# Imbalanced Graph Classification via Graph-of-Graph Neural Networks

Yu Wang
yu.wang.1@vanderbilt.edu
Vanderbilt University

Yuying Zhao
yuying.zhao@vanderbilt.edu
Vanderbilt University

Neil Shah
nshah@snap.com
Snap Research

Tyler Derr
tyler.derr@vanderbilt.edu
Vanderbilt University

## ABSTRACT

Graph Neural Networks (GNNs) have achieved unprecedented success in learning graph representations to identify categorical labels of graphs. However, most existing graph classification problems with GNNs follow a balanced data splitting protocol, which is misaligned with many real-world scenarios in which some classes have much fewer labels than others. Directly training GNNs under this imbalanced situation may lead to uninformative representations of graphs in minority classes, and compromise the overall performance of downstream classification, which signifies the importance of developing effective GNNs for handling imbalanced graph classification. Existing methods are either tailored for non-graph structured data or designed specifically for imbalance node classification while few focus on imbalance graph classification. To this end, we introduce a novel framework, Graph-of-Graph Neural Networks ($G^2$GNN), which alleviates the graph imbalance issue by deriving extra supervision globally from neighboring graphs and locally from graphs themselves. Globally, we construct a graph of graphs (GoG) based on kernel similarity and perform GoG propagation to aggregate neighboring graph representations, which are initially obtained by node-level propagation with pooling via a GNN encoder. Locally, we employ topological augmentation via masking nodes or dropping edges to improve the model generalizability in discerning topology of unseen testing graphs. Extensive graph classification experiments conducted on seven benchmark datasets demonstrate our proposed $G^2$GNN outperforms numerous baselines by roughly 5% in both F1-macro and F1-micro scores.

**ACM Reference Format:**
Yu Wang, Yuying Zhao, Neil Shah, and Tyler Derr. 2018. Imbalanced Graph Classification via Graph-of-Graph Neural Networks. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/1122445.1122456


*Woodstock '18, June 03–05, 2018, Woodstock, NY*

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

## 1 INTRODUCTION

Employing graph representations for classification has recently attracted significant attention due to the emergence of Graph Neural Networks (GNNs) associated with its unprecedented power in expressing informative graph representations [34]. A typical GNN architecture for graph classification begins with a graph encoder that extracts node representations by iteratively propagating neighborhood information followed by pooling operations that integrate node representations into graph representations, which are then fed into a classifier to predict graph labels [7]. Although numerous GNN variants have been proposed by configuring different propagation and pooling schemes, most works are framed under a balanced data-split setting where an equal number of labeled graphs are provided as the training data for all classes [25]. However, collecting such high-quality balanced data tends to be time-intensive and resource-expensive, and thus often impossible in reality [17].

In many real-world graph datasets, the distribution of graphs across classes varies from a slight bias to a severe imbalance where a large portion of classes (minority classes) contain a limited number of labeled graphs while few classes (majority classes) contain enough labeled graphs [5]. For example, despite the huge chemical space, few compounds are labeled active with the potential to interact with a target biomacromolecule; the remaining majority are labeled inactive [13]. Since most GNNs are framed on balanced datasets, directly employing them on imbalanced datasets would lead to sub-optimal representations of graphs in minority classes, and hence lower the overall classification performance. As one sub-branch of deep learning on graph-structured data, GNNs similarly inherit two severe problems of traditional deep learning on imbalanced datasets: inclination to learning towards majority classes [14] and poor generalization from given scarce training data to abounding unseen testing data [25, 38]. Aimed at these two challenges, traditional methods for handling class imbalance include augmenting data via under- or over-sampling [3, 30], assigning weights to adjust the portion of training loss of different classes [28], and constructing synthetic training data via interpolation over minority instances to extend the training distribution [2]. However, these methods have been primarily performed on simple point-based data and their performance on graph-structured data is unclear/unexplored.

Imbalance on graph-structured data could lie either in the node or graph domain where nodes (graphs) in different classes have different amount of training data. Nearly all existing related GNN works focus on imbalanced node classification by either pre-training or

adversarial training to reconstruct the graph topology [2, 22, 23, 33, 39], while to the best of our knowledge, imbalanced graph classification remains largely unexplored. On one hand, unlike node classification where we can derive extra supervision for minority nodes from their neighborhoods via propagation, graphs are individual instances that are isolated from each other and thus we cannot aggregate information directly from other graphs by propagation. On the other hand, compared with imbalance on regular grid or sequence data (e.g., images or text) where imbalance mainly lies in feature or semantic domain, the imbalance of graph-structured data could also be attributed to the graph topology since unrepresentative topology provided by limited training instances ill-defines minority classes.

To address the aforementioned challenges, we present Graph-of-Graph Neural Networks (G$^2$GNN). The proposed framework consists of two essential components that seamlessly work together to derive extra supervision globally from neighboring graphs and locally from graphs themselves. Globally, a graph kernel-based GoG construction is proposed to establish a $k$-nearest neighbor (kNN) graph and hence enable two-level propagation, where graph representations are first obtained by pooling after node-level propagation via a GNN encoder and then neighboring graph representations are aggregated together through the GoG propagation on the established kNN graph. Locally, we employ topological augmentation via masking nodes or removing edges with self-consistency regularization to create novel supervision from each individual graph. The GoG propagation serves as a global governance to retain the model discriminability by smoothing intra-class graphs while separating inter-class graphs. Meanwhile the topological augmentation functions as a local explorer to enhance the model generalizability in discerning unseen topology of testing graphs. In summary, the main contributions of our work are as follows:

- **Problem:** We investigate a novel problem of imbalanced graph classification with GNNs. In particular, we emphasize its importance in real-world applications and further provide a formal problem definition.
- **Algorithm:** We propose a novel framework G$^2$GNN for imbalanced graph classification that derives extra supervision by globally aggregating information of neighboring graphs and locally augmenting graph topology.
- **Experiments:** We perform extensive experiments on various real-world datasets were the experimental results show the effectiveness of our proposed framework G$^2$GNN for imbalanced graph classification.

## 2 PROBLEM FORMULATION

Let $G = (\mathcal{V}^G, \mathcal{E}^G, \mathbf{X}^G, \mathbf{A}^G)$ denote an attributed graph with node feature vectors $\mathbf{X}^G \in \mathbb{R}^{|\mathcal{V}^G| \times d}$ and adjacency matrix $\mathbf{A}^G \in \mathbb{R}^{|\mathcal{V}^G| \times |\mathcal{V}^G|}$ where $\mathbf{A}^G_{ij} = 1$ if there is an edge between nodes $v_i, v_j$ and vice versa. In graph classification, given a set of $N$ graphs $\mathcal{G} = \{G_1, G_2, ..., G_N\}$ with each graph $G_i = (\mathcal{V}^{G_i}, \mathcal{E}^{G_i}, \mathbf{X}^{G_i}, \mathbf{A}^{G_i})$ as defined above and their labels $\mathbf{Y} \in \mathbb{R}^{N \times C}$ where $C$ is the total number of classes, we aim to learn graph representations $\mathbf{P} \in \mathbb{R}^{N \times d'}$ with $\mathbf{P}_i$ for each $G_i \in \mathcal{G}$ that is well-predictive of its one-hot encoded label $\mathbf{Y}_i$. Based on the previously defined notations, the imbalanced graph classification can be mathematically formalized as:

*Given a set of attributed graphs $\mathcal{G}$ with labels for a subset of $l$ graphs $\mathcal{G}^l$ that are imbalanced among different classes, we aim to learn a graph classifier $\mathcal{F} : \mathcal{F}(\mathbf{X}^{G_i}, \mathbf{A}^{G_i}) \rightarrow \mathbf{Y}_i$ that works well for graphs in both majority and minority classes.*

## 3 RELATED WORK

In this section, we present related work in the areas of graph imbalance problem, graph of graphs, and data augmentations.

