihood estimation of BAT.

Estimation	Cora		CiteSeer		PubMed		Computation Time(ms)
	BAcc	Macro-F1	BAcc	Macro-F1	BAcc	Macro-F1	
Random	63.85±2.17	61.94±2.68	46.51±3.27	41.70±4.33	64.32±1.23	53.58±2.48	0.0883
0th-order (BAT_0)	65.54±1.25	63.28±1.07	52.65±1.08	51.55±1.28	68.62±0.77	67.16±1.53	0.1251
1st-order (BAT_1)	69.80±1.30	68.68±1.49	55.37±1.39	54.94±1.44	67.57±3.22	64.40±3.68	0.3030

C.2. On the Further Speedup of BAT

As stated in the paper, thanks to its simple and efficient design, BAT can be integrated into the GNN training process to perform dynamic topology augmentation based on the training state. By default, we run BAT in every iteration of GNN training, i.e., the granularity of applying BAT is 1, as described in Alg. 1. However, we note that in practice, this granularity can be increased to further reduce the cost of applying BAT. This operation can result in a significant linear speedup ratio: setting the granularity to N reduces the computational overhead of BAT to $1/N$ of the original (i.e., Nx speedup ratio), with

minor performance degradation. This could be helpful for scaling BAT to large-scale graphs in practice. In this section, we design experiments to validate the influence of different BAT granularity (i.e., the number of iterations per each use of BAT) in real-world CIGL tasks. We set the granularity to 1/5/10/50/100 and test the performance of BAT_T with a vanilla GCN classifier on the Cora/CiteSeer/PubMed dataset with an imbalance ratio of 10. Fig. 9 shows the empirical results from 10 independent runs. The red horizontal line in each subfigure represents the baseline (vanilla GCN) performance. It can be observed that setting a larger BAT granularity is an effective way to further speed up BAT in practice. The performance drop of adopting this trick is relatively minor, especially considering the significant linear speedup ratio it brings. The predictive performance boost brought by BAT is still significant even with a large granularity at 100 (i.e., with 100x BAT speedup).



(a) Influence of BAT granularity on BAcc. (b) Influence of BAT granularity on Macro-F1.

Figure 9. Influence of the BAT granularity (i.e., the number of iterations per each use of BAT). Note that this brings a linear speedup ratio in practice, e.g., granularity = 100 $\Leftrightarrow$ 100x speedup.

C.3. Choosing between BAT_0 and BAT_1

In this section, we summarize the strengths and limitations of BAT_0 and BAT_1, and give suggestions for choosing between them in practice. In short, we recommend using BAT_1 to achieve better classification performance. But in case that computational resources are limited, BAT_0 can serve as a more efficient alternative. The reasons are as follows:

Performance. We observe a noticeable performance gap between BAT_0 and BAT_1, wherein BAT_1 consistently demonstrates superior classification performance due to its incorporation of local topological structure. Across the 15 scenarios outlined in Table 1 (the best scores for 3 datasets x 5 GNN backbones): (1) BAT_1 outperforms BAT_0 significantly in 12 out of 15 scenarios for BAcc/F1 scores, with an average F1 advantage of 1.692 in the 11 leading cases. (2) Conversely, BAT_0 exhibits a less pronounced advantage in the instances where it outperforms BAT_1, with an average advantage of 0.518 in the 4 leading scenarios[6].

Efficiency. On the other hand, it's worth noting that BAT_1 generally incurs higher time and space complexity compared to BAT_0. Specifically, BAT_0 demonstrates linear complexity concerning the number of nodes, whereas BAT_1 exhibits linear growth in complexity with the number of edges. Given that real-world graph data often features a significantly larger number of edges than nodes, BAT_0 is usually the more efficient option (especially for densely connected graphs).

C.4. Limitations and Future Works

One potential limitation of the proposed BAT framework is its reliance on exploiting model prediction for risk and likelihood estimation. This strategy may not provide accurate estimation when the model itself exhibits extremely poor predictive performance. However, this rarely occurs in practice and can be prevented by more careful fine-tuning of parameters and model architectures. In addition to this, as described in Section 3, we adopt several fast measures to estimate node uncertainty, prediction risk, and posterior likelihood for the sake of efficiency. Other techniques for such purposes (e.g., deterministic (Liu et al., 2020c; Zhao et al., 2020)/Bayesian (Zhang et al., 2019; Hasanzadeh et al., 2020)/Jackknife (Kang et al., 2022a) uncertainty estimation) can be easily integrated into the proposed BAT framework, although the computational efficiency might be a major bottleneck. How to exploit alternative uncertainty/risk measures while retaining computational efficiency is an interesting future direction.

Beyond the class imbalance in the node label distribution, graph data can also exhibit multi-facet skewness in other aspects. For instance, class imbalance may also exist in edge-level (e.g., in edge classification/prediction (Cai et al., 2021; Pandey

[6]Despite BAT_1 holding a relative performance edge, both BAT_0 and BAT_1 substantially enhance the performance of the *best-performing* CIGL baseline methods. Over the 15 scenarios, BAT_0 yields an average improvement of 4.789/5.848 in the best BAcc/F1, while BAT_1 brings an average improvement of 5.805/6.950.

Table 7. Experiments on large-scale graphs, the numbers in parentheses are the performance gain brought by BAT.

