U2B: Scale-unbiased Representation Converter for Graph Classification with Imbalanced and Balanced Scale Distributions

Guanjun Wang[1], Jianhao Zhang[3], Jiaming Ma[1], Sheng Huang[1], Pengkun Wang[1,2], Zhengyang Zhou[1,2], Binwu Wang[1,2,*], Yang Wang[1,2,*]

[1] University of Science and Technology of China, Hefei 230236, China
[2] Suzhou Institute for Advanced Research, University of Science and Technology of China, Suzhou 215123, China
[3] Shanghai Jiao Tong University, Shanghai, 200240, China
{always, JiamingMa, shenghuang}@mail.ustc.edu.cn,
zhangjianhao_0715@sjtu.edu.cn, {wbw2024, pengkun, zzy0929, angyan}@ustc.edu.cn

Abstract

Graph classification is a critical task in analyzing graph data, with applications across various domains. While graph neural networks (GNNs) have achieved remarkable results, their ability to generalize across graphs of varying scales remains a challenge. Conventional models often perform well on large-scale graphs but struggle with distributions that are skewed towards small scales. Conversely, models tailored to address scale imbalances frequently prioritize small-scale graphs, leading to diminished performance in more balanced scenarios. To overcome these limitations, we introduce a **U**nbalanced-**B**alanced Representation Converter (U2B), which exhibits no explicit bias toward graph scales. U2B employs a two-step workflow: a distillation phase to extract base features from both node-level and graph-level representations, followed by a refinement phase to generate unbiased representations for improved balance. In the distillation phase, a static constraint guides node-level adjustments, improving the representation of nodes in small graphs. Simultaneously, a dynamic constraint in the graph-level process mitigates biases toward features from large graphs. To ensure harmony between the representations, a consistency alignment loss is introduced, aligning node-level and graph-level features to create more cohesive and balanced graph representations. Extensive experiments on multiple datasets show that U2B achieves competitive performance.

Code — https://github.com/ALWAYS1815/U2B

Introduction

In recent years, Graph Neural Networks (GNNs) have been widely applied across various domains, such as graph classification tasks, owing to their powerful message-passing mechanisms that effectively capture and represent hierarchical graph structures (Hamilton 2020; Sun et al. 2022b; Guo et al. 2025). The goal is to predict graph-level labels from each graph's structure and node/edge attributes (You et al. 2020; Fu et al. 2021).

In real-world applications, graph data often exhibit complex variations in scale. As shown in Figure 1(a), three commonly used graph classification datasets highlight these differences. For instance, datasets like PROTEINS and PTC-MR follow a power-law distribution, characterized by a few large "head" graphs and numerous small "tail" graphs, forming a long-tail pattern. In contrast, the NCI1 dataset displays a normal distribution, lacking the long-tail characteristics.

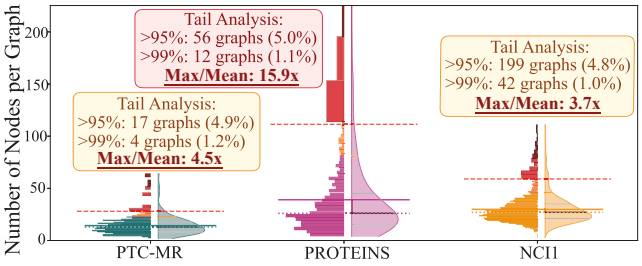

Figure 1: Graph-scale distribution of different datasets.

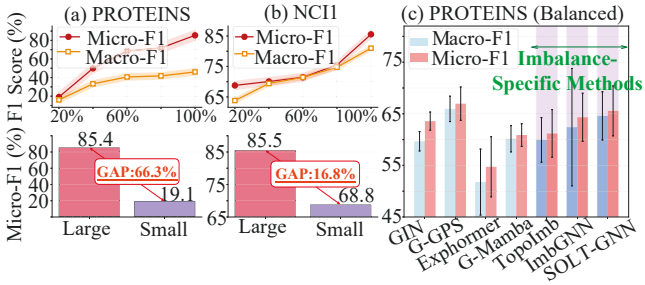

Figure 2: Graph classification performance of GIN across different graph scales on PROTEINS (a) and NCI1 (b) datasets as examples. (c) shows performance comparison between scale imbalance-specific methods and conventional models.

The diverse scale distributions in graph datasets pose significant challenges to the generalization capabilities of existing graph classification methods. Conventional approaches often prioritize large-scale graphs while neglecting small-scale ones, resulting in poor performance on datasets with long-tail characteristics (Liu et al. 2022; Qin et al. 2025). As illustrated in Figure 2(a) and (b), the predictive performance of GIN varies significantly across different graph

*Binwu Wang and Yang Wang are the corresponding authors.

scales on the PROTEINS and NCI1 datasets, with further details provided in Table 4. On the other hand, Imbalance-specific methods upweight small-scale graphs but fail in balanced settings lacking significant scale disparity, underperforming conventional models. For example, Figure 2(c) shows that such methods can sometimes underperform compared to standard GIN (Xu et al. 2018). Consequently, the development of a robust approach that maintains strong performance across both balanced and imbalanced graph scale distributions remains an urgent challenge.

To address these challenges, we propose the **U**nbalanced-**B**alanced Representation Converter (U2B). The key idea behind U2B is to distill scale-invariant base features, which are then used to reconstruct biased node-level and graph-level representations. This process creates a scale-unbiased feature space, ultimately improving performance across diverse graph classification scenarios.

U2B features a structured approach centered on extracting and reconstructing key features. The method begins by establishing learnable compressed tokens, which are used in a distillation process to adaptively extract fundamental features from input samples. These features are then refined to reconstruct representations that effectively capture the underlying graph structure. At the node level, we introduce a static balancing constraint during the distillation stage to ensure balanced activation of the compressed tokens. This prevents overemphasis on nodes from large graphs and enhances the model's ability to recognize structural patterns in small graphs. The resulting node representations are pooled to generate graph representations, which form the input for the next stage. At the graph level, we employ dynamic balancing constraints to align feature distributions across graphs of varying scales. This fosters the creation of unbiased representations that improve prediction accuracy across both balanced and imbalanced datasets. To further enhance the quality of the representations, we incorporate a consistency alignment loss, promoting compactness and semantic coherence. Our contributions of this work are as follows:

❶ To the best of our knowledge, this is the first study to systematically address graph classification across comprehensive graph scale distributions, effectively handling both balanced and imbalanced settings.

❷ We propose a novel approach, U2B, designed to generate scale-unbiased graph representations through a two-stage framework consisting of distillation and refinement. This approach incorporates two specific constraint terms to regulate the distillation process, ensuring the extraction of representative fundamental features.

❸ Our extensive experimental results on real-world datasets demonstrate that U2B can achieve competitive performance and favorable scalability.

Related Work

Graph Representation Learning Graph representation learning encodes graph data into vector forms and has been proven effective in various fields, such as transportation (Wang et al. 2024a, 2023, 2024b), atmospheric science (Ma et al. 2025), and recommendation systems (Wu et al. 2022). In this paper, we focus on the graph classification task (Sun et al. 2022a; Rampášek et al. 2022; Sun et al. 2025). Early approaches mainly relied on spectral methods, such as Laplacian operators (Noutahi et al. 2019) and random walks (Verdier et al. 2021). The success of deep learning in Euclidean domains has led to the emergence of graph neural networks (GNNs). GNNs leverage a message-passing framework that combines node features with graph topology, enabling nodes to exchange information while preserving both structural and content-related information (Sun et al. 2023). Inspired by the success of Transformer, recent research has extended these techniques to graphs, enabling more effective capture of global dependencies and structural characteristics (Yun et al. 2019). However, these methods have demonstrated suboptimal performance in graph classification scenarios involving topology-imbalanced graphs.