### 3.1 Graph Imbalance Problem

Graph imbalance exists in many real-world scenarios [39]. Apart from the classical methods handling the general class imbalance problem [12, 16], graph topology can naturally be harnessed to derive extra supervision for learning representations. DR-GCN [23] handles multi-class imbalance by class-conditional adversarial training and latent distribution regularization. RECT [33] merges a GNN and proximity-based embeddings for the completely-imbalanced setting (i.e., some classes have no labeled nodes). GraphSMOTE [39] attempts to generate edges by pre-training an edge generator for isolated synthetic nodes from SMOTE [2]. Most recently, imGAGN [22] simulates both distributions of node attributes in minority classes and graph structures via generative adversarial graph network model. However, all of these recent works are proposed for node imbalance classification and to the best of our knowledge, graph imbalance classification with GNNs remains largely unexplored.

### 3.2 Graph of Graphs

Graphs model entities by nodes and their relationships by edges. Sometimes, the entities (i.e., nodes) at a higher level in a graph can themselves be modeled as graphs at a lower level, which is termed as graph of graphs (GoG) or network of networks (NoN) [20]. This hierarchical relationship encoded by GoG was initially used in [20] to compare nodes in a broader context and rank them at a finer granularity. Recently, [32] and [11] customize models to either predict missing links/interactions between graphs or classify unlabeled graphs on GoG-structured data, which is assumed as the provided input. Conversely, in this work, we construct a kNN GoG and apply it to derive extra information from neighboring graphs with similar topology for imbalanced graph classification. To the best our knowledge, this is the first work in graph classification constructing GoG from basic graph datasets and leveraging it for propagation.

### 3.3 Data Augmentation

Augmentation aims to expand training data via artificially creating more reasonable virtual data from existing limited data [19]. Training model with augmented data would increase the generalizability of the model such that the performance on unseen testing data could also be improved. Recent years have witnessed successful applications of data augmentation in computer vision (CV) [24] and natural language processing (NLP) [8]. As its derivative in graph domain, graph augmentation also enriches the training graph data and therefore can be effectively leveraged to alleviate the class imbalance problem in graph classification. In this work, we conduct the augmentation via heuristic modification of graph topology, which includes removing edges and masking nodes [36].

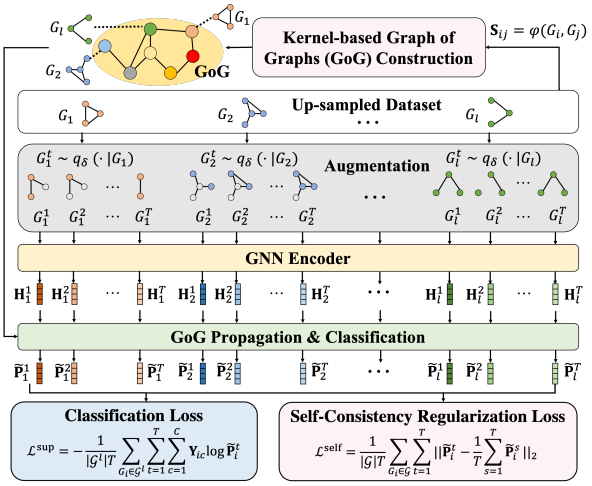


**Figure 1: An overview of the proposed Graph-of-Graph Neural Network (G$^2$GNN) for imbalanced graph classification. Here we up-sample minority graphs to reduce imbalance effect, augment graphs $T$ times followed by a GNN encoder to get their representations, aggregate neighboring graph representations by propagation on constructed GoG, and finally use their obtained logits for classification and the self-consistency regularization.**

# 4 THE PROPOSED FRAMEWORK

In this section, we introduce our proposed G$^2$GNN framework. Figure 1 presents an overview of G$^2$GNN that is composed of four modules. Firstly, each graph is augmented to generate various graphs, which are further fed into a graph encoder to get their representations. Then we construct a GoG based on kernel similarity and perform GoG propagation to aggregate neighboring graph information, which are then forwarded through a classifier to compute the classification loss and the self-consistency regularization loss. Next, we will introduce details of each module.

## 4.1 Initial GNN Encoder

Our graph classifier begins with a graph encoder that utilizes graph topology and node features to learn graph representation. In this work, we employ a well-known GNN variant, graph isomorphism network (GIN), as the encoder to learn the representation for graph $G_i$ due to its distinguished discriminative power of different topology [34]. However, our framework holds straightforwardly for any other GNN-based encoder. One GIN layer is defined as:

$$\mathbf{X}^{G_i,k+1} = \text{MLP}^k((\mathbf{A}^{G_i} + (1+\epsilon)\mathbf{I})\mathbf{X}^{G_i,k}), \tag{1}$$

where $\mathbf{X}^{G,k}$ is the intermediate node representation at layer $k$, $\mathbf{X}^{G_i,0} = \mathbf{X}^{G_i}$ is the original feature matrix of the graph $G_i$, and MLP is a multi-layer perceptron at layer $k$. After $K$ iterations of GIN convolution, each node aggregates information from its neighborhoods up to $K$ hops away and a graph-level readout function is applied to integrate node representations into the overall graph representation $\mathbf{H}_i$ for graph $G_i$ as:

$$\mathbf{H}_i = \text{READOUT}(\{\mathbf{X}_j^{G_i,K} | v_j \in \mathcal{V}^{G_i}\}) \tag{2}$$

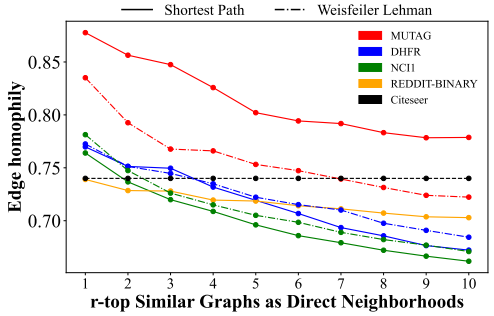


**Figure 2: Edge homophily of constructed kNN GoGs.**

Instead of solely depending on semantic representation $\mathbf{H}_i$ to classify graphs that only provides limited supervision for minority graphs, we construct a kNN graph to connect these independent graphs and perform GoG propagation to derive extra supervision from neighboring graphs. We begin with constructing GoG and empirically demonstrate its high homophily, which naturally motivates the GoG propagation.

## 4.2 Graph of Graphs Construction

Given a set of graphs $\mathcal{G}$, we expect to construct a high-level graph where every graph $G_i \in \mathcal{G}$ is represented by a node and two similar graphs are linked by an edge. In this work, we mainly consider the topological similarity since graphs with similar topology typically possess similar functions [40]. We leverage the graph kernel to compute the similarity matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$. Specifically, each entry $\mathbf{S}_{ij}$ measures the topological similarity between two graph instances $G_i$ and $G_j$ computed by $\phi$ as:

$$\mathbf{S}_{i,j} = \phi(G_i, G_j), \tag{3}$$

where multiple choices of kernel functions $\phi$ could be adopted here depending on specific types of topological similarity that downstream tasks require and further details can be referred in Appendix B. Then we construct a kNN graph $\mathcal{G}^{\text{kNN}}$ by connecting each graph $G_i$ with their top-$r$ similar graphs based on the similarity matrix $\mathbf{S}$ and then measure its edge homophily as:

$$\chi^{\mathcal{G}^{\text{kNN}}} = \frac{|\{(G_i, G_j) \in \mathcal{E}^{\mathcal{G}^{\text{kNN}}} : \mathbf{Y}_i = \mathbf{Y}_j\}|}{|\mathcal{E}^{\mathcal{G}^{\text{kNN}}}|}, \tag{4}$$

where high $\chi^{\mathcal{G}^{\text{kNN}}}$ means most edges connect graphs of the same class and by varying $r$, we end up with multiple $\mathcal{G}^{\text{kNN}}$ with different homophily level. Figure 2 visualizes the homophily of $\mathcal{G}^{\text{kNN}}$ constructed using Shortest-Path and Weisfeiler-Lehman kernels on four graphs populating in the literature. We can clearly see that edge homophily decreases as $r$ increases because graphs with low topological similarity have higher chance to be selected as neighborhoods while they likely belong to different classes from corresponding center graphs. However, edge homophily even with $r$ up to 5 is still in $[0.7, 0.8]$ and comparable to Citeseer dataset[1] indicating most edges in the constructed $\mathcal{G}^{\text{kNN}}$ connects graphs of the same class. Motivated by this observation, we perform GoG propagation on the generated kNN graph $\mathcal{G}^{\text{kNN}}$ to aggregate neighboring graph information.