Metric	Dataset	Setting	CIGL Baseline						
			ERM	Reweight	ReNode	Resample	SMOTE	GraphSMOTE	GraphENS
BAcc.↑	CoraFull	Base	36.12	40.93	40.10	35.71	35.59	39.25	43.76
		+BAT$_0$	39.40 (+3.28)	43.81 (+2.88)	42.78 (+2.68)	40.77 (+5.05)	40.31 (+4.72)	43.71 (+4.46)	46.20 (+2.44)
		+BAT$_1$	40.88 (+4.76)	43.77 (+2.84)	42.82 (+2.72)	40.67 (+4.95)	41.04 (+5.46)	43.70 (+4.45)	47.19 (+3.44)
	OGBN-arXiv	Base	32.20	35.69	OOM	32.24	32.18	OOM	OOM
		+BAT$_0$	34.36 (+2.16)	37.59 (+1.90)	OOM	37.20 (+4.96)	36.88 (+4.69)	OOM	OOM
		+BAT$_1$	36.57 (+4.37)	39.28 (+3.60)	OOM	37.36 (+5.12)	37.50 (+5.32)	OOM	OOM
Macro-F1↑	CoraFull	Base	33.54	38.86	37.98	32.84	32.70	37.70	41.28
		+BAT$_0$	37.25 (+3.71)	41.21 (+2.35)	40.58 (+2.61)	39.04 (+6.19)	38.35 (+5.65)	41.25 (+3.55)	43.90 (+2.61)
		+BAT$_1$	38.58 (+5.04)	41.16 (+2.30)	40.37 (+2.39)	38.48 (+5.64)	39.18 (+6.48)	41.75 (+4.05)	44.61 (+3.33)
	OGBN-arXiv	Base	29.90	32.14	OOM	30.16	29.96	OOM	OOM
		+BAT$_0$	32.42 (+2.52)	34.51 (+2.38)	OOM	34.50 (+4.34)	34.44 (+4.48)	OOM	OOM
		+BAT$_1$	33.99 (+4.08)	34.78 (+2.64)	OOM	34.55 (+4.38)	34.69 (+4.74)	OOM	OOM

*OOM: Out-Of-Memory on a NVIDIA® Tesla V100 32GB GPU.

et al., 2019)) and graph-level (e.g., in graph classification/alignment (Zeng et al., 2023a; 2024; Yan et al., 2021b)). Handling class imbalance can be even more challenging on evolving/dynamic graphs (Fu & He, 2021; Yan et al., 2021a). Beyond the quantity imbalance among classes, skewness may also exists in the topological structure, such as degree imbalance (Kang et al., 2022b), and motif-level imbalance (Zhao et al., 2022; Zeng et al., 2023b). How to jointly consider the multi-facet node/edge/graph-level imbalance to benefit more graph learning tasks is an exciting future direction. Finally, recent work on fairness-aware graph learning (Kang et al., 2020; Fu et al., 2023) has observed that the topology of the graph can also introduce bias/discrimination towards certain groups: extending BAT's concept to mitigate group unfairness through topology augmentation poses an intriguing future direction.

D. Additional Experimental Results and Analysis

D.1. Results on Additional Large-scale Graphs

To further test the scalability of BAT and the baseline CIGL methods, we extended the experiment to two large-scale graph datasets, *CoraFull* (19,793 nodes, 126,842 edges) (Bojchevski & Günnemann, 2017) and *arXiv* (169,343 nodes, 1,166,243 edges) (Hu et al., 2020). To ensure comprehensive and fair comparisons, we followed the same protocol as Table 1 to evaluate six baseline methods and the performance gains brought by BAT. We used GCN as the backbone. It is worth noting that CIGL on these large graphs is a highly challenging task due to (i) the inherent task complexity (with 70/172 classes), (ii) the label scarcity for numerous minority classes, and (iii) the requirement for the algorithm's scalability. This may be the reason why most existing works (ReNode (Chen et al., 2021), GraphSMOTE (Zhao et al., 2021b), GraphENS (Park et al., 2022), TAM (Song et al., 2022a), LTE4G (Yun et al., 2022), etc.) did not consider these large datasets. Nevertheless, we conduct the experiments report the results in Table 7. We can observe that:

- BAT can scale to large-scale CIGL tasks, consistently and significantly enhancing the performance of various CIGL baselines (overall with 10% relative performance gain).
- The linear complexity of BAT makes it scalable to large graphs, while GraphENS, ReNode, and GraphSMOTE face scalability issues when applied to the arXiv dataset (with 169,343 nodes) due to their $O(n^2)$ space complexity.

D.2. Comparison with Additional CIGL Baselines

In the main results, we chose representative *model-independent* CIGL baselines to test BAT's ability to cooperate and boost various CIGL techniques. In this section, we compare BAT with other representative CIGL techniques that are either model-dependent or with design constraints making them incompatible with BAT. Specifically, we further include three CIGL baselines LTE4G (Yun et al., 2022), GraphMixup (Wu et al., 2022), TAM (Song et al., 2022a), and compare them with BAT on seven datasets (including the newly introduced large-scale graphs *CoraFull* and *arXiv* in Section D.1). We use GCN as the backbone, and follow the main experiment protocol used in Table 1 to ensure fair and comparable results.