Imbalanced Learning for Graph Classification Graph imbalanced classification tasks can be further subdivided into class imbalance and topological imbalance (Qin et al. 2024; Wang, Ding, and Xie 2025). Methods designed for class imbalance emphasize learning from tail classes. For instance, G^2GNN (Wang et al. 2022) and ImbGNN (Xu et al. 2024) achieve data augmentation by fusing head and tail graphs. In this work, we focus on the fine-grained phenomenon of scale imbalance, which is prevalent in graph datasets. TopoImb (Zhao et al. 2022) mitigates structural imbalance by integrating topology-aware modeling and instance weighting, while SOLT-GNN (Liu et al. 2022) balances learning across graphs of different sizes by identifying co-occurring subgraph patterns. Despite these advances, such specialized models often sacrifice performance on large-scale graphs, leading to suboptimal results on datasets with non-long-tail distributions.

Problem Formulation

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{A}, X\}$ be a graph, where $\mathcal{V}$ is the node set with $|\mathcal{V}|$ nodes, $\mathcal{E}$ is the edge set, $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix, $X \in \mathbb{R}^{|\mathcal{V}|\times d}$ is the node feature matrix with d-dimension. Given a graph set $\mathbb{G} = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\}$ with N graphs with their corresponding label set $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_N\}$, the goal of graph-level classification task is to learn a mapping function $\mathcal{F} : \mathcal{G} \rightarrow \mathbb{R}^f$ to map any graph to a low-dimensional vector $h \in \mathbb{R}^f$. This representation is subsequently processed by a classifier to generate the predicted label distribution, resulting in the final output prediction for the graph sample.

Method

Overall Architecture of U2B

As illustrated in Figure 3, U2B incorporates both node-level and graph-level unbalanced-to-balanced converters to adaptively refine the generated representations, thereby producing scale-unbiased graph embeddings. The converter first extracts fundamental features from unbalanced graph data and applies targeted strategies to constrain the learning process toward balanced feature distributions, effectively alleviating scale-induced biases. Formally, for a batch of in-

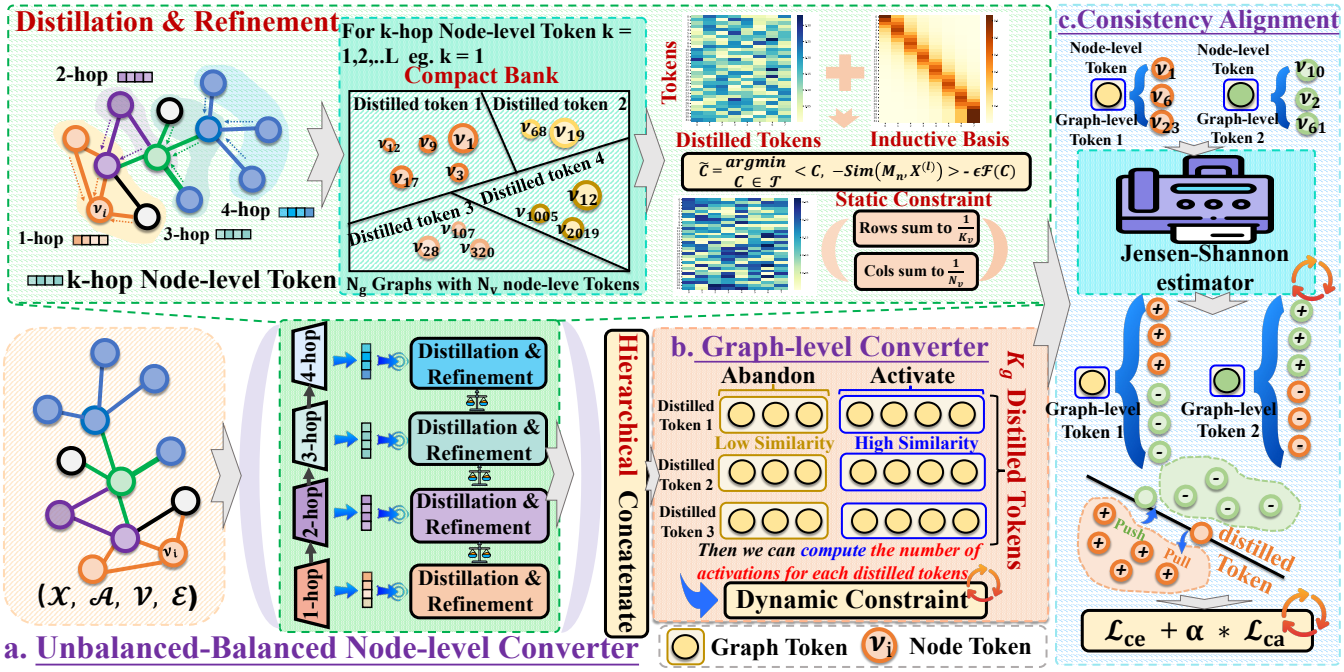

Figure 3: Overview of U2B, including (a) Node-level Converter, (b) Graph-level Converter, (c) Consistency Alignment Loss.

put graphs, we concatenate their adjacency matrices into $\mathbf{A} \in \mathbb{R}^{N_v \times N_v}$ and node features into $\mathbf{X} \in \mathbb{R}^{N_v \times d}$, where N_v is the total number of nodes in the batch and d is the feature dimensionality. We employ stacked GNN encoders (e.g., GIN (Xu et al. 2018)) to model complex node-level interactions, generating hierarchical node representations. These representations from each GNN layer are then passed to a node-level unbalanced-balanced converter. The refined node features are aggregated via a READOUT function into a graph-level representation $\mathbf{H}$, which is further processed by a graph-level converter to yield the final scale-unbiased representation $\widetilde{\mathbf{H}}$. A classifier is applied on $\widetilde{\mathbf{H}}$ for label prediction. The overall training objective combines the cross-entropy classification loss $\mathcal{L}_{ce}$ with a proposed consistency-aware loss $\mathcal{L}_{ca}$.

Unbalanced-Balanced Node-Level Representation

GNNs for Representation. GNNs are a class of deep learning models designed to operate on graph-structured data, with notable examples including GCN (Kipf and Welling 2016) and GIN (Xu et al. 2018). A key operation in GNNs is neighborhood aggregation, where each node recursively gathers and transforms information from its neighbors to construct its own representation. Given the input in the l-th GNN layer $\mathbf{X}^{(l-1)}$ with $\mathbf{X}^{(0)} = \mathbf{X}$, it can be formally expressed as follows:

$$\mathbf{X}_v^{(l)} = \mathbf{X}_v^{(l-1)} + \mathrm{AGG}\left(\left\{\left(\mathbf{X}_v^{(l-1)}, \mathbf{X}_u^{(l-1)}\right) : \mathcal{U} \in \mathcal{N}(v)\right\}\right) \tag{1}$$

where $\mathbf{X}_v^{(l)}$ denotes the representation of node v at the l-th layer. And $\mathcal{N}(v)$ represents the set of neighbors of node v in the graph. The function $\mathrm{AGG}(\cdot)$ represents an aggregation operation such as a weighted average or attention mechanism.