[1] Citeseer is a well-known node classification dataset and commonly used for benchmarking GNNs [25]

### 4.3 Graph of Graphs Propagation

Denoting the adjacency matrix with added self-loops of the constructed graph $\mathcal{G}^{\text{kNN}}$ as $\hat{\mathbf{A}}^{\text{kNN}} = \mathbf{A}^{\text{kNN}} + \mathbf{I}$ and the corresponding degree matrix as $\hat{\mathbf{D}}^{\text{kNN}}$, the $k^{\text{th}}$-layer GoG propagation is formulated as:

$$\mathbf{P}^{k+1} = (\hat{\mathbf{D}}^{\text{kNN}})^{-1}\hat{\mathbf{A}}^{\text{kNN}}\mathbf{P}^{k}, \tag{5}$$

where $\mathbf{P}^0 = \mathbf{H}$ includes representations of all individual graphs $\mathbf{H}_i$ that are previously obtained from GIN followed by the graph pooling, Eqs. (1)-(2). After $K$ layers propagation, the representation of a specific graph $\mathbf{P}_i^K$ aggregates information from neighboring graphs up to $K$ hops away. The intuition behind the GoG propagation is smoothing among neighboring graphs since the representation of every graph $\mathbf{P}_i^{k+1}$ is the weighted average of its neighboring graphs' representations at previous layer $k$:

$$\mathbf{P}_i^{k+1} = \frac{1}{\hat{\mathbf{d}}_i}\sum_{G_j \in \mathcal{N}_i} \mathbf{P}_j^k, \tag{6}$$

where $\mathcal{N}_i$ is the set of neighboring graphs of $G_i$ and $\hat{\mathbf{d}}_i = \hat{\mathbf{D}}_{ii}^{\text{kNN}}$ is the degree of graph $G_i$ in the constructed $\mathcal{G}^{\text{kNN}}$. Feature smoothing naturally connects with label smoothing by the following theorem:

Theorem 4.1. *Suppose that the latent ground-truth mapping $\mathcal{M}: \mathbf{P}_i^k \rightarrow \mathbf{Y}_i$ from graph representations to graph labels is differentiable and satisfies $\mu$−Lipschitz constraints, i.e., $|\mathcal{M}(\mathbf{P}_i^k) - \mathcal{M}(\mathbf{P}_j^k)| \leq \mu||\mathbf{P}_i^k - \mathbf{P}_j^k||_2$ for any pair of graphs $G_i, G_j$ ($\mu$ is a constant), then the label smoothing is upper bounded by the feature smoothing among graph $G_i$ and its neighboring graphs $\mathcal{N}_i$ through (7) with an error $\boldsymbol{\epsilon}_i^k = \mathbf{P}_i^{k+1} - \mathbf{P}_i^k$:*

$$\underbrace{(\hat{\mathbf{d}}_i^{-1}\sum_{G_j \in \mathcal{N}_i} \mathbf{Y}_j - \mathbf{Y}_i)}_{\text{Label smoothing}} - \underbrace{(\hat{\mathbf{d}}_i^{-1}\sum_{G_j \in \mathcal{N}_i} o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2))}_{\text{feature smoothing}} \leq \mu\boldsymbol{\epsilon}_i^k. \tag{7}$$

Specifically $\boldsymbol{\epsilon}_i^k$ quantifies the difference of the graph $G_i$'s representation between $k^{\text{th}}$ and $(k+1)^{\text{th}}$ iteration, which decreases as propagation proceeds [18] and eventually converges after infinite propagation $\lim_{k\to\infty} \boldsymbol{\epsilon}_i^k = 0$ as Theorem 4.2 shows:

Theorem 4.2. *Given a graph $\mathcal{G}$ with $m$ connected components $\{C_1, C_2, ..., C_m\}$ and their corresponding row-normalized adjacency matrices $\{\tilde{\mathbf{A}}^{C_1}, \tilde{\mathbf{A}}^{C_2}, ..., \tilde{\mathbf{A}}^{C_m}\}$ where $\tilde{\mathbf{A}}^{C_i} = (\hat{\mathbf{D}}^{C_i})^{-1}\hat{\mathbf{A}}^{C_i} \in \mathbb{R}^{\mathcal{V}^{C_i}\times\mathcal{V}^{C_i}}$, then for each component $C_i$, $\lim_{k\to\infty}(\tilde{\mathbf{A}}^{C_i})^k\mathbf{P}^{C_i} = \boldsymbol{\Pi}^{C_i}\mathbf{P}^{C_i}$, where $\boldsymbol{\Pi}_j^{C_i} = \boldsymbol{\pi}^{C_i} \in \mathbb{R}^n$ is the unique left eigenvector of $\tilde{\mathbf{A}}^{C_i}$.*

Proof of Theorem 4.1 and Theorem 4.2 are provided in the Appendix A. Theorem 4.2 reveals that after infinite propagation, representations of all nodes in component $C_i$ converges to a stationary point $\boldsymbol{\pi}^{C_i}\mathbf{P}^{C_i}$. Applying Theorem 4.2 on the $\mathcal{G}^{\text{kNN}}$, each graph $G_i$ is treated as a node and its representation $\mathbf{P}_i^k$ gradually converges since $\lim_{k\to\infty}\epsilon_i^k = 0$. Such feature smoothing further leads to the label smoothing based on Theorem 4.1. Therefore propagating features according to (6) is equivalent to propagating labels among neighboring graphs, which derives extra supervision for imbalance classification. Given the high-homophily of the $\mathcal{G}^{\text{kNN}}$ in Figure 2, i.e, neighboring graphs tend to share the same class, the extra supervision derived from feature propagation (label propagation) is very likely beneficial for downstream classification.

### 4.4 Graph Augmentation with Self-consistency

Even though feature propagation globally derives extra label information for minority classes, training with limited graph instances still restricts the power of the model in recognizing unseen topology for testing graphs. To retain the model generalizability to unseen testing graphs, we further apply topological augmentation to enrich the graph topology via heuristic modification of existing graphs' topology. Specifically, we leverage two types of augmenting schemes, removing edges and masking nodes, which are formulated along with the self-consistency regularization respectively as:

*4.4.1 Removing Edges:* For each graph $G_i \in \mathcal{G}$, we randomly remove a subset of edges $\widehat{\mathcal{E}}^{G_i}$ from the original edge set $\mathcal{E}^{G_i}$ with probability: $P(e_{uv} \in \widehat{\mathcal{E}}^{G_i}) = 1 - \delta_{uv}^{G_i}$, where $\delta_{uv}^{G_i}$ could be uniform or adaptive for different edges. Given uniformly removing edges, (i.e., $\delta_{uv}^{G_i} = \delta$) already enjoys a boost over baselines as shown in Section 5, we leave the adaptive one as future work.

*4.4.2 Masking Nodes:* Instead of directly masking nodes that may disconnect the original graph into several components, we retain the graph structure by simply masking entire features of some nodes [9]. Formally, we randomly sample a binary mask $\eta_j^{G_i} \sim \text{Bernoulli}(1 - \delta_j^{G_i})$ for each node $v_j$ in graph $G_i$, and then multiply that node's feature vector with its corresponding mask, i.e., $\widehat{\mathbf{X}}_j^{G_i} = \eta_j^{G_i}\mathbf{X}_j^{G_i}$ [38]. Same as removing edges, we only consider uniformly masking nodes in this work. Collectively, we unify the probability of removing edges $\delta_{uv}^{G_i}$ and the ratio of masking nodes $\delta_j^{G_i}$ as augmentation ratio $\delta$ for model simplicity.