The results are reported in Table 8. In short, we observe that: (i) BAT consistently demonstrates better performance and scalability compared to the additional baselines. (ii) Some methods (such as GraphMixup and LTE4G) exhibit higher space complexity and thus cannot scale to large datasets. (iii) Among the baseline methods, LTE4G achieves the overall best

performance due to its three-stage training (encoder pre-training, expert training, student training) and divide-and-conquer strategy. We note that its complex training and inference strategies introduce additional computational overhead and make it challenging to integrate with other methods. In comparison, BAT has better compatibility and can further achieve better performance with existing class-rebalancing techniques.

Table 8. Comparison with independent CIGL baselines. We use **bold**/*italics* to mark the best/second-best results.

Metric	Method	Dataset						
		Cora	**CiteSeer**	**PubMed**	**CS**	**Physics**	**CoraFull**	**arXiv**
BAcc.↑	GCN	61.6	37.6	64.2	75.4	80.1	32.2	36.1
	GraphMixup	64.1	50.8	OOM	OOM	OOM	OOM	OOM
	LTE4G	*68.0*	*51.7*	63.7	*77.2*	81.7	**36.6**	OOM
	TAM	65.3	50.1	*65.7*	77.0	*82.8*	35.0	*36.3*
	BAT (Ours)	**69.8**	**55.4**	**68.6**	**82.6**	**87.6**	**36.6**	**40.1**
Macro-F1↑	GCN	60.1	28.1	55.1	72.7	80.7	33.5	29.9
	GraphMixup	62.7	47.3	OOM	OOM	OOM	OOM	OOM
	LTE4G	*67.1*	*49.7*	61.7	*76.7*	82.1	*37.7*	OOM
	TAM	65.7	43.4	*63.5*	75.2	*82.3*	32.8	*31.2*
	BAT (Ours)	**68.7**	**54.9**	**67.2**	**78.6**	**88.8**	**38.6**	**34.0**

*OOM: Out-Of-Memory on a NVIDIA® Tesla V100 32GB GPU.

D.3. Full Empirical Results with Additional GNN Backbones

Due to space limitation, we report the key results of our experiments in Table 1 and 2. We now provide complete results for all settings with the standard error of 5 independent runs. Specifically, Table 9 & 10 & 11 complement Table 1, and Table 12 complements Table 2. We further include APPNP (Gasteiger et al., 2018) and GPRGNN (Chien et al., 2020) as additional GNN backbones for supported CIGL techniques. Note that the official codebase of (Park et al., 2022) only implemented GraphENS using modified (to enable saliency-based mixup) GCN, GAT, and GraphSAGE backbones, and extending it to other GNN backbones is difficult. The results indicate that BAT can consistently boost various CIGL baselines with all GNN backbones under different types and levels of class imbalance, which aligns with our conclusions in the paper.

Table 9. Balanced accuracy of combining BAT with 6 IGL baselines × 5 GNN backbones.