Node-level Converter. In GNN message passing, nodes in small graphs often aggregate insufficient neighborhood information, resulting in under-informed embeddings. To mitigate this issue, the distillation stage in U2B compresses node representations into coarse-grained base features that capture shared patterns across graphs. In the refinement stage, these features are decoded into fine-grained, node-specific representations, thus restoring representational balance.

Specifically, given the output of l-th GNN layer $\mathbf{X}^{(l)} \in \mathbb{R}^{N_v \times d}$, where N_v represents the number of nodes in the batch and d is the number of channels, the converter introduces a compact bank of node-level distilled token representations, denoted as $\mathbf{M}_n \in \mathbb{R}^{K_v \times d}$. These tokens distill base features from the input. The hyperparameter K indicates the number of compact tokens, set to $K_v \ll N_v$.

Distillation Process. We employ the self-attention mechanism to compute the similarity between the distilled token representation and node representations, enabling the model to adaptively perceive their shared components as follows:

$$\mathrm{Sim}(\mathbf{M}_n, \mathbf{X}^{(l)}) = \mathrm{Softmax}\left(\mathbf{M}_n (W_q^{(l)} \mathbf{X}^{(l)})^\top / \sqrt{d} + \psi\right) \tag{2}$$

where the generated distillation coefficient matrix is denoted as $\mathbf{Att}_n \in \mathbb{R}^{K_v \times N_v}$, and $W_q^{(l)}$ is the learnable parameter. $\psi = W^p (\mathbf{E})^\top \in \mathbb{R}^{K_v \times N_v}$, where $W^p \in \mathbb{R}^{K_v \times 3}$ is the learnable parameter and $\mathbf{E} \in \mathbb{R}^{N_v \times 3}$ encodes the degree of each node, the total number of nodes in the graph to which it belongs, and the average degree of that graph, which serves as inductive bias to guide the subsequent distillation process.

The distillation process relies on fine-grained interactions between nodes and coarse-grained entities. However, it can face the challenge of "coarse collapse," where most nodes are assigned to only a few clusters, leaving the remaining clusters sparsely populated or empty (Caron et al. 2018; Zeng et al. 2023). This imbalance obstructs the effective extraction of base patterns. Inspired by prior work (Caron et al. 2020), we incorporate a Sinkhorn-Knopp-based static constraint to ensure balanced access to each distilled token during the distillation process, preventing collapse and improving pattern extraction. Specifically, the Sinkhorn-Knopp strategy begins by establishing a constraint matrix, $\mathbf{C} \in \mathbb{R}^{K_v \times N_v}$. This matrix serves as a guide for the distillation process, ensuring balanced interactions between coarse-grained tokens and fine-grained nodes. The resulting objective is defined as:

$$\tilde{\mathbf{C}} = \underset{\mathbf{C} \in \mathcal{T}}{\operatorname{argmin}} \left\langle \mathbf{C}, -\operatorname{Sim}\left(\mathbf{M}_n, \mathbf{X}^{(l)}\right) \right\rangle - \epsilon \mathcal{F}(\mathbf{C}) \tag{3}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product between two matrices, $\mathcal{F}(\cdot)$ is the entropy function and $\mathcal{F}(\mathbf{C}) = -\sum_{\mathrm{i}=1}^{K_v} \sum_{\mathrm{j}=1}^{N_v} \mathbf{C}_{\mathrm{ij}} \log \mathbf{C}_{\mathrm{ij}}$. $\mathbf{C}_{\mathrm{ij}}$ means the element in the i-th row and j-th column of $\mathbf{C}$. ϵ controls the smoothness of the assignment (i.e., the entropy regularization strength). Following previous work (Asano, Rupprecht, and Vedaldi 2019), we enforce equal partitioning by constraining the matrix $\mathbf{C}$ within the transportation polytope. We propose to adapt their solution to mini-batches by restricting the transportation polytope to each mini-batch:

$$\mathcal{T} = \left\{ \mathbf{C} \in \mathbb{R}_{+}^{K_v \times N_v} \;\middle|\; \sum_{j=1}^{N_v} C_{ij} = \frac{1}{K_v} \,\forall\, i,\ \sum_{i=1}^{K_v} C_{ij} = \frac{1}{N_v} \,\forall\, j \right\} \tag{4}$$

These constraints ensure that, on average, each distilled token is selected at least $\frac{N_v}{K_v}$ times. In fact, solving Equation 3 corresponds to an entropy-regularized optimal transport problem. The work (Caron et al. 2020) demonstrates that obtaining a continuous solution to this problem yields better performance. Therefore, the solution $\tilde{\mathbf{C}}$ to the optimization problem of Equation 3 can be approximated by a soft assignment matrix that is normalized along both rows and columns.

$$\tilde{\mathbf{C}} = \operatorname{Diag}(\mathbf{u}) \exp\left(\operatorname{Sim}(\mathbf{M}_n, \mathbf{X}^{(l)})/\epsilon\right) \operatorname{Diag}(\mathbf{v}) \tag{5}$$

where $\mathbf{u}$ and $\mathbf{v}$ are renormalization vectors in $\mathbb{R}^{K_v}$ and $\mathbb{R}^{N_v}$ respectively, and $\epsilon > 0$ is the entropy regularization coefficient, which controls the smoothness of the transport matrix and it is a fixed value in our case.

Refinement Process. This process entails mapping the extracted coarse-grained base representations back to fine-grained node representations during distillation, for which we continue to employ the attention mechanism.

Finally, the computation of the node-level converter, which includes the distillation and refinement processes, can be formulated as follows:

$$\widetilde{\mathbf{X}}^{(l)} = \sigma\left(W_q^{(l)} \mathbf{X}^{(l)} \left(\mathbf{M}_n\right)^{\top} / \sqrt{d}\right) \tilde{\mathbf{C}} W_v^{(l)} \mathbf{X}^{(l)} \tag{6}$$

where σ is the Sigmoid function, $W_q^{(l)}$ and $W_v^{(l)}$ are learnable parameters. $\widetilde{\mathbf{X}}^{(l)} \in \mathbb{R}^{N_v \times d}$ is the calibrated output representation, which will serve as the input for the next layer $\mathbf{X}^{(l+1)}$. Finally, we stack L GNN layers and concatenate the outputs of each layer to form node representations:

$$\overline{\mathbf{X}} = \operatorname{CONCAT}\left(\left\{\widetilde{\mathbf{X}}^{(l)}\right\}_{l=1}^{L}\right) \in \mathbb{R}^{N_v \times (L \times d)} \tag{7}$$

Unbalanced-Balanced Graph-Level Representation

Graph-level Converter. After passing through L GNN layers, the output node representations $\overline{\mathbf{X}}$ are pooled using the READOUT function (Sun et al. 2019) to generate the graph-level representation $\mathbf{H} \in \mathbb{R}^{N_g \times d}$, where N_g represents the number of input graphs. Despite pooling, the representations of large-scale and small-scale graphs still differ in information content, which may lead to biased learning in the final model. To address this, we introduce a two-stage graph-level converter that distills essential features at the graph level to enhance the representations of small-scale graphs. Analogous to previous steps, the process begins by initializing the graph-level distilled token representations $\mathbf{M}_g \in \mathbb{R}^{K_g \times d}$.