*4.4.3 Self-Consistency Regularization.* Arbitrary modification of graph topology without any regularization could unintentionally introduce invalid or even abnormal topology that corrupt learned representations. Therefore, we leverage self-consistency regularization to supervise augmented graphs by themselves. Formally, given a set of augmented graphs $\widehat{\mathcal{G}}_i = \{G_i^1, G_i^2, ..., G_i^T | G_i^t \sim q_\delta(\cdot|G_i)\}$ where $q_\delta(\cdot|G_i)$ is the augmentation distribution conditioned on the original graph $G_i$ parameterized by the augmentation ratio $\delta$, we feed them through a graph encoder by Eq. (1)-(2) and the GoG propagation by Eq. (5) to obtain their representations $\{\mathbf{P}_i^1, \mathbf{P}_i^2, ..., \mathbf{P}_i^T\}$. Then the self-consistency regularization loss for the graph $G_i$ is formulated as the average $L_2$ distance between the class distribution of each augmented graph $\mathbf{P}_i^t$ and their average class distribution:

$$\mathcal{L}_i^{\text{self}} = \frac{1}{T}\sum_{t=1}^{T}||\widetilde{\mathbf{P}}_i^t - \frac{1}{T}\sum_{s=1}^{T}\widetilde{\mathbf{P}}_i^s||_2, \tag{8}$$

where $\widetilde{\mathbf{P}}_i^t = \sigma(g_{\boldsymbol{\theta}_g}(\mathbf{P}_i^t))$ is the class distribution of each augmented graph obtained by softmax normalization on the final prediction $g_{\boldsymbol{\theta}_g}(\mathbf{P}_i^t)$. $g$ is a trainable classifier parametrized by $\boldsymbol{\theta}_g$.

### 4.5 Objective Function

The overall objective function of G$^2$GNN can be formally defined as follows:

$$\mathcal{L} = \mathcal{L}^{\text{sup}} + \mathcal{L}^{\text{self}} = -\frac{1}{|\mathcal{G}^l|T} \sum_{G_i \in \mathcal{G}^l} \sum_{t=1}^{T} \sum_{c=1}^{C} \mathbf{Y}_{ic} \log \widetilde{\mathbf{P}}_i^t + \frac{1}{|\mathcal{G}|T} \sum_{G_i \in \mathcal{G}} \sum_{t=1}^{T} ||\widetilde{\mathbf{P}}_i^t - \frac{1}{T} \sum_{s=1}^{T} \widetilde{\mathbf{P}}_i^s||_2, \quad (9)$$

where $\mathcal{L}^{\text{sup}}$ is the cross entropy loss over all training graphs in $\mathcal{G}^l$ with known label information as previously defined with $C$ graph classes to be predicted, and $\mathcal{L}^{\text{self}}$ is the self-consistency regularization loss defined by Eq. (8) over all training graphs.

## 4.6 Algorithm

In Algorithm 1, we present a holistic overview of the key stages in the proposed G$^2$GNN framework. We furthermore present a detailed complexity analysis in Appendix C.

**Algorithm 1:** The algorithm of G$^2$GNN

**Input:** an imbalanced set of labeled graphs $\mathcal{G}^l$, a kernel function $\phi$, a graph encoder $f_{\theta_f}$, a classifier $g_{\theta_g}$, the augmentation distribution $q_\delta$

**Output:** $f_{\theta_f}$ and $g_{\theta_g}$ with learned parameters $\theta_f, \theta_g$

```
Initialize the model parameters θ_f, θ_g
Compute pairwise similarity matrix S by Eq. (3) and construct G^kNN following Section 4.2
Up-sample minority labeled graphs in G^l
while not converged do
    for mini-batch of graphs G^B = {G_i | G_i ∈ G^l} do
        Find top-r similar graphs for each G_i ∈ G^B based on S and incorporate them into G^B
        Obtain the subgraph G^{kNN,B} from G^kNN induced by graphs in G^B
        For each G_i ∈ G^B generate T augmented graphs Ĝ_i = {G_i^t | G_i^t ~ q_δ(·|G_i)}
        Apply graph encoder f_{θ_f} by Eqs. (1)-(2), the GoG propagation by Eq. (5), and the classifier g_{θ_g} to predict graph class distribution {P̃_i^t | G_i ∈ G^l, t ∈ T}
        Update θ_f, θ_g by minimizing Eq. (9)
```

# 5 EVALUATION

In this section, we evaluate the effectiveness of G$^2$GNN by conducting extensive imbalanced graph classification on multiple types of graphs with different levels of imbalance. We begin by introducing the experimental setup, including datasets, baselines, evaluation metrics, and parameter settings.

## 5.1 Experimental Setup

We conduct experiments on seven real-world datasets [27, 35]. Specifically, this includes the following graph classification datasets: (1) Chemical compounds: PTC-MR, NCI1, and MUTAG. (2) Protein compounds: PROTEINS, D&D, and DHFR. (3) Social network: REDDIT-B. We note that details on these datasets can be found in Table 3.

*5.1.1 Baselines.* To evaluate the effectiveness of the proposed G$^2$GNN, we select three models designed for graph classification, which includes: (1) **GIN** [34]: A basic supervised GNN model for graph classification due to its distinguished expressiveness of graph topology. (2) **InfoGraph** [26]: An unsupervised GNN model for learning graph representations via maximizing the mutual information between the whole graph and its substructures of different scales. (3) **GraphCL** [36]: Stepping further from InfoGraph, GraphCL proposes four strategies to augment graphs and learns graph representations by maximizing the mutual information between the original graph and its augmented variants.

Since imbalance datasets naturally provided weak supervision, unsupervised GNNs outweight supervised counterparts and selecting them as baselines could more confidently justify the superiority of our model. Equipping the above three backbones with up-sampling, re-weight, and SMOTE strategies tailored specifically for imbalance classification, we end up with 10 baselines. Specifically, we equip up-sampling and re-weight with all three backbones. Since applying SMOTE leads to similar or even worse performance gains, we only stack it on the GIN backbone.

*5.1.2 Evaluation Metrics.* Following existing work in imbalanced classification [39], we use two criterion: F1-macro and F1-micro to measure the performance of our proposed G$^2$GNN and baselines. F1-macro computes the accuracy independently for each class and then takes the average (i.e., treating classes equally). F1-micro computes accuracy over all testing examples at once, which may underweight the minority classes.

*5.1.3 Parameter Settings.* We implement our proposed G$^2$GNN and some necessary baselines using Pytorch Geometric [10]. For InfoGraph[2] and GraphCL[3] we use the original authors' code with any necessary modifications. Aiming to provide a rigorous and fair comparison across models on each dataset, we tune hyperparameters for all models individually as: the weight decay $\in [0, 0.1]$, the encoder hidden units $\in \{128, 256\}$, the learning rate $\in \{0.001, 0.01\}$, the inter-network level propagation $K \in \{1, 2, 3\}$, the augmentation ratio $\delta \in \{0.05, 0.1, 0.2\}$, and $r \in \{2, 3, 4\}$. We employ Shortest Path Kernel to compute similarity matrix S and set the trainable classifier $g$ as a 2-layer MLP. For reproducibility, model code and corresponding hyperparameter configurations are publicly available[4].

## 5.2 Performance Comparison

In this subsection, we compare the performance of **G$^2$GNN$_e$** and **G$^2$GNN$_n$**, which represent the the G$^2$GNN framework using the edge removal or node masking augmentations, respectively, against the aforementioned baselines. Since class distributions of most datasets are not strictly imbalanced, we use an imitative imbalanced setting: we randomly set 25%/25% graphs as training/validation and varying the training instances in each class till the imbalance ratio reaches 1:9 (with details in Appendix D.1). Table 1 reports the average performance per metric across 50 different data splits.

We observe from Table 1 that G$^2$GNN performs the best in all 7 datasets under both F1-macro and F1-micro. Moreover, edge removing augmentation (i.e., G$^2$GNN$_e$) benefits more on the social network (i.e., REDDIT-B) while node masking (i.e., G$^2$GNN$_n$) enhances more on biochemical molecules (e.g., MUTAG, D&D, NCI1 and PTC-MR), which conforms to [37] and may be partially attributed to no node attributes presented in the social network. Models that are specifically designed for tackling the class imbalance issue generally perform better than the corresponding bare backbones. The inferior performance of GIN$_{rw(st)}$ to GIN$_{us}$ is because we either set weights for adjusting training loss of different classes or generate synthetic samples based on training data

[2] https://github.com/fanyun-sun/InfoGraph
[3] https://github.com/Shen-Lab/GraphCL
[4] Code for G$^2$GNN: https://github.com/YuWVandy/G2GNN