Dataset (IR=10)		Cora			CiteSeer			PubMed		
Metric: BAcc.↑		Base	+ BAT$_0$	+ BAT$_1$	Base	+ BAT$_0$	+ BAT$_1$	Base	+ BAT$_0$	+ BAT$_1$
GCN	Vanilla	61.56±1.24	65.54±1.25	**69.80**±1.30	37.62±1.61	52.65±1.08	**55.37**±1.39	64.23±1.55	**68.62**±0.77	67.57±3.22
	Reweight	67.65±0.64	70.97±1.28	**72.14**±0.72	42.49±2.66	57.91±0.98	**58.36**±1.09	71.20±2.33	**74.19**±1.12	73.37±0.96
	ReNode	66.60±1.33	71.37±0.62	**71.84**±1.25	42.57±1.05	57.47±0.62	**59.28**±0.59	71.52±2.16	**73.20**±0.71	72.53±1.62
	Resample	59.48±1.53	72.51±0.68	**74.24**±0.91	39.15±2.05	57.90±0.33	**58.78**±1.44	64.97±1.94	72.53±0.85	**72.87**±1.16
	SMOTE	58.27±1.05	72.16±0.53	**73.89**±1.06	39.27±1.90	60.06±0.81	**61.97**±1.19	64.41±1.95	**73.17**±0.84	73.13±0.77
	GSMOTE	67.99±1.37	68.52±0.81	**71.55**±0.50	45.05±1.95	**57.68**±1.03	57.65±1.18	73.99±0.88	73.09±1.30	**76.57**±0.42
	GENS	70.12±0.43	72.22±0.57	**72.58**±0.58	56.01±1.17	60.60±0.63	**62.67**±0.42	73.66±1.04	76.11±0.60	**76.91**±1.03
	Best	70.12	72.51	**74.24**	56.01	60.60	**62.67**	73.99	76.11	**76.91**
GAT	Vanilla	61.53±1.13	66.27±0.83	**70.13**±1.07	39.25±1.84	55.66±1.23	**60.34**±1.66	65.46±0.69	73.19±0.86	**74.75**±1.18
	Reweight	66.94±1.24	**71.80**±0.48	71.61±0.85	41.29±3.39	59.33±0.51	**61.23**±0.99	68.37±1.41	**75.30**±1.07	74.52±1.14
	ReNode	66.81±0.98	**72.14**±1.24	70.31±1.38	43.25±1.78	58.26±1.98	**59.05**±0.88	71.18±2.13	**75.55**±1.01	75.22±0.84
	Resample	57.76±1.73	71.90±0.88	**73.29**±1.08	35.97±1.42	60.10±1.26	**60.33**±0.75	65.14±0.86	73.27±0.61	**73.89**±0.40
	SMOTE	58.81±0.64	70.50±0.44	**72.19**±0.75	36.95±1.86	60.59±1.19	**62.36**±1.18	64.81±1.47	73.90±0.68	**74.08**±0.51
	GSMOTE	64.68±1.02	69.29±1.82	**71.14**±0.96	41.82±1.14	56.11±1.23	**57.71**±2.58	68.72±1.69	**74.65**±0.65	74.41±1.57
	GENS	69.76±0.45	70.63±0.40	**71.02**±1.22	51.50±2.21	60.95±1.51	**63.49**±0.75	73.13±1.18	74.34±0.35	**75.65**±0.82
	Best	69.76	72.14	**73.29**	51.50	60.95	**63.49**	73.13	75.55	**75.65**
SAGE	Vanilla	59.17±1.23	66.24±0.92	**66.53**±0.80	42.96±0.28	**54.99**±2.51	53.18±2.90	67.56±0.84	75.31±0.93	**77.38**±0.68
	Reweight	63.76±0.89	70.15±1.15	**71.14**±0.84	45.91±2.05	**57.95**±0.73	55.90±0.93	68.03±1.69	74.56±0.41	**75.39**±0.38
	ReNode	65.32±1.07	71.31±1.29	**71.54**±0.85	48.55±2.31	56.32±0.40	**56.49**±1.73	69.08±2.04	74.24±0.20	**75.28**±0.69
	Resample	57.77±1.35	71.24±1.08	**73.01**±1.02	39.37±1.40	61.41±1.11	**61.93**±1.40	69.22±1.28	74.91±1.09	**75.80**±0.39
	SMOTE	58.81±1.97	70.31±1.35	**73.02**±2.29	38.42±1.69	64.14±0.75	**66.35**±0.70	64.96±1.56	74.59±0.96	**77.31**±0.45
	GSMOTE	61.57±1.78	69.88±0.96	**72.28**±1.48	42.21±2.12	60.91±1.33	**62.32**±1.06	71.55±0.64	74.74±0.81	**76.14**±0.21
	GENS	68.84±0.41	69.78±1.18	**71.92**±0.71	52.57±1.78	**64.36**±0.68	63.84±0.68	71.38±0.99	75.89±1.17	**76.46**±1.29
	Best	68.84	71.31	**73.02**	52.57	64.36	**66.35**	71.55	75.89	**77.38**
APPNP	Vanilla	55.37±1.65	58.13±1.69	**61.71**±1.66	35.69±0.14	35.68±0.15	**36.02**±0.25	**59.30**±0.50	55.62±0.31	57.82±0.29
	Reweight	72.62±0.47	**73.62**±0.89	72.51±0.87	50.88±3.64	63.54±1.02	**65.57**±1.11	72.00±0.81	**72.15**±0.60	71.22±1.10
	ReNode	73.74±1.12	**75.02**±1.54	72.15±0.76	50.50±3.51	63.73±0.54	**65.13**±0.40	**72.76**±1.37	71.54±0.96	71.88±0.70
	Resample	65.78±1.72	73.14±0.94	**73.57**±0.92	40.79±1.87	**66.54**±0.49	59.51±4.16	67.74±1.94	72.25±0.81	**74.41**±0.95
	SMOTE	65.34±1.68	**73.18**±1.02	72.88±0.90	40.79±2.05	**66.62**±0.33	58.82±4.59	67.24±2.10	72.67±1.65	**73.33**±1.37
	GSMOTE	71.13±0.72	73.37±0.82	**73.78**±0.71	45.37±2.75	**64.95**±0.11	62.95±2.58	69.57±2.20	73.37±0.95	**74.90**±1.27
	Best	73.74	**75.02**	73.78	50.88	**66.62**	65.57	72.76	73.37	**74.90**
GPRGNN	Vanilla	67.97±0.51	71.99±1.14	**73.38**±1.18	42.31±2.16	55.85±0.89	**58.82**±1.91	67.04±1.82	57.92±0.45	**77.49**±1.15
	Reweight	72.15±0.57	72.90±1.33	**73.22**±0.55	53.22±2.89	59.78±0.76	**61.00**±1.82	73.35±1.07	75.22±1.02	**76.86**±0.76
	ReNode	73.38±0.67	73.71±0.57	**73.93**±1.60	54.66±2.82	59.69±0.73	**60.34**±1.31	73.56±0.98	75.69±1.10	**76.25**±0.67
	Resample	67.00±1.33	72.94±1.02	**74.89**±0.86	42.27±2.15	**64.16**±0.62	63.89±0.98	70.42±1.51	73.79±0.83	**75.31**±0.54
	SMOTE	66.99±1.33	74.01±1.51	**74.41**±1.05	40.97±2.02	**63.88**±0.55	62.60±1.72	70.29±1.47	73.89±0.69	**75.48**±1.02
	GSMOTE	70.94±0.57	73.63±1.25	**74.02**±0.90	48.01±3.28	**63.03**±0.92	61.68±0.86	71.51±1.91	72.16±0.58	**74.77**±0.83
	Best	73.38	74.01	**74.89**	54.66	**64.16**	63.89	73.56	75.69	**77.49**

Table 10. Macro-F1 score of combining BAT with 6 IGL baselines × 5 GNN backbones.