Distillation Process. The graph-level distilled attention is defined as follows:

$$\operatorname{Sim}(\mathbf{M}_g, \mathbf{H}) = \operatorname{Softmax}\left(\operatorname{TopK}\left(\mathbf{M}_g \left(W_q^g \mathbf{H}\right)^{\top} / \sqrt{d} + \gamma\right)\right) \tag{8}$$

where W_q^g is the learnable parameter. We retain the TopK largest attention coefficients, as the number of graphs is much smaller than the number of nodes, making this approach more effective for distilling the base features. The generated distillation coefficient matrix is denoted as $\mathbf{Att}_g \in \mathbb{R}^{K_g \times N_g}$, and $\gamma \in \mathbb{R}^{K_g}$ is a dynamic constraint designed to prevent attention from being concentrated on only a few tokens.

Specifically, we first calculate the number of times each distillation token is activated. For k-th distilled token, u_k means the number of nonzero elements in the k-th row of the matrix $\mathbf{Att}_g$. To achieve balanced activation of the distillation tokens, the expected activation frequency for each distilled token is set to $\widetilde{\mu} = \frac{N_g \times \mathrm{TopK}}{K_g}$. γ captures the difference between $\widetilde{\mu}$ and μ, incorporating it as an intermediate term in the learning process. Since the discrete number of activations is non-differentiable, the update of γ is optimized subject to the following constraints as follows:

$$\gamma \leftarrow \gamma - \eta \nabla \mathcal{G}\left(\widetilde{\mu}, \mu\right), \tag{9}$$

$$\nabla \mathcal{G}\left(\widetilde{\mu}, \mu\right) = \begin{cases} \widetilde{\mu} - \mu, & \text{if } |\widetilde{\mu} - \mu| \leq \delta \\ \delta \operatorname{sign}\left(\widetilde{\mu} - \mu\right), & \text{otherwise} \end{cases} \tag{10}$$

where η is the learning rate, and δ is smoothing threshold. $\operatorname{sign}(\cdot)$ is a function of the numerical sign. When the activation count of a particular token exceeds the expected average, γ imposes an additional penalty on the subsequent attention coefficients, thereby ensuring balanced activation across all tokens.

Unlike enforcing strict balance on the distillation coefficients in the node-level converter using the Sinkhorn algorithm, we adopt the aforementioned dynamic constraint strategy to adjust the attention weights in a more flexible manner, thereby capturing the representative base patterns in the graph structure more effectively.

Refinement Process. Similarly, during the distillation process, we map the extracted coarse-grained base representations back to fine-grained node representations, for which we continue to use the TopK attention mechanism.

Finally, the calculation of the graph-level converter can be written as:

$$\widetilde{\mathbf{H}} = \sigma\left(\mathrm{TopK}\left(W_q^g \mathbf{H}\left(\mathbf{M}_g\right)^\top / \sqrt{d}\right)\right) \mathrm{Sim}(\mathbf{M}_g, \mathbf{H}) W_v^g \mathbf{H} \tag{11}$$

where σ is the Sigmoid function. Then, the representation $\widetilde{\mathbf{H}}$ is fed into a decoder to predict the label of the graphs.

Optimizing Loss with Consistency Alignment. To generate compact graph representations, we propose a consistency constraint optimization that aligns node-level representations with their corresponding graph-level representations while repelling the graph-level representations of non-corresponding graphs. The loss can be formulated as:

$$\begin{aligned}\mathcal{L}_{ca} = -\sum_{\mathcal{G}\in\mathbb{G}} \frac{1}{|\mathcal{G}|} \sum_{\mathcal{V}\in\mathcal{G}} \Big[\mathbb{E}\left[-\log\left(1+\mathrm{e}^{-\Gamma\left(\overline{\mathbf{X}}_{\mathcal{V}}, \widetilde{\mathbf{H}}_{\mathcal{G}}\right)}\right)\right] \\ +\mathbb{E}\left[\log\left(1+\mathrm{e}^{-\Gamma\left(\overline{\mathbf{X}}_{\mathcal{V}}, \widetilde{\mathbf{H}}_{\mathcal{G}'}\right)}\right)\right]\Big]\end{aligned} \tag{12}$$

where $\overline{\mathbf{X}}_{\mathcal{V}} \in \mathbb{R}^d$ represents the representation of node $\mathcal{V}$. $\widetilde{\mathbf{H}}_{\mathcal{G}}$ and $\widetilde{\mathbf{H}}_{\mathcal{G}'}$ denote the representation of the graph to which node $\mathcal{V}$ belongs and the representation of non-corresponding graphs, respectively. $\Gamma(\cdot)$ is the Jensen-Shannon estimator (Nowozin, Cseke, and Tomioka 2016) to express the affinity between two representations. The final loss function integrates the cross-entropy classification loss $\mathcal{L}_{ce}$ and the consistency alignment loss $\mathcal{L}_{ca}$:

$$\mathcal{L} = \mathcal{L}_{ce} + \alpha * \mathcal{L}_{ca} \tag{13}$$

where α is the loss balance factor.

Experiment

Experiment Setup

Datasets. We conduct graph classification experiments on 6 real-world widely-adopted graph datasets including 4 binary-class datasets and 2 multi-class datasets (Wang et al. 2022; Qin et al. 2024). These datasets are collected from various domains such as chemistry, protein biology, and social networks. We summarized the statistics in Table 1.

Dataset	# Graphs	Avg. #Nodes	# Features	# Classes	Avg. #Degree
PTC-MR	344	14.29	18	2	2.06
PROTEINS	1113	39.06	3	2	3.73
NCI1	4110	29.87	37	2	2.16
REDDIT-B	2000	429.63	-	2	2.32
COLLAB	5000	74.49	-	3	65.97
IMDB-MULTI	1500	13.00	-	3	10.14

Table 1: Statistics of the graph classification datasets.

Baselines. Our experiments include six categories of baseline methods, comprising a total of **16** models. These methods include *(1) Graph kernel methods*: GK (Graphlet Kernel) (Shervashidze et al. 2009) and WL (Weisfeiler-Lehman subtree kernel) (Shervashidze et al. 2011). *(2) Classic graph neural networks*: GIN (Xu et al. 2018), GIN$^+$ (Luo, Shi, and Wu 2025), GraphSAGE, and GCN. *(3) Advanced graph neural networks*: DiffPool (Ying et al. 2018),CurGraph (Wang et al. 2021a), and Mixup (Wang et al. 2021b), and DGCNN (Zhang et al. 2018). *(4) Graph contrastive learning methods*: InfoGraph (Sun et al. 2019) and GraphCL (You et al. 2020). *Graph Transformer methods*: GraphGPS (Rampášek et al. 2022), Exphormer (Shirzad et al. 2023), and G-Mamba (Wang et al. 2024c). *(5) Topology imbalance-oriented methods*: TopoImb (Zhao et al. 2022), ImbGNN (Xu et al. 2024), and SOLT-GNN (Liu et al. 2022). We use GIN as the backbone for all models, whenever applicable.