**Table 1: Imbalanced graph classification results (best performance in bold and second underlined).**

| Model | MUTAG (5:45) | | PROTEINS (30:270) | | D&D (30:270) | | NCI1 (100:900) | |
|---|---|---|---|---|---|---|---|---|
| | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro |
| GIN | 52.50 | 56.77 | 25.33 | 28.50 | 9.99 | 11.88 | 18.24 | 18.94 |
| $GIN_{us}$ | 78.03 | 78.77 | 65.64 | 71.55 | 41.15 | 70.56 | 59.19 | 71.80 |
| $GIN_{rw}$ | 77.00 | 77.68 | 54.54 | 55.77 | 28.49 | 40.79 | 36.84 | 39.19 |
| $GIN_{st}$ | 74.61 | 75.11 | 56.07 | 57.85 | 27.08 | 39.01 | 40.40 | 44.48 |
| InfoGraph | 69.11 | 69.68 | 35.91 | 36.81 | 21.41 | 27.68 | 33.09 | 34.03 |
| $InfoGraph_{us}$ | 78.62 | 79.09 | 62.68 | 66.02 | 41.55 | 71.34 | 53.38 | 62.20 |
| $InfoGraph_{rw}$ | 80.85 | 81.68 | 65.73 | 69.60 | 41.92 | 72.43 | 53.05 | 62.45 |
| GraphCL | 66.82 | 67.77 | 40.86 | 41.24 | 21.02 | 26.80 | 31.02 | 31.62 |
| $GraphCL_{us}$ | 80.06 | 80.45 | 64.21 | 65.76 | 38.96 | 64.23 | 49.92 | 58.29 |
| $GraphCL_{rw}$ | 80.20 | 80.84 | 63.46 | 64.97 | 40.29 | 67.96 | 50.05 | 58.18 |
| $G^2GNN_e$ | 80.37 | 81.25 | **67.70** | 73.10 | 43.25 | 77.03 | 63.60 | 72.97 |
| $G^2GNN_n$ | **83.01** | **83.59** | 67.39 | **73.30** | **43.93** | **79.03** | **64.78** | **74.91** |

| Model | PTC-MR (9:81) | | DHFR (12:108) | | REDDIT-B (50:450) | | Avg. Rank | |
|---|---|---|---|---|---|---|---|---|
| | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro |
| GIN | 17.74 | 20.30 | 35.96 | 49.46 | 33.19 | 36.02 | 12.00 | 12.00 |
| $GIN_{us}$ | 44.78 | 55.43 | 55.96 | 59.39 | 66.71 | 83.00 | 5.00 | 4.43 |
| $GIN_{rw}$ | 36.96 | 43.09 | 55.16 | 57.78 | 45.17 | 51.92 | 8.86 | 8.86 |
| $GIN_{st}$ | 36.30 | 40.04 | 56.06 | 58.48 | 60.05 | 73.59 | 8.29 | 8.43 |
| InfoGraph | 25.85 | 26.71 | 50.62 | 56.28 | 57.67 | 67.10 | 10.00 | 10.14 |
| $InfoGraph_{us}$ | 44.29 | 48.91 | 59.49 | 61.62 | 67.01 | 78.68 | 5.00 | 5.00 |
| $InfoGraph_{rw}$ | 44.09 | 49.17 | 58.67 | 60.24 | 65.79 | 77.35 | 4.43 | 4.29 |
| GraphCL | 24.22 | 25.16 | 50.55 | 56.31 | 53.40 | 62.19 | 10.71 | 10.57 |
| $GraphCL_{us}$ | 45.12 | 53.50 | 60.29 | 61.71 | 62.01 | 75.84 | 5.29 | 5.43 |
| $GraphCL_{rw}$ | 44.75 | 52.22 | 60.87 | 61.93 | 62.79 | 76.15 | 5.00 | 5.29 |
| $G^2GNN_e$ | 46.40 | 56.61 | **61.63** | **63.61** | **68.39** | **86.35** | **1.71** | 1.86 |
| $G^2GNN_d$ | **46.61** | **56.70** | 59.72 | 61.27 | 67.52 | 85.43 | **1.71** | **1.71** |

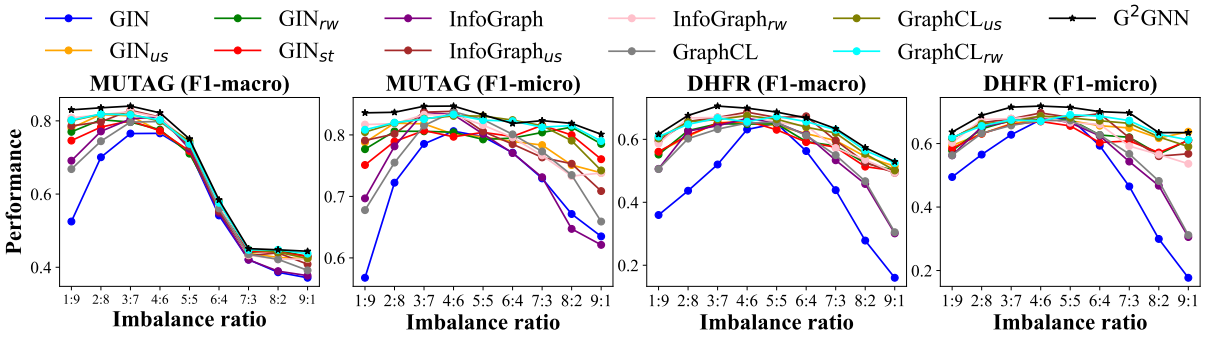


**Figure 3: Graph classification results under different class imbalance ratios where 5:5 corresponds to balanced scenario while 1:9 and 9:1 correspond to highly imbalance scenario. In both F1-macro and F1-micro scores our $G^2GNN$ model outperforms all baselines in nearly all the imbalance ratio settings. Furthermore, the margin increases as the level of imbalance increases (i.e., deviates from the balanced scenario). Note that here we use the same amount of training and validation graphs (25%/25%) as used in Table 1.**

at current batch. Since the number of training instances in each batch may not strictly follow the prescribed imbalance ratio, the batch-dependent weight or synthetic samples hardly guarantee balance globally. InfoGraph(GraphCL)-based variants do not suffer from the issue introduced by batch-training since once we obtain graph representations from pre-trained models by mutual information maximization, we feed them through downstream classifiers all at once without any involvement of batch process. Therefore, the performance of $InfoGraph(GraphCL)_{rw(st)}$ is comparable to $InfoGraph(GraphCL)_{us}$.

## 5.3 Influence of Imbalance Ratio

Due to space limitation, we omit details of this experimental setting but leave it in Appendix D.2. In Figure 3, we can clearly see that the performance of all models first rises and then declines as the imbalance ratio increases from 0.1 to 0.9, which demonstrates the detrimental effect of data imbalance on the model performance and such detrimental effect becomes even worse when the imbalance becomes more severe. Furthermore, the F1-macro score of our $G^2GNN$ model clearly outperforms all other baselines on both MUTAG and DHFR under each imbalance ratio, which soundly justifies the superority and robustness of our model in alleviating imbalance of different level. Different from supervision presented directly from given labeled data, the extra supervision derived by leveraging neighboring graphs' information via propagation and topological augmentation is weakly controlled by the amount of training data. Therefore, the margin achieved by our model grows when imbalance ratio is either too low or too high since the extra supervision stays the same while the basic supervision encoded in the training data decreases. Besides, our model also performs comparable or even slightly better then all other baselines under balanced scenario, which additionally signifies the potentiality of

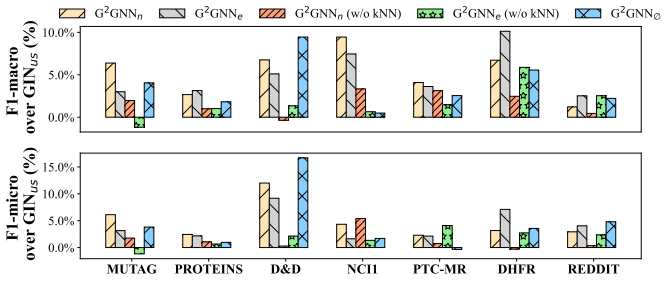


**Figure 4: Ablation study results of G$^2$GNN where we report the improvement over GIN$_{us}$ due to its simplicity and effectiveness (seen in Table 1) for understanding relative improvements of each G$^2$GNN component.**

our model in traditional balance situation. Among other baselines, GraphCL$_{rw}$ performs the best since it applies re-weight strategy to balance the training loss and further leverages the graph augmentation coupled with mutual information maximization to extract the most relevant information for downstream classification. An interesting observation is that the optimal performance is not always achieved when the labeled data is strictly balanced, which reflects the uneven distribution of informatic supervision embedded in data across different classes.