Dataset (IR=10)		Cora			CiteSeer			PubMed		
Metric: Macro-F1↑		Base	+ BAT$_0$	+ BAT$_1$	Base	+ BAT$_0$	+ BAT$_1$	Base	+ BAT$_0$	+ BAT$_1$
GCN	Vanilla	60.10±1.53	63.28±1.07	**68.68**±1.49	28.05±2.53	51.55±1.28	**54.94**±1.44	55.09±2.48	**67.16**±1.53	64.40±3.68
	Reweight	67.85±0.62	69.41±1.01	**70.31**±0.82	36.59±3.66	56.84±1.06	**57.54**±1.08	67.07±3.42	72.94±0.81	**73.24**±0.90
	ReNode	66.66±1.59	69.79±0.79	**70.59**±1.25	34.64±1.54	56.69±0.64	**58.07**±0.77	67.86±3.99	**72.61**±0.41	72.25±0.89
	Resample	57.34±2.27	71.36±0.39	**72.82**±1.13	29.73±2.77	57.17±0.48	**58.03**±1.42	56.74±3.54	71.19±0.83	**73.13**±1.33
	SMOTE	55.65±1.62	71.04±0.16	**72.82**±0.86	29.39±2.81	59.53±0.88	**61.53**±1.24	56.14±3.74	71.72±0.60	**72.83**±1.20
	GSMOTE	67.60±1.67	68.01±1.00	**70.28**±0.48	40.07±3.02	**56.64**±1.09	56.25±1.50	70.60±1.17	72.95±1.39	**75.70**±0.35
	GENS	69.96±0.29	71.62±0.64	**72.28**±0.65	54.45±1.69	59.89±0.68	**62.46**±0.43	71.28±1.84	75.77±0.55	**76.86**±0.93
	Best	69.96	71.62	**72.82**	54.45	59.89	**62.46**	71.28	75.77	**76.86**
GAT	Vanilla	60.71±1.61	64.27±0.95	**68.93**±0.79	31.12±3.15	54.71±1.18	**59.42**±1.55	57.32±1.55	71.27±1.11	**74.03**±1.08
	Reweight	66.49±1.34	**69.84**±0.91	69.79±0.77	34.94±4.09	58.53±0.68	**60.28**±1.12	63.82±1.60	**75.13**±1.13	73.88±1.38
	ReNode	67.27±1.23	**70.61**±0.83	68.24±1.48	37.72±2.61	57.64±2.11	**58.57**±0.75	67.38±3.22	74.88±0.99	**74.96**±1.18
	Resample	55.36±2.47	70.87±0.94	**72.31**±1.07	25.71±1.97	**59.77**±1.31	59.66±0.95	57.24±1.54	72.53±0.66	**73.09**±0.83
	SMOTE	57.49±0.60	69.68±0.66	**71.74**±1.03	26.05±2.30	59.83±1.33	**61.75**±1.30	55.66±2.76	**73.33**±1.00	73.30±0.16
	GSMOTE	64.34±1.69	68.23±1.80	**69.77**±1.08	35.07±1.77	55.86±1.10	**57.13**±2.69	63.35±2.92	**74.23**±0.84	73.34±2.06
	GENS	69.96±0.62	69.83±0.41	**70.71**±1.16	48.34±2.19	60.04±1.85	**62.55**±0.86	71.78±1.19	72.69±0.84	**74.42**±1.12
	Best	69.96	70.87	**72.31**	48.34	60.04	**62.55**	71.78	**75.13**	74.96
SAGE	Vanilla	57.36±1.77	64.90±0.87	**65.61**±0.97	36.07±1.06	**54.76**±2.47	51.86±3.25	63.75±1.24	74.35±0.74	**76.92**±0.63
	Reweight	63.72±1.10	69.06±0.90	**69.59**±0.53	39.64±2.57	**57.17**±0.76	54.83±0.74	62.83±2.57	73.88±0.40	**75.42**±0.48
	ReNode	65.59±1.44	**69.99**±1.35	69.86±1.27	44.20±3.68	55.41±0.48	**55.78**±1.63	64.97±3.00	74.33±0.20	**74.88**±0.53
	Resample	55.29±2.12	70.40±1.11	**71.49**±0.79	30.14±2.20	60.71±1.25	**61.29**±1.48	65.23±2.26	74.28±0.96	**75.48**±0.44
	SMOTE	56.72±2.69	69.42±1.29	**71.71**±1.94	29.22±2.33	63.61±0.87	**65.91**±0.68	57.60±3.22	72.98±0.69	**76.45**±0.77
	GSMOTE	59.44±2.25	69.10±0.95	**71.30**±1.47	34.86±3.46	60.53±1.27	**61.96**±1.12	67.23±0.61	74.36±1.02	**75.68**±0.31
	GENS	68.23±0.72	69.76±0.95	**71.11**±0.81	51.05±2.03	**63.87**±0.82	63.41±0.57	70.06±0.86	75.33±1.46	**76.01**±1.14
	Best	68.23	70.40	**71.71**	51.05	63.87	**65.91**	70.06	75.33	**76.92**
APPNP	Vanilla	50.39±2.81	54.19±2.58	**59.99**±2.49	22.21±0.13	22.54±0.25	**22.89**±0.22	44.50±0.21	**44.67**±0.07	44.59±0.06
	Reweight	72.63±0.53	**72.71**±0.60	70.61±0.65	45.25±4.85	63.08±1.03	**65.20**±1.20	69.53±1.14	72.24±0.58	**72.26**±0.80
	ReNode	73.67±0.98	**73.67**±1.18	69.79±0.72	44.91±4.99	62.97±0.78	**64.47**±0.40	70.65±1.66	**72.33**±0.90	72.18±0.55
	Resample	65.20±2.08	72.25±0.82	**72.72**±0.97	31.04±2.76	**66.06**±0.54	54.57±6.08	62.42±3.62	72.32±0.93	**74.27**±1.08
	SMOTE	64.70±2.06	**72.90**±0.83	72.31±0.94	30.90±2.86	**66.18**±0.37	53.90±6.26	61.83±3.65	72.55±1.61	**73.87**±1.37
	GSMOTE	71.20±0.67	73.02±0.74	**73.22**±0.92	37.90±4.29	**64.56**±0.18	60.41±3.84	65.65±3.06	72.54±0.85	**74.61**±1.36
	Best	73.67	**73.67**	73.22	45.25	**66.18**	65.20	70.65	72.55	**74.61**
GPRGNN	Vanilla	67.86±0.79	70.80±1.16	**72.32**±1.18	35.00±2.96	55.06±0.89	**56.31**±2.87	59.01±3.62	50.12±1.46	**77.62**±1.04
	Reweight	**71.66**±0.85	70.46±0.98	71.24±0.49	49.19±3.61	59.11±0.73	**60.30**±2.04	71.18±0.95	75.47±0.90	**77.01**±0.52
	ReNode	**73.08**±0.66	71.52±0.50	71.72±1.51	50.34±3.18	**59.10**±0.75	58.94±1.36	71.45±1.19	75.08±1.06	**75.76**±0.84
	Resample	66.42±1.65	71.70±0.86	**73.54**±0.83	32.60±2.71	**63.59**±0.65	63.12±1.06	66.58±2.08	73.66±0.86	**75.42**±0.35
	SMOTE	66.43±1.74	72.89±1.23	**73.47**±1.12	31.38±2.70	**63.41**±0.55	61.23±2.59	66.78±1.97	73.98±0.70	**75.63**±0.91
	GSMOTE	70.87±0.53	72.53±0.85	**73.12**±0.95	42.82±4.52	**62.09**±1.04	60.82±0.88	67.93±3.01	72.72±0.72	**74.66**±0.74
	Best	73.08	72.89	**73.54**	50.34	**63.59**	63.12	71.45	75.47	**77.62**