Setting. We conduct evaluations under both scale-imbalanced and scale-balanced scenarios, following the protocols of the latest graph imbalance learning benchmark, IGL-Bench (Qin et al. 2024). For each dataset, the graphs with the largest 20% scale are defined as *large-scale*, while the remaining 80% are defined as *small-scale*. We split each dataset into 10% for training, 10% for validation, and 80% for testing. Subsequently, we sample to generate scale-imbalanced data, with an imbalance ratio of up to 12:1. Our model is implemented in PyTorch and trained on a 40GB NVIDIA A100 GPU. We use the Adam optimizer (Kingma 2014) with a learning rate of 0.001. Hyperparameters for all models are selected via Bayesian optimization on the validation set. All experiments are repeated for 10 runs, each with 1000 epochs, and early stopping is applied. For evaluation, we adopt the widely used Macro-F1 and Micro-F1 metrics in graph classification tasks (Wang et al. 2022). The hyperparameter η is set to 0.001. The search space for the number of distilled tokens K_v in the node-level compact token bank and K_g in the graph-level distilled token bank is set to $[1, 128]$, and the TopK selection threshold for graph-level compact tokens is searched in the range $[1, K_g]$.

Experimental Result Analysis

Scale-imbalance Graph Classification. Table 2 presents the overall performance, while Table 4 reports the fine-grained performance on large-scale and small-scale graphs. Specifically, GIN and most graph kernel methods perform the worst on scale-imbalanced datasets, exposing their limitations. Graph pooling methods (e.g., DGCNN and DiffPool) achieve better performance than GIN in classifying small-scale graphs. Although CurGraph leverages curriculum learning, it exhibits instability on certain datasets. Graph Transformer (GTs) models (such as Exphormer and G-Mamba) further enhance GIN with global attention mechanisms, thus alleviating the small-scale graph issue. Counterintuitively, topology-specific models do not achieve dom-

Model	PROTEINS		NCI1		PTC-MR		REDDIT-B		COLLAB		IMDB-MULTI	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GK	55.41 ± 2.30	60.77 ± 1.64	63.19 ± 0.76	63.33 ± 0.57	36.84 ± 7.50	51.56 ± 5.59	68.48 ± 12.26	71.74 ± 10.46	28.56 ± 7.97	32.24 ± 7.95	27.09 ± 6.82	33.68 ± 3.02
WL	58.62 ± 2.24	59.89 ± 2.25	44.20 ± 10.87	53.63 ± 4.07	35.69 ± 8.32	50.24 ± 9.27	65.29 ± 4.37	66.33 ± 5.61	14.95 ± 6.20	30.80 ± 15.87	16.60 ± 0.20	33.16 ± 0.53
GIN	53.48 ± 2.03	58.00 ± 4.19	61.60 ± 2.20	61.84 ± 2.29	34.56 ± 6.32	48.41 ± 7.07	66.60 ± 2.27	67.41 ± 2.23	62.83 ± 1.41	64.10 ± 1.53	20.80 ± 4.91	34.73 ± 2.16
CurGraph	41.89 ± 6.97	62.13 ± 1.25	63.24 ± 2.28	59.78 ± 1.37	35.76 ± 8.13	43.44 ± 3.46	64.33 ± 2.35	65.75 ± 4.10	24.11 ± 0.85	56.65 ± 0.11	16.50 ± 0.31	32.89 ± 1.25
Mixup	63.36 ± 6.28	65.57 ± 4.33	49.49 ± 16.16	57.81 ± 7.88	32.55 ± 5.75	45.17 ± 7.24	63.64 ± 0.12	67.99 ± 0.07	21.58 ± 5.50	49.66 ± 14.78	16.67 ± 0.20	33.17 ± 0.62
DGCNN	66.56 ± 2.31	71.43 ± 1.73	60.39 ± 2.53	61.74 ± 1.40	41.30 ± 9.49	46.84 ± 6.01	58.47 ± 12.77	63.57 ± 7.67	34.98 ± 17.93	47.29 ± 15.60	32.21 ± 6.97	37.62 ± 2.62
DiffPool	68.38 ± 4.20	72.05 ± 3.61	66.97 ± 1.83	67.39 ± 1.66	49.02 ± 4.57	50.56 ± 4.86	63.47 ± 6.47	63.75 ± 6.38	49.85 ± 7.01	50.96 ± 7.55	28.55 ± 6.83	36.36 ± 2.96
InfoGraph	55.31 ± 3.30	60.68 ± 3.52	62.25 ± 1.53	62.69 ± 2.01	44.68 ± 8.14	48.33 ± 4.91	69.56 ± 3.46	69.63 ± 4.28	63.28 ± 1.68	66.20 ± 1.33	33.36 ± 1.60	36.42 ± 1.10
GraphCL	57.62 ± 5.50	61.91 ± 3.84	63.62 ± 5.43	64.01 ± 4.75	41.82 ± 6.67	49.75 ± 5.25	67.25 ± 7.92	68.31 ± 6.36	61.58 ± 4.16	59.88 ± 4.01	30.39 ± 2.53	33.90 ± 1.57
GraphGPS	65.54 ± 4.22	69.26 ± 2.48	62.96 ± 3.51	64.02 ± 2.40	44.14 ± 6.86	49.97 ± 6.71	66.16 ± 4.19	68.42 ± 5.32	24.11 ± 8.74	56.65 ± 2.81	16.87 ± 0.53	33.49 ± 0.62
Exphormer	64.01 ± 2.18	67.33 ± 2.78	65.16 ± 3.19	65.23 ± 7.31	46.45 ± 5.30	50.66 ± 4.56	66.48 ± 14.59	67.81 ± 7.45	21.03 ± 7.89	36.71 ± 15.27	25.52 ± 5.03	33.60 ± 2.70
G-Mamba	68.46 ± 3.91	72.09 ± 3.16	64.09 ± 2.82	64.63 ± 2.14	42.27 ± 5.86	50.69 ± 6.99	64.81 ± 12.47	67.06 ± 8.99	13.64 ± 5.97	27.37 ± 14.82	17.63 ± 3.07	33.21 ± 0.52
GIN$^+$	59.99 ± 3.50	60.93 ± 3.65	65.77 ± 2.80	65.90 ± 2.76	43.86 ± 7.23	49.13 ± 5.15	69.02 ± 1.05	69.19 ± 1.10	67.20 ± 1.99	69.13 ± 2.13	33.53 ± 0.45	38.54 ± 0.61
TopoImb	43.82 ± 11.74	57.07 ± 13.48	63.57 ± 1.69	64.17 ± 1.33	52.29 ± 2.09	52.61 ± 2.18	67.66 ± 3.93	69.29 ± 3.47	66.33 ± 1.38	68.27 ± 1.69	32.75 ± 5.19	40.50 ± 2.97
ImbGNN	66.87 ± 2.59	67.07 ± 2.75	61.74 ± 1.89	61.92 ± 1.10	43.28 ± 8.46	52.94 ± 4.69	64.55 ± 6.88	66.49 ± 6.86	49.23 ± 1.94	50.22 ± 1.82	22.21 ± 4.90	33.18 ± 1.02
SOLT-GNN	69.34 ± 3.05	71.38 ± 3.34	60.14 ± 2.38	60.58 ± 2.04	40.70 ± 3.27	51.74 ± 5.25	56.69 ± 8.70	58.62 ± 9.02	62.46 ± 4.80	65.05 ± 5.77	34.25 ± 2.71	39.58 ± 2.00
Ours	**83.14** ± 2.08	**83.42** ± 2.15	**80.23** ± 3.77	**80.36** ± 2.70	**54.47** ± 0.45	**56.08** ± 1.91	**72.62** ± 4.03	**73.94** ± 3.09	**75.57** ± 2.76	**80.45** ± 2.22	**41.85** ± 2.28	**45.05** ± 1.49

Table 2: Performance comparison on **scale-imbalance** datasets. The best results are **highlighted** in bold, while the second-best results are underlined. We report the average and standard deviation over 10 runs.