## 5.4 Ablation Study

In this section, we conduct ablation study to fully understand the effect of each component in G$^2$GNN on alleviating the imbalance issue. In Figure 4, we present performance improvement over the baseline GIN$_{us}$ achieved by our proposed framework (**G$^2$GNN$_{e(n)}$**) along with variants that remove the GoG propagation (**G$^2$GNN$_{e(n)}$ (w/o kNN)**) and remove the topological augmentation (**G$^2$GNN$_\emptyset$**). First, we notice that solely employing GoG propagation increases the performance on almost all datasets except PTC-MR on F1-micro score. This is because PTC-MR dataset has lower homophily shown in Figure 7 and therefore neighboring graphs tend to share different labels, propagating information of which compromises the performance of downstream classification. Second, augmenting via removing edges hurts the performance on MUTAG. This is because the size of graphs in MUTAG are relatively small and thus removing edges may undermine crucial topological information related to downstream classification. Furthermore, we observe that the proposed GoG propagation and the topological augmentation generally achieve more performance boost on F1-macro than F1-micro. This is because the derived supervision significantly enhance the generalizability of training data in minority classes. However, for majority classes where majority training instances already guarantee high generalizability, the enhancement would be less obvious.

*5.4.1 Effect of Neighborhood Numbers.* Here we investigate the effect that number of neighborhoods has on the performance of classification. Figure 5 shows relationships between the number of neighborhoods, the model performance and the edge homophily on MUTAG. We can clearly see that both of the F1-macro and F1-micro increases first as $r$ increases from 1 to 2 since higher $r$ means more number of neighboring graphs that are more likely to share the same label, as the homophily level at this stage is over 80% by the

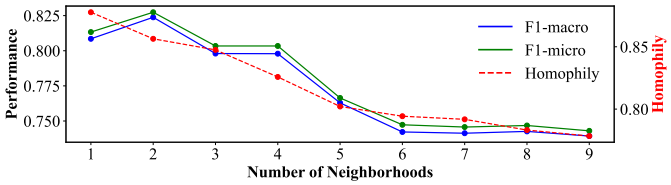


**Figure 5: Relationship between neighborhood number, edge homophily, and performance on MUTAG. The performance first increases and then decreases as the number of propagations/neighborhoods increase on $\mathcal{G}^{\mathbf{kNN}}$.**

red line, therefore we derive more and beneficial supervision. However, as we further increases $r$ from 3 to 6, the performance begins to decrease since the homophily decreases quickly at this medium stage and the additional neighborhoods have different labels and provide adverse information that compromises classification. In the last stage when $r$ proceeds to increase beyond 6, the performance gradually becomes stable, this is because directly linking each graph with its 6-top similar graphs leads to a very dense GoG and propagation on this dense GoG directly incorporates information from most of the other graphs and therefore the information that each graph receives is too noisy. Similar performance and homophily trend is also observed on DHFR in Figure 6 of Appendix E.

## 6 CONCLUSION

In this paper, we focused on imbalanced graph classification, which widely exists in the real-world while rarely explored in the literature. Noticing that unlike the node imbalance problem where we can propagate neighboring nodes' information to obtain extra supervision, graphs are isolated and have no connections with each other. Therefore, we employ a kernel-based GoG construction to establish a kNN graph and devise a two-level propagation to derive extra supervision from neighboring graphs globally. By theoretically proving the feature smoothing is upper bounded by the label smoothing and empirically showing the high homophily on the constructed kNN GoG, we guarantee the derived supervision is beneficial for downstream classification. Moreover, we employ local topological augmentation to enhance the model generalizability in discerning unseen topology of testing graphs. Experiments on 7 real-world datasets demonstrate the effectiveness of the proposed G$^2$GNN in relieving the graph imbalance issue. For future work, we plan to study adaptive GoG and incorporate attention mechanism in the GoG propagation.

## REFERENCES

[1] Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-path kernels on graphs. In *ICDM*.
[2] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *JAIR* 16 (2002), 321–357.
[3] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. 2004. Special issue on learning from imbalanced data sets. *SIGKDD Explorations* 6, 1 (2004).
[4] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *J. Med. Chem.* 34, 2 (1991), 786–797.
[5] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *CIKM*. 295–304.

[6] Paul D Dobson and Andrew J Doig. 2003. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology* 330, 4 (2003), 771–783.

[7] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A Fair Comparison of Graph Neural Networks for Graph Classification. In *ICLR*.

[8] Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp. *arXiv:2105.03075* (2021).

[9] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph Random Neural Network for Semi-Supervised Learning on Graphs. *arXiv:2005.11079* (2020).

[10] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

[11] Shawn Gu, Meng Jiang, Pietro Hiram Guzzi, and Tijana Milenkovic. 2021. Modeling multi-scale data via a network of networks. *arXiv:2105.12226* (2021).

[12] Haibo He and Yunqian Ma. 2013. Imbalanced learning: foundations, algorithms, and applications. (2013).

[13] Gabriel Idakwo, Sundar Thangapandian, Joseph Luttrell, Yan Li, Nan Wang, Zhaoxian Zhou, Huixiao Hong, Bei Yang, Chaoyang Zhang, and Ping Gong. 2020. Structure–activity relationship-based chemical classification of highly imbalanced Tox21 datasets. *Journal of cheminformatics* 12, 1 (2020), 1–19.

[14] Justin M Johnson and Taghi M Khoshgoftaar. 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6, 1 (2019), 1–54.

[15] Jonatan Kilhamn. 2015. *Fast shortest-path kernel computations using aproximate methods.* Master's thesis.

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

[17] Joffrey L Leevy, Taghi M Khoshgoftaar, Richard A Bauder, and Naeem Seliya. 2018. A survey on addressing high-class imbalance in big data. *Journal of Big Data* 5, 1 (2018), 1–30.

[18] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*.

[19] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. 2018. What makes good synthetic training data for learning disparity and optical flow estimation? *IJCV* 126, 9 (2018), 942–960.

[20] Jingchao Ni, Hanghang Tong, Wei Fan, and Xiang Zhang. 2014. Inside the atoms: ranking on a network of networks. In *KDD*. 1356–1365.

[21] Aaron Plavnick. 2008. The fundamental theorem of markov chains. *University of Chicago VIGRE REU* (2008).

[22] Liang Qu, Huaisheng Zhu, Ruiqi Zheng, Yuhui Shi, and Hongzhi Yin. 2021. ImGAGN: Imbalanced Network Embedding via Generative Adversarial Graph Networks. *arXiv:2106.02817* (2021).

[23] Min Shi, Yufei Tang, Xingquan Zhu, David Wilson, and Jianxun Liu. 2020. Multi-class imbalanced graph convolutional network learning. In *IJCAI*.

[24] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data* 6, 1 (2019), 1–48.

[25] Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. 2021. Graph-based Semi-supervised Learning: A Comprehensive Review. *arXiv:2102.13303* (2021).

[26] Fan-Yun Sun, Jordan Hoffmann, Vikas Verma, and Jian Tang. [n.d.]. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR 2020*.

[27] Jeffrey J Sutherland, Lee A O'brien, and Donald F Weaver. 2003. Spline-fitting with a genetic algorithm: A method for developing classification structure- activity relationships. *J Chem Inform Comput Sci* 43, 6 (2003).

[28] Nguyen Thai-Nghe, Zeno Gantner, and Lars Schmidt-Thieme. 2010. Cost-sensitive learning methods for imbalanced data. In *IJCNN*. IEEE.

[29] Hannu Toivonen, Ashwin Srinivasan, Ross D King, Stefan Kramer, and Christoph Helma. 2003. Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics* 19, 10 (2003), 1183–1193.

[30] Jason Van Hulse, Taghi M Khoshgoftaar, and Amri Napolitano. 2007. Experimental perspectives on learning from imbalanced data. In *ICML*. 935–942.

[31] Nikil Wale, Ian A Watson, and George Karypis. 2008. Comparison of descriptor spaces for chemical compound retrieval and classification. *KAIS* 14, 3 (2008), 347–375.