Table 11. Performance deviation of combining BAT with 6 IGL baselines × 5 GNN backbones.

Dataset (IR=10)		**Cora**			**CiteSeer**			**PubMed**		
Metric: PerfStd↓		Base	+ BAT$_0$	+ BAT$_1$	Base	+ BAT$_0$	+ BAT$_1$	Base	+ BAT$_0$	+ BAT$_1$
GCN	Vanilla	27.88±1.79	21.27±1.76	**18.49**±2.68	29.93±1.38	**13.82**±2.06	13.93±0.80	34.73±2.14	**9.23**±2.78	21.81±4.51
	Reweight	22.29±1.41	**14.43**±2.51	18.32±2.20	25.47±1.78	**19.10**±1.48	22.64±0.79	19.33±5.26	10.21±1.58	**5.88**±0.83
	ReNode	22.88±1.64	**14.65**±2.07	17.00±2.13	30.31±1.51	**20.22**±0.88	22.99±1.09	18.14±5.79	12.99±1.57	**10.96**±1.92
	Resample	31.57±1.85	**15.13**±2.14	15.25±2.79	31.00±1.32	**16.30**±1.89	20.79±0.43	30.90±5.67	11.63±3.20	**7.82**±0.80
	SMOTE	33.32±1.38	**16.33**±1.12	17.95±2.50	32.61±1.45	**17.27**±0.86	18.25±0.89	31.79±5.21	**10.56**±1.82	11.66±2.58
	GSMOTE	21.78±1.79	**17.90**±2.75	18.44±2.20	22.64±2.69	21.37±1.25	**21.01**±1.45	15.87±2.34	**3.35**±1.08	5.83±1.27
	GENS	20.04±1.12	**16.98**±3.02	18.02±2.23	16.95±2.64	**14.94**±0.75	15.54±0.60	11.93±3.46	5.95±1.85	**5.15**±0.80
	Best	20.04	**14.43**	15.25	16.95	**13.82**	13.93	11.93	**3.35**	5.15
GAT	Vanilla	27.38±1.71	19.23±0.80	**17.97**±2.65	28.32±2.07	**15.62**±0.77	15.90±0.95	30.94±1.27	10.77±2.04	**8.51**±2.48
	Reweight	22.90±1.67	**16.44**±2.71	17.32±2.83	30.27±1.26	**18.64**±1.30	20.83±1.06	24.92±1.88	**3.01**±0.96	5.44±1.33
	ReNode	23.13±1.54	**15.05**±1.59	18.96±1.65	25.21±1.85	**20.48**±0.82	20.51±0.49	18.15±4.37	**4.77**±1.22	6.17±0.42
	Resample	32.73±2.12	17.87±2.04	**17.64**±2.57	32.59±0.89	**17.76**±1.79	18.73±1.02	31.67±0.98	6.18±1.35	**4.58**±1.14
	SMOTE	31.17±0.69	18.40±1.03	**18.26**±1.87	33.32±0.88	**10.68**±0.68	13.24±1.12	32.79±2.13	8.14±2.00	**7.56**±1.05
	GSMOTE	24.84±1.60	**15.48**±2.08	18.23±2.00	26.74±1.53	**18.34**±1.64	19.76±0.42	24.50±2.78	**5.12**±1.38	8.54±2.03
	GENS	20.08±1.56	**17.75**±2.40	17.88±2.50	26.49±1.18	**12.89**±0.77	15.09±0.95	10.29±2.75	7.83±2.27	**7.55**±2.38
	Best	20.08	**15.05**	17.32	25.21	**10.68**	13.24	10.29	**3.01**	4.58
SAGE	Vanilla	29.94±1.75	**18.62**±2.13	19.49±1.67	26.75±1.58	**14.56**±1.16	18.13±1.32	21.09±3.43	10.96±1.99	**4.09**±1.17
	Reweight	25.61±1.60	**15.24**±2.66	17.54±2.45	29.95±1.83	**19.05**±1.60	22.94±0.49	25.47±3.49	**3.35**±0.72	8.09±0.19
	ReNode	24.12±1.73	**13.32**±3.03	15.45±2.41	22.41±4.31	**22.20**±0.97	22.75±0.87	22.92±4.36	7.63±1.23	**5.77**±1.55