Model	PROTEINS		NCI1		PTC-MR		REDDIT-B		COLLAB		IMDB-MULTI	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GK	56.70 ± 6.67	59.68 ± 2.27	59.37 ± 6.99	62.66 ± 3.74	43.91 ± 3.41	46.88 ± 5.01	66.60 ± 5.53	67.94 ± 4.67	59.20 ± 0.83	61.07 ± 1.10	23.20 ± 4.21	34.86 ± 1.47
WL	60.43 ± 1.40	64.42 ± 0.79	54.15 ± 7.71	57.01 ± 3.36	46.60 ± 4.22	47.78 ± 3.99	64.12 ± 3.21	65.29 ± 4.17	14.76 ± 7.31	31.10 ± 18.90	16.70 ± 0.24	33.42 ± 0.65
GIN	59.67 ± 1.87	63.58 ± 1.76	61.89 ± 5.08	63.30 ± 3.19	48.26 ± 3.11	52.17 ± 4.36	66.14 ± 1.90	67.41 ± 2.32	58.45 ± 1.80	64.34 ± 3.55	35.91 ± 5.03	44.92 ± 3.66
CurGraph	42.25 ± 6.55	60.22 ± 6.54	52.32 ± 1.43	52.88 ± 1.62	49.70 ± 1.43	51.04 ± 1.65	62.73 ± 3.58	64.59 ± 4.18	26.78 ± 2.12	58.65 ± 3.31	16.55 ± 0.15	33.03 ± 0.39
Mixup	56.60 ± 0.60	62.23 ± 0.38	42.42 ± 13.88	54.09 ± 6.28	44.43 ± 2.88	46.25 ± 4.26	65.40 ± 10.14	67.01 ± 11.69	24.48 ± 6.47	56.65 ± 17.25	16.64 ± 0.23	33.29 ± 0.60
DGCNN	58.59 ± 13.15	62.27 ± 9.25	46.21 ± 12.22	53.77 ± 6.01	45.15 ± 5.81	49.62 ± 5.19	46.08 ± 8.71	46.90 ± 8.29	43.74 ± 12.72	50.27 ± 7.37	22.12 ± 4.88	33.52 ± 1.27
DiffPool	55.65 ± 3.38	56.90 ± 2.79	65.01 ± 1.97	77.71 ± 1.36	46.94 ± 4.03	47.92 ± 3.32	56.48 ± 13.94	61.79 ± 9.55	28.32 ± 6.52	28.50 ± 6.07	29.29 ± 3.29	35.41 ± 2.89
InfoGraph	61.18 ± 2.22	64.27 ± 2.58	63.44 ± 4.42	64.69 ± 4.10	47.19 ± 3.73	51.07 ± 2.84	71.31 ± 5.94	72.22 ± 6.73	60.03 ± 4.26	63.75 ± 4.82	37.22 ± 6.35	41.83 ± 7.01
GraphCL	63.72 ± 3.03	65.08 ± 4.29	65.08 ± 3.27	66.82 ± 3.19	48.57 ± 4.23	52.90 ± 4.65	70.48 ± 4.28	71.75 ± 5.35	58.47 ± 7.88	62.82 ± 6.38	36.98 ± 5.16	40.81 ± 6.48
GraphGPS	65.95 ± 2.46	66.96 ± 3.23	63.20 ± 0.94	63.46 ± 0.81	49.81 ± 2.95	50.07 ± 2.88	66.42 ± 3.21	67.95 ± 4.17	13.18 ± 8.47	28.13 ± 22.09	16.90 ± 0.20	33.95 ± 0.53
Exphormer	51.77 ± 6.39	54.73 ± 5.82	49.17 ± 8.89	54.77 ± 3.48	44.99 ± 3.29	45.97 ± 2.60	56.98 ± 6.23	57.25 ± 6.24	15.06 ± 4.85	20.75 ± 8.44	21.11 ± 4.48	33.03 ± 1.93
G-Mamba	60.16 ± 2.54	60.88 ± 2.20	64.49 ± 1.92	64.70 ± 1.70	47.85 ± 3.22	48.19 ± 2.91	67.45 ± 10.88	68.18 ± 5.78	28.65 ± 17.81	31.96 ± 16.16	16.61 ± 0.20	33.16 ± 0.53
GIN$^+$	55.23 ± 4.02	56.32 ± 4.96	66.27 ± 1.20	66.33 ± 1.14	50.20 ± 3.44	52.12 ± 4.27	65.03 ± 3.64	66.08 ± 3.80	58.38 ± 1.37	59.75 ± 1.44	40.94 ± 2.74	45.93 ± 2.86
TopoImb	59.94 ± 4.32	61.19 ± 4.61	62.41 ± 0.66	62.46 ± 0.66	48.83 ± 2.90	49.71 ± 1.98	69.15 ± 3.83	69.47 ± 3.70	60.91 ± 2.15	62.71 ± 2.42	39.09 ± 2.72	45.25 ± 1.64
ImbGNN	62.41 ± 11.37	64.33 ± 4.65	62.01 ± 0.94	62.09 ± 0.99	48.72 ± 4.22	50.33 ± 3.26	54.47 ± 9.62	58.47 ± 8.74	53.42 ± 0.74	55.23 ± 1.01	25.97 ± 5.55	37.24 ± 2.85
SOLT-GNN	64.60 ± 4.66	65.56 ± 4.83	58.75 ± 0.73	59.59 ± 0.77	43.63 ± 1.61	47.54 ± 4.33	49.17 ± 6.76	49.57 ± 6.78	59.76 ± 1.81	61.79 ± 2.09	37.59 ± 3.24	40.43 ± 3.27
Ours	**73.36** ± 2.71	**75.92** ± 1.30	**72.78** ± 1.50	**73.45** ± 2.12	**52.28** ± 1.86	**53.30** ± 1.56	**75.63** ± 2.02	**75.69** ± 2.56	**63.54** ± 2.81	**78.03** ± 1.26	**46.46** ± 2.08	**48.04** ± 1.58

Table 3: Performance comparison on **scale-balance** datasets.

inant performance. For instance, TopoImb and ImbGNN improve graph representations through topology-aware modeling and enhanced G^{2}G, respectively. Furthermore, we report the performance on large-scale and small-scale graphs in Table 4. It can be observed that these topology-imbalance methods fail to achieve the expected results on small-scale graphs, and their performance on large-scale graphs is generally inferior to that on small-scale graphs. In contrast, U2B achieves dominating performance by balancing learning at both the node and graph levels and integrating consistency loss.

Scale-balance Graph Classification. As shown in Table 3, in scale-balanced scenarios, models designed for topology-imbalanced learning perform suboptimally compared with conventional models such as GIN, indicating that their strategies may be ineffective in this setting. Graph contrastive learning methods, such as InfoGraph and GraphCL, improve performance through mutual information consistency and graph augmentation. GIN$^+$ integrates techniques like residual connections and positional encodings, outperforming most baselines. Graph Transformer models excel on binary classification tasks, but their performance is less impressive on balanced multi-class datasets. In this scenario, U2B still achieves competitive accuracy, as it effectively extracts base features that aid the model in classification.