[32] Hanchen Wang, Defu Lian, Ying Zhang, Lu Qin, and Xuemin Lin. 2020. Gognn: Graph of graphs neural network for predicting structured entity interactions. *arXiv:2005.05537* (2020).

[33] Zheng Wang, Xiaojun Ye, Chaokun Wang, Jian Cui, and Philip Yu. 2020. Network embedding with completely-imbalanced labels. *IEEE TKDE* (2020).

[34] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.

[35] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *KDD*. 1365–1374.

[36] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *NeurIPS*.

[37] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *NeurIPS* 33 (2020).

[38] Tong Zhao, Tianwen Jiang, Neil Shah, and Meng Jiang. 2021. A Synergistic Approach for Graph Anomaly Detection With Pattern Mining and Feature Learning. *IEEE TNNLS* (2021).

[39] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. In *WSDM*.

[40] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open* (2020).


# A PROOF

Theorem A.1. *Suppose that the latent ground-truth mapping* $\mathcal{M}: \mathbf{P}_i^k \rightarrow \mathbf{Y}_i$ *from graph representations to graph labels is differentiable and satisfies* $\mu$*−Lipschitz constraints, i.e.,* $|\mathcal{M}(\mathbf{P}_i^k) - \mathcal{M}(\mathbf{P}_j^k)| \leq \mu||\mathbf{P}_i^k - \mathbf{P}_j^k||_2$ *for any pair of graphs* $G_i, G_j$ *(*$\mu$ *is a constant), then the label smoothing is upper bounded by the feature smoothing among graph* $G_i$ *and its neighboring graphs* $\mathcal{N}_i$ *through* (5) *with an error* $\boldsymbol{\epsilon}_i^k = \mathbf{P}_i^{k+1} - \mathbf{P}_i^k$*:*

$$\underbrace{(\hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} \mathbf{Y}_j - \mathbf{Y}_i)}_{\text{Label smoothing}} - \underbrace{(\hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2))}_{\text{feature smoothing}} \leq \mu \boldsymbol{\epsilon}_i^k,$$

Proof:

$$\begin{aligned}
&\hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} \mathbf{Y}_j = \hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} \mathcal{M}(\mathbf{P}_j^k) \\
&= \hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} (\mathcal{M}(\mathbf{P}_i^k) + \frac{\partial \mathcal{M}(\mathbf{P}_i^k)}{\partial \mathbf{P}^{k\top}} (\mathbf{P}_j^k - \mathbf{P}_i^k) + o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2)) \\
&= \mathcal{M}(\mathbf{P}_i^k) + \hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} \frac{\partial \mathcal{M}(\mathbf{P}_i^k)}{\partial \mathbf{P}^{k\top}} (\mathbf{P}_j^k - \mathbf{P}_i^k) + \hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2) \\
&= \mathbf{Y}_i + \frac{\partial \mathcal{M}(\mathbf{P}_i^k)}{\partial \mathbf{P}^{k\top}} (\hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} \mathbf{P}_j^k - \mathbf{P}_i^k) + \hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2) \\
&= \mathbf{Y}_i + \frac{\partial \mathcal{M}(\mathbf{P}_i^k)}{\partial \mathbf{P}^{k\top}} (\mathbf{P}_i^{k+1} - \mathbf{P}_i^k) + \hat{\mathbf{d}}_i^{-1} \sum_{G_j \in \mathcal{N}_i} o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2),
\end{aligned} \tag{10}$$

By $\mu$-Lipschitz property, we have:

$$\underbrace{(\mathbf{d}_i^{-1} \sum_{G_j \in \mathcal{N}_i} \mathbf{Y}_j - \mathbf{Y}_i)}_{\text{Label smoothing}} - \underbrace{(\mathbf{d}_i^{-1} \sum_{G_j \in \mathcal{N}_i} o(||\mathbf{P}_j^k - \mathbf{P}_i^k||_2))}_{\text{feature smoothing}} \leq \mu(\mathbf{P}_i^{k+1} - \mathbf{P}_i^k) = \mu \boldsymbol{\epsilon}_i^k,$$

□

Theorem A.2. *Given a graph* $\mathcal{G}$ *with m connected components* $\{C_1, C_2, ..., C_m\}$ *and their corresponding row-normalized adjacency matrices* $\{\tilde{\mathbf{A}}^{C_1}, \tilde{\mathbf{A}}^{C_2}, ..., \tilde{\mathbf{A}}^{C_m}\}$ *where* $\tilde{\mathbf{A}}^{C_i} = (\hat{\mathbf{D}}^{C_i})^{-1}\hat{\mathbf{A}}^{C_i} \in \mathbb{R}^{\mathcal{V}^{C_i}}$*, then for each component* $C_i$*,* $\lim_{k\rightarrow\infty}(\tilde{\mathbf{A}}^{C_i})^k = \mathbf{\Pi}^{C_i}$*, where* $\mathbf{\Pi}_j^{C_i} = \boldsymbol{\pi}^{C_i} \in \mathbb{R}^n$ *is the unique left eigenvector of* $\tilde{\mathbf{A}}^{C_i}$*.*

Proof: For each component $C_i$, the row-normalized adjacency matrix $\tilde{\mathbf{A}}^{C_i}$ can be viewed as a transition matrix because all entries are non-negative and each row sums to 1 due to row-normalization. Further, since $C_i$ is connected with self-loops added on each node, the corresponding Markov chain is irreducible and aperiodic. By the fundamental theorem of Markov chains [21], we have $\lim_{k\rightarrow\infty}(\tilde{\mathbf{A}}^{C_i})^k = \mathbf{\Pi}^{C_i}$, where $\mathbf{\Pi}_j^{C_i} = \boldsymbol{\pi}^{C_i} \in \mathbb{R}^n$ is unique left eigenvector of $\tilde{\mathbf{A}}^{C_i}$.

□

## B GRAPH KERNELS

In graph-structured mining, a graph kernel is essentially a kernel function that computes an inner product on graphs, which intuitively measures the similarity between pairs of graphs. Our work leverages the graph kernel to compute the pairwise similarity matrix S and further construct graph of graphs (GoG) by selecting top-$r$ similar graphs as direct neighborhoods based on S for each graph. Specifically, we use Shortest Path kernel and Weisfeiler-Lehman kernel to compute the edge homophily and only use Shortest Path kernel for constructing GoG. Therefore, we give a brief introduction on each of these two kernels as follows:

- **Shortest Path kernel** [1]: this graph kernel decomposes graphs into shortest paths and compares pairs of shortest paths according to their lengths and the labels of their endpoints.
- **Weisfeiler-Lehman kernel** [35]: this graph kernel first computes multiple rounds of the Weisfeiler-Lehman algorithm and then computes the similarity of two graphs as the inner product of vectors that collect the number of times a color occurs in the graph. Note that for two isomorphic graphs, the kernel returns a maximal similarity since the two feature vectors are identical.

## C COMPLEXITY ANALYSIS

In this section, we compare our proposed G$^2$GNN with vanilla GNN-based encoders by analyzing the time and model compelxity. Since we employ shortest path kernel for all experiments in this work, we only analyze our models with this specific graph kernel.