	Resample	31.66±1.47	15.77±2.75	**15.08**±2.74	30.29±1.16	18.72±0.90	**18.48**±2.00	21.41±2.88	**4.68**±1.42	4.76±1.09
	SMOTE	30.86±2.64	17.30±2.09	**14.87**±3.23	32.07±1.00	13.17±1.33	**12.78**±0.43	31.62±2.86	13.88±1.44	**11.63**±2.28
	GSMOTE	27.71±1.86	17.28±2.25	**16.10**±2.94	28.77±2.61	18.69±0.76	**18.05**±1.21	20.10±0.90	5.37±1.21	**4.64**±1.61
	GENS	19.81±1.65	**17.50**±2.05	17.63±2.11	19.76±2.07	**15.99**±0.81	16.99±0.85	11.76±2.91	**7.63**±1.51	8.31±1.64
	Best	19.81	**13.32**	14.87	19.76	13.17	**12.78**	11.76	**3.35**	4.09
APPNP	Vanilla	38.32±1.94	35.50±2.10	**32.18**±1.68	36.82±0.10	**36.67**±0.25	36.83±0.36	42.13±0.27	**40.45**±0.11	41.34±0.16
	Reweight	19.83±1.46	**17.33**±2.88	18.46±2.42	26.19±2.93	20.96±0.58	**19.38**±0.72	16.96±2.84	**8.04**±1.94	9.13±1.05
	ReNode	18.09±2.52	**16.87**±2.95	19.47±2.00	25.95±3.66	22.09±1.43	**20.42**±1.57	14.49±3.81	10.25±1.93	**3.95**±1.02
	Resample	27.28±2.13	**18.37**±2.34	18.72±2.32	32.71±1.23	**15.87**±1.02	23.72±4.14	25.86±4.38	13.60±1.68	**9.49**±1.65
	SMOTE	27.86±1.78	**18.61**±2.35	19.42±1.81	33.26±1.10	**14.91**±0.93	22.90±4.49	26.37±4.47	13.37±1.97	**8.70**±2.29
	GSMOTE	20.98±1.45	**18.19**±2.59	18.55±2.28	29.39±2.20	**16.49**±1.12	19.19±3.44	22.32±4.21	11.53±3.00	**10.69**±2.27
	Best	18.09	**16.87**	18.46	25.95	**14.91**	19.19	14.49	8.04	**3.95**
GPRGNN	Vanilla	22.96±1.20	18.12±2.29	**17.00**±2.98	27.57±1.32	**17.10**±1.17	20.94±2.58	29.94±3.68	36.57±1.46	**5.30**±0.91
	Reweight	20.94±1.21	**17.83**±2.82	19.67±1.81	22.43±2.39	21.52±1.06	**20.03**±1.81	16.12±1.84	7.54±0.49	**5.48**±1.49
	ReNode	18.84±2.19	**16.78**±2.53	17.89±2.96	24.14±1.47	**19.84**±1.79	22.83±1.40	14.40±3.18	9.75±2.20	**6.61**±1.47
	Resample	25.62±1.80	19.23±2.26	**17.61**±2.77	33.08±0.66	17.04±0.78	**15.98**±0.93	22.59±2.75	**7.62**±2.50	7.76±0.85
	SMOTE	25.44±1.88	**16.97**±3.19	17.38±2.78	32.85±0.95	**15.09**±1.23	16.85±2.51	21.35±2.76	9.41±2.67	**6.09**±0.62
	GSMOTE	21.23±1.48	**18.02**±2.62	19.06±2.28	24.21±3.06	**14.83**±0.95	19.11±1.73	20.08±3.77	**5.99**±1.49	8.27±0.75
	Best	18.84	**16.78**	17.00	22.43	**14.83**	15.98	14.40	5.99	**5.30**

Table 12. Performance of BAT under varying types and levels of class imbalance. For each setting, we report the relative gain over base and mark the best/second-best score in **bold**/underlined.