Scalability Analysis of U2B

To demonstrate the scalability of U2B, we further evaluated it in scale-imbalanced scenarios by employing different GNN variants and Graph Transformers as backbone. The results (as shown in Table 5) indicate that all backbone variants achieved significant performance improvements when integrated with U2B. These findings validate the strong scalability of our approach, which can substantially enhance the performance of conventional graph classification models.

Ablation Study

To evaluate the contribution of each component in U2B, we create six variants: ❶ *w/o NC* means that the Node-level Converter is removed; ❷ *w/o GC* means that the Graph-level Converter is removed; ❸ *w/o Sin* means that the a Sinkhorn Knopp-based static Constraint in the Node-level Converter is removed; ❹ *w/o Dyn* means that the Dynamic Constraint in the Graph-level Converter is removed; ❺ *w/o Con* means that the Consistency Alignment Loss between graph representation and node representations is removed.

Model	PROTEINS		NCI1	
	Small-scale	Large-scale	Small-scale	Large-scale
GIN	56.12 ± 3.18	64.28 ± 3.52	58.39 ± 3.12	70.88 ± 6.01
CurGraph	54.70 ± 2.22	78.62 ± 2.25	51.20 ± 4.34	68.07 ± 6.25
DGCNN	66.28 ± 2.25	81.96 ± 0.63	56.85 ± 1.78	78.75 ± 0.12
DiffPool	68.67 ± 3.40	83.65 ± 6.20	65.93 ± 2.22	72.46 ± 6.42
InfoGraph	58.42 ± 2.67	65.10 ± 3.19	60.76 ± 3.29	70.64 ± 5.17
GraphCL	60.00 ± 2.28	65.88 ± 3.41	61.33 ± 4.67	74.75 ± 5.02
GraphGPS	66.02 ± 2.85	80.40 ± 6.46	62.81 ± 4.77	68.18 ± 7.61
Exphormer	65.42 ± 2.84	73.91 ± 9.07	63.96 ± 12.87	68.83 ± 2.48
G-Mamba	68.26 ± 3.61	85.25 ± 3.52	63.49 ± 3.60	68.60 ± 6.11
GIN+	60.32 ± 3.19	63.02 ± 7.43	65.92 ± 2.34	65.78 ± 8.22
TopoImb	55.36 ± 6.68	62.87 ± 36.75	61.72 ± 1.24	72.92 ± 1.57
ImbGNN	68.30 ± 0.88	63.23 ± 3.07	58.68 ± 1.86	70.89 ± 1.56
SOLT-GNN	67.34 ± 3.57	83.88 ± 3.64	58.52 ± 1.91	68.77 ± 7.96
Ours	**82.34** ± 1.92	**86.31** ± 2.68	**79.68** ± 2.30	**81.42** ± 2.05

Table 4: Micro-F1 Performance comparison on **Scale-imbalance** datasets.

Method	PROTEINS		NCI1	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GIN	53.48 ± 2.03	58.00 ± 4.19	61.60 ± 2.20	61.84 ± 2.29
GIN + U2B	**83.14** ± 2.08	**83.42** ± 2.15	**80.23** ± 3.77	**80.36** ± 2.70
GCN	51.29 ± 1.10	57.02 ± 5.02	60.39 ± 1.21	60.48 ± 1.17
GCN + U2B	**74.65** ± 2.49	**76.44** ± 2.61	**74.72** ± 6.89	**74.73** ± 1.87
GraphSAGE	52.95 ± 1.10	59.18 ± 1.17	61.24 ± 0.66	61.63 ± 0.74
GraphSAGE + U2B	**78.43** ± 1.67	**78.46** ± 1.33	**75.19** ± 4.43	**75.82** ± 4.38
Exphormer	64.01 ± 2.18	67.33 ± 2.78	65.16 ± 3.19	65.23 ± 7.31
Exphormer + U2B	**80.21** ± 3.42	**80.28** ± 2.02	**78.48** ± 2.19	**78.61** ± 3.20
G-Mamba	68.46 ± 3.91	72.09 ± 3.16	64.09 ± 2.82	64.63 ± 2.14
G-Mamba + U2B	**86.71** ± 3.04	**86.93** ± 4.15	**82.29** ± 2.47	**82.45** ± 2.54

Table 5: **Scale-imbalance** graph classification performance with other models as backbone.

From Figure 4, all variants perform worse than U2B. The w/o NC variant has the lowest accuracy on PROTEINS datatset, showing that the node-level Converter improves small-scale graph expressiveness by capturing key node patterns. The w/o GC variant also underperforms, highlighting the effectiveness of graph-level Converter in enriching graph representations. Additionally, the constraint regularization terms (w/o Sin and w/o Dyn) are crucial for balancing graphs of different scales.

Parameters Sensitivity Analysis

We evaluate the sensitivity of two key hyperparameters in the U2B framework : ❶ the number of node-level distilled tokens K_v and ❷ the number of graph-level distilled tokens K_g. Specifically, using four datasets under a scale-imbalance scenario, Figure 5 illustrates that the optimal range for K_g and K_v is between 16 and 64. When the number of tokens is too large, the model may struggle to concentrate on extracting representative base features.

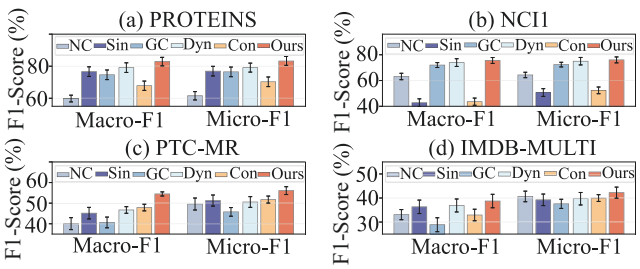

Figure 4: Ablation Study on Scale-imbalance datasets.

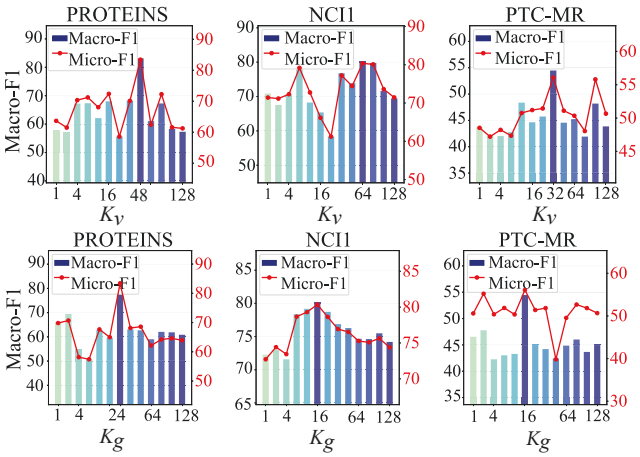

Figure 5: Impact of K_v and K_g on Scale-imbalance datasets.

Figure 6: Representation visualization on NCI1 dataset.

Visualization of Graph Representations

As shown in Figure 6, we further visualized the final graph representations used for classification, using the scale-imbalanced NCI1 dataset as an example to analyze the quality of the representations. The initial GIN representations were disorganized. As we progressively integrate node-level and graph-level converters into GIN, the graph representations become increasingly discriminative. This demonstrates that U2B can effectively balance graph representations across different scales and enhance overall performance.