	Dataset	Cora		CiteSeer		PubMed		CS		Physics	
	Step IR	10	20	10	20	10	20	10	20	10	20
BAcc.↑	Base	61.6	52.7	37.6	34.2	64.2	60.8	75.4	65.3	80.1	67.7
	+ BAT	69.8+13.4%	71.3+35.2%	55.4+47.2%	51.3+49.9%	68.6+6.8%	63.3+4.1%	82.6+9.6%	79.9+22.2%	87.6+9.4%	88.0+29.9%
	BestIGL	70.1+13.9%	66.5+26.2%	56.0+48.9%	47.2+38.0%	74.0+15.2%	71.1+17.0%	84.1+11.6%	81.3+24.4%	89.4+11.6%	85.7+26.6%
	+ BAT	**74.2**+20.6%	**71.6**+35.9%	**62.7**+66.6%	**62.5**+82.6%	**76.9**+19.7%	**75.7**+24.5%	**86.3**+14.5%	**85.6**+31.0%	**91.2**+13.9%	**90.9**+34.2%
Macro-F1↑	Base	60.1	47.0	28.1	21.9	55.1	46.4	72.7	59.2	80.7	64.7
	+ BAT	68.7+14.3%	69.6+48.1%	54.9+95.8%	48.9+123.5%	67.2+21.9%	60.7+30.8%	78.6+8.1%	74.7+26.1%	88.8+10.0%	87.8+35.8%
	BestIGL	70.0+16.4%	66.2+40.9%	54.5+94.1%	45.0+105.6%	71.3+29.4%	68.9+48.3%	83.9+15.3%	80.9+36.7%	89.5+10.9%	86.2+33.2%
	+ BAT	**72.8**+21.2%	**70.2**+49.4%	**62.5**+122.7%	**62.1**+183.6%	**76.9**+39.5%	**74.9**+61.2%	**85.4**+17.5%	**84.6**+43.0%	**90.7**+12.4%	**90.0**+39.2%
PerfStd↓	Base	27.9	39.0	29.9	35.1	34.7	41.5	21.2	32.1	22.2	36.0
	+ BAT	21.3-23.7%	24.4-37.5%	**13.9**-53.5%	**16.7**-52.5%	21.8-37.2%	29.1-29.9%	17.4-18.2%	22.9-28.8%	11.5-48.3%	25.6-29.0%
	BestIGL	20.0-28.1%	21.9-43.8%	16.9-43.4%	18.0-48.6%	11.9-65.6%	14.2-65.7%	8.9-58.3%	12.3-61.8%	**6.3**-71.7%	12.4-65.5%
	+ BAT	**15.2**-45.3%	**17.5**-55.2%	13.9-53.5%	16.7-52.5%	**5.1**-85.2%	**4.6**-89.0%	**7.9**-62.7%	**10.1**-68.5%	6.6-70.2%	**6.9**-80.8%
	Natural IR	50	100	50	100	50	100	50	100	50	100
BAcc.↑	Base	58.1	61.8	44.9	44.7	52.0	51.1	73.8	71.4	76.0	77.7
	+ BAT	69.1+18.9%	68.3+10.6%	58.4+29.9%	57.4+28.5%	55.6+7.0%	56.5+10.4%	82.1+11.3%	81.9+14.8%	86.9+14.3%	84.1+8.3%
	BestIGL	71.0+22.3%	73.8+19.5%	56.3+25.3%	56.3+26.0%	72.7+39.8%	72.8+42.5%	81.2+10.0%	81.4+14.0%	85.8+12.9%	87.2+12.2%
	+ BAT	**73.1**+25.8%	**76.9**+24.5%	**62.1**+38.2%	**61.3**+37.3%	**75.8**+45.7%	**75.9**+48.5%	**85.0**+15.1%	**84.5**+18.5%	**88.6**+16.5%	**89.7**+15.4%
Macro-F1↑	Base	58.7	61.4	37.5	36.2	47.3	45.1	75.3	73.2	78.0	79.8
	+ BAT	68.7+17.1%	67.5+10.0%	57.1+52.6%	55.8+54.3%	52.8+11.6%	52.0+15.4%	82.6+9.7%	82.6+12.8%	87.6+12.3%	85.2+6.8%
	BestIGL	71.1+21.2%	73.4+19.5%	54.3+44.8%	53.8+48.8%	72.9+53.9%	73.7+63.6%	82.5+9.5%	82.4+12.6%	87.7+12.4%	88.3+10.6%
	+ BAT	**72.7**+23.9%	**76.0**+23.9%	**60.2**+60.8%	**59.4**+64.3%	**75.3**+59.2%	**76.1**+68.8%	**85.7**+13.7%	**85.1**+16.2%	**88.8**+13.8%	**89.4**+12.0%
PerfStd↓	Base	28.8	31.0	38.7	39.8	36.2	38.2	26.3	28.2	23.8	21.0
	+ BAT	18.3-36.4%	25.4-18.1%	24.9-35.6%	33.1-17.0%	33.3-8.1%	35.9-6.2%	19.0-27.9%	19.5-30.9%	17.0-28.7%	19.6-6.7%
	BestIGL	18.9-34.4%	17.3-44.4%	28.7-25.9%	29.7-25.3%	6.0-83.4%	9.6-75.0%	14.4-45.4%	15.4-45.5%	11.2-53.1%	9.7-53.8%
	+ BAT	**15.9**-44.8%	**14.7**-52.8%	**21.9**-43.4%	**19.8**-50.3%	**4.2**-88.3%	**5.6**-85.3%	**12.2**-53.5%	**12.8**-54.7%	**7.4**-68.9%	**7.2**-65.7%