Conclusion

In this paper, we develop U2B to tackle the challenges posed by comprehensive scale distribution scenarios in graph classification. U2B employs a novel distillation-refinement framework to extract fundamental features at both the node and graph levels. By introducing static constraints at the node-level converter and dynamic constraints at the graph-level converter, U2B reduces bias toward large-scale graphs while maintaining performance on small-scale graphs. Finally, we introduce a consistency alignment loss to further enhance the coherence between node-level and graph-level representations. Extensive experiments demonstrate that U2B achieves optimal accuracy.

Acknowledgements

This paper is partially supported by the National Natural Science Foundation of China (No.12227901) and the Natural Science Foundation of Jiangsu Province (BK20250482). The AI-driven experiments, simulations and model training were performed on the robotic AI-Scientist platform of Chinese Academy of Science.

References

Asano, Y. M.; Rupprecht, C.; and Vedaldi, A. 2019. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*.

Caron, M.; Bojanowski, P.; Joulin, A.; and Douze, M. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, 132–149.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33: 9912–9924.

Fu, X.; Li, J.; Wu, J.; Sun, Q.; Ji, C.; Wang, S.; Tan, J.; Peng, H.; and Yu, P. S. 2021. ACE-HGNN: Adaptive curvature exploration hyperbolic graph neural network. In *2021 IEEE international conference on data mining (ICDM)*, 111–120. IEEE.

Guo, Z.; Sun, Q.; Yuan, H.; Fu, X.; Zhou, M.; Gao, Y.; and Li, J. 2025. GraphMoRE: Mitigating Topological Heterogeneity via Mixture of Riemannian Experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 11754–11762.

Hamilton, W. L. 2020. *Graph representation learning*. Morgan & Claypool Publishers.

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

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Liu, Z.; Mao, Q.; Liu, C.; Fang, Y.; and Sun, J. 2022. On size-oriented long-tailed graph classification of graph neural networks. In *Proceedings of the ACM web conference 2022*, 1506–1516.

Luo, Y.; Shi, L.; and Wu, X.-M. 2025. Can Classic GNNs Be Strong Baselines for Graph-level Tasks? Simple Architectures Meet Excellence. *arXiv preprint arXiv:2502.09263*.

Ma, J.; Cui, Z.; Wang, B.; Wang, P.; Zhou, Z.; Zhao, Z.; and Wang, Y. 2025. Causal Learning Meet Covariates: Empowering Lightweight and Effective Nationwide Air Quality Forecasting. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, 3171–3179.

Noutahi, E.; Beaini, D.; Horwood, J.; Giguère, S.; and Tossou, P. 2019. Towards interpretable sparse graph representation learning with laplacian pooling. *arXiv preprint arXiv:1905.11577*.

Nowozin, S.; Cseke, B.; and Tomioka, R. 2016. f-gan: Training generative neural samplers using variational divergence minimization. *Advances in neural information processing systems*, 29.

Qin, J.; Huang, P.; Sun, Q.; Ji, C.; Fu, X.; and Li, J. 2025. Graph Size-imbalanced Learning with Energy-guided Structural Smoothing. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, 457–465.

Qin, J.; Yuan, H.; Sun, Q.; Xu, L.; Yuan, J.; Huang, P.; Wang, Z.; Fu, X.; Peng, H.; Li, J.; et al. 2024. Igl-bench: Establishing the comprehensive benchmark for imbalanced graph learning. *arXiv preprint arXiv:2406.09870*.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35: 14501–14515.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9).

Shervashidze, N.; Vishwanathan, S.; Petri, T.; Mehlhorn, K.; and Borgwardt, K. 2009. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, 488–495. PMLR.

Shirzad, H.; Velingker, A.; Venkatachalam, B.; Sutherland, D. J.; and Sinop, A. K. 2023. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, 31613–31632. PMLR.

Sun, F.-Y.; Hoffmann, J.; Verma, V.; and Tang, J. 2019. Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. *arXiv preprint arXiv:1908.01000*.

Sun, Q.; Li, J.; Peng, H.; Wu, J.; Fu, X.; Ji, C.; and Yu, P. S. 2022a. Graph structure learning with variational information bottleneck. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, 4165–4174.

Sun, Q.; Li, J.; Yang, B.; Fu, X.; Peng, H.; and Yu, P. S. 2023. Self-organization preserved graph structure learning with principle of relevant information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 4643–4651.

Sun, Q.; Li, J.; Yuan, H.; Fu, X.; Peng, H.; Ji, C.; Li, Q.; and Yu, P. S. 2022b. 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*, 1848–1857.

Sun, Q.; Luo, J.; Yuan, H.; Fu, X.; Peng, H.; Li, J.; and Yu, P. S. 2025. Evolving Graph Learning for Out-of-Distribution Generalization in Non-stationary Environments. *arXiv preprint arXiv:2511.02354*.

Verdier, H.; Duval, M.; Laurent, F.; Cassé, A.; Vestergaard, C. L.; and Masson, J.-B. 2021. Learning physical properties of anomalous random walks using graph neural networks. *Journal of Physics A: Mathematical and Theoretical*, 54(23): 234001.

Wang, B.; Ma, J.; Wang, P.; Wang, X.; Zhang, Y.; Zhou, Z.; and Wang, Y. 2024a. Stone: A spatio-temporal ood learning framework kills both spatial and temporal shifts. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2948–2959.

Wang, B.; Wang, P.; Zhang, Y.; Wang, X.; Zhou, Z.; Bai, L.; and Wang, Y. 2024b. Towards dynamic spatial-temporal graph learning: A decoupled perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 9089–9097.

Wang, B.; Zhang, Y.; Wang, X.; Wang, P.; Zhou, Z.; Bai, L.; and Wang, Y. 2023. Pattern expansion and consolidation on evolving graphs for continual traffic prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2223–2232.

Wang, S.; Ding, Z.; and Xie, X. 2025. SamGoG: A Sampling-Based Graph-of-Graphs Framework for Imbalanced Graph Classification. *arXiv preprint arXiv:2507.13741*.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021a. Curgraph: Curriculum learning for graph classification. In *Proceedings of the web conference 2021*, 1238–1248.

Wang, Y.; Wang, W.; Liang, Y.; Cai, Y.; and Hooi, B. 2021b. Mixup for node and graph classification. In *Proceedings of the Web Conference 2021*, 3663–3674.

Wang, Y.; Xu, Y.; Yang, J.; Wu, M.; Li, X.; Xie, L.; and Chen, Z. 2024c. Graph-aware contrasting for multivariate time-series classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 15725–15734.

Wang, Y.; Zhao, Y.; Shah, N.; and Derr, T. 2022. Imbalanced graph classification via graph-of-graph neural networks. In *Proceedings of the 31st ACM international conference on information & knowledge management*, 2067–2076.

Wu, S.; Sun, F.; Zhang, W.; Xie, X.; and Cui, B. 2022. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5): 1–37.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Xu, W.; Wang, P.; Zhao, Z.; Wang, B.; Wang, X.; and Wang, Y. 2024. When imbalance meets imbalance: Structure-driven learning for imbalanced graph classification. In *Proceedings of the ACM Web Conference 2024*, 905–913.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; and Shen, Y. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33: 5812–5823.

Yun, S.; Jeong, M.; Kim, R.; Kang, J.; and Kim, H. J. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.

Zeng, L.; Li, L.; Gao, Z.; Zhao, P.; and Li, J. 2023. Imgcl: Revisiting graph contrastive learning on imbalanced node classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, 11138–11146.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

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