In comparison to vanilla GNN-based encoders, additional computational requirements potentially come from three components: kernel-based GoG construction, topological augmentation and GoG propagation. In kernel-based GoG constructions, applying shortest path kernel to calculate the similarity between every pair of graphs requires $O(n^3)$ [1] time and thus the total time complexity of this part is $O(\binom{|\mathcal{G}|}{2}\tilde{n}^3)$ ($\tilde{n} = \max_{G_i \in \mathcal{G}}(|\mathcal{V}^{G_i}|)$) due to the total $|\mathcal{G}|$ graphs. After computing the pairwise similarity, we can construct the GoG by naively thresholding out the top$-r$ similar graphs for each graph and the time complexity here is $O(|\mathcal{G}|r)$. By default $r \leq |\mathcal{G}|$, we directly have $O(|\mathcal{G}|r) < O(|\mathcal{G}|^2) = O(\binom{|\mathcal{G}|}{2}) < O(\binom{|\mathcal{G}|}{2}\tilde{n}^3)$ and hence the time complexity of the first module is $O(\binom{|\mathcal{G}|}{2}\tilde{n}^3)$. Despite the prohibitively heavy computation of $O(\binom{|\mathcal{G}|}{2}\tilde{n}^3)$, the whole module is a pre-procession computation once for all and we can further save the already computed similarity matrix S for future use, which therefore imposes no computational challenge. In topological augmentation, we augment graphs $T$ times during each training epoch and each time we either go over all its edges or nodes, therefore the total time complexity of this module during each training epoch is $O(T\sum_{G_i \in \mathcal{G}^B}(|\mathcal{V}^{G_i}| + |\mathcal{E}^{G_i}|))$. Since augmenting graphs multiple times gains no further improvement than 2 [37], we fix $T$ to be the constant 2 and therefore the total complexity of this part is linearly proportional to the size of each graph, which imposes no additional time compared with GNN encoders. Among the GoG propagation component, the most computational part comes from propagation in Eq. (5), which can be efficiently computed by applying power iteration from the edge view in $O(K|\mathcal{E}^{\mathcal{G}^{\text{kNN},B}}|)$ for each subgraph induced by graphs in batch $\mathcal{G}^B$. Based on experimental results in Fig. 6, we usually choose $r$ to be small to ensure the sparcity and the high homophily of GoG, then $O(K|\mathcal{E}^{\mathcal{G}^{\text{kNN},B}}|)$ can be neglected compared with applying GNN encoders to get representations of each graph, $O(K\sum_{G_i \in \mathcal{G}^B}|\mathcal{E}^{G_i}|)$.

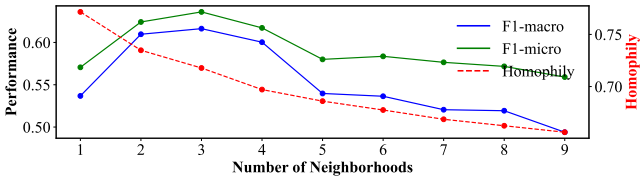


**Figure 6: Relationship between neighborhood number, edge homophily, and the performance on DHFR. The performance first increases and then decreases as the number of neighborhoods increases.**

**Table 2: Time of applying Shortest Path Kernel in calculating pairwise similarity matrix S.**

| Networks | #Graphs | Time (s) |
|---|---|---|
| REDDIT-B [35] | 2000 | 3376 |
| PTC-MR [29] | 344 | 0.257 |
| NCI1 [31] | 4110 | 11.21 |
| MUTAG [4] | 188 | 0.212 |
| PROTEINS [35] | 1113 | 11.36 |
| D&D [6] | 1178 | 574.71 |
| DHFR [27] | 756 | 3.70 |

For the model complexity, besides the parameters of GNN encoders, G$^2$GNN introduces no additional parameters and therefore its model complexity is exactly the same as traditional GNN encoders such as GIN.

In summary, our model introduces no extra model complexity but $O(\binom{|\mathcal{G}|}{2}\tilde{n}^3)$ extra time complexity in the pre-procession stage. We further presents the actual time used for applying Shortest Path kernel to compute S in Table 2. It can be clearly see that similarity matrix S is calculated in a short time for each dataset other than D&D and REDDIT-B since graphs in these two dataset are on average denser than other datasets as shown in Table 3. However, we can simply pre-compute this S once for all and reuse it for G$^2$GNN. Moreover, we can make this computation feasible by either employing the fast shortest-path kernel computations by sampling-based approximation where we sample pairs of nodes and compute shortest paths between them [15] or constructing the graph of graphs via other representation learning techniques such as graph neural networks.

Additionally, we present the edge homophily of constructed kNN graphs based on Shortest Path kernel by varying $r$ on the above seven datasets in Figure 7. Note that we also have the edge homophily of GoG for some datasets based on WL kernel in Figure 2 in the main text.

## D EXPERIMENTAL SETTING

### D.1 Performance Comparison

We randomly select 25%/25% graphs as training/validation instances and among each of them, we choose one class as minority and reduce its training graphs (increase the other one) till the imbalance ratio reaches 1:9, which creates an extremely imbalanced scenario[5].

[5] We select the amount of training and validation data as 25% to ensure the sufficiency of minority instances in both training and validation set given the data distribution is at such a skewed level

**Table 3: Statistics of datasets**

| Networks | # Graphs | # Avg-Node | # Avg-Edge | # Classes | # Node Attr | Type |
|---|---|---|---|---|---|---|
| REDDITB [35] | 2000 | 429.63 | 497.75 | 2 | \ | Reddit online discussion |
| PTCMR [29] | 344 | 14.29 | 14.69 | 2 | 18 | Chemical compound |
| NCI1 [31] | 4110 | 29.87 | 32.30 | 2 | 37 | Chemical compound |
| MUTAG [4] | 188 | 17.93 | 19.79 | 2 | 7 | Chemical compound |
| PROTEINS [35] | 1113 | 39.06 | 72.82 | 2 | 3 | Protein compound |
| D&D [27] | 1178 | 284.32 | 715.66 | 2 | 89 | Protein compound |
| DHFR [27] | 756 | 42.43 | 44.54 | 2 | 3 | Protein compound |

**Table 4: Graph classification performance on seven datasets. Note that the standard deviation is relatively higher due to the fact we focus on the imbalance problem and use 50 different data splits (i.e., having different training data distributions), which significantly affect the downstream classification.**

| Model | **MUTAG** (5:45) | | **PROTEINS** (30:270) | | **D&D** (30:270) | | **NCI1** (100:900) | |
|---|---|---|---|---|---|---|---|---|
| | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro |
| GIN | 52.50 ± 18.70 | 56.77 ± 14.14 | 25.33 ± 7.53 | 28.50 ± 5.82 | 9.99 ± 7.44 | 11.88 ± 9.49 | 18.24 ± 7.58 | 18.94 ± 7.12 |
| $GIN_{us}$ | 78.03 ± 7.62 | 78.77 ± 7.67 | 65.64 ± 2.67 | 71.55 ± 3.19 | 41.15 ± 3.74 | 70.56 ± 10.28 | 59.19 ± 4.39 | 71.80 ± 7.02 |
| $GIN_{rw}$ | 77.00 ± 9.59 | 77.68 ± 9.30 | 54.54 ± 6.29 | 55.77 ± 7.11 | 28.49 ± 5.92 | 40.79 ± 11.84 | 36.84 ± 8.46 | 39.19 ± 10.05 |
| $GIN_{st}$ | 74.61 ± 9.66 | 75.11 ± 9.87 | 56.07 ± 7.95 | 57.85 ± 8.70 | 27.08 ± 8.63 | 39.01 ± 15.87 | 40.40 ± 9.63 | 44.48 ± 12.05 |
| InfoGraph | 69.11 ± 9.03 | 69.68 ± 7.77 | 35.91 ± 7.58 | 36.81 ± 6.51 | 21.41 ± 4.51 | 27.68 ± 7.52 | 33.09 ± 3.30 | 34.03 ± 3.68 |
| $InfoGraph_{us}$ | 78.62 ± 6.84 | 79.09 ± 6.86 | 62.68 ± 2.70 | 66.02 ± 3.18 | 41.55 ± 2.32 | 71.34 ± 6.76 | 53.38 ± 1.88 | 62.20 ± 2.63 |
| $InfoGraph_{rw}$ | 80.85 ± 7.75 | 81.68 ± 7.83 | 65.73 ± 3.10 | 69.60 ± 3.68 | 41.92 ± 2.28 | 72.43 ± 6.63 | 53.05 ± 1.12 | 62.45 ± 1.89 |
| GraphCL | 66.82 ± 11.56 | 67.77 ± 9.78 | 40.86 ± 6.94 | 41.24 ± 6.38 | 21.02 ± 3.05 | 26.80 ± 4.95 | 31.02 ± 2.69 | 31.62 ± 3.05 |
| $GraphCL_{us}$ | 80.06 ± 7.79 | 80.45 ± 7.86 | 64.21 ± 2.53 | 65.76 ± 2.61 | 38.96 ± 3.01 | 64.23 ± 8.10 | 49.92 ± 2.15 | 58.29 ± 3.30 |
| $GraphCL_{rw}$ | 80.20 ± 7.27 | 80.84 ± 7.43 | 63.46 ± 2.42 | 64.97 ± 2.41 | 40.29 ± 3.31 | 67.96 ± 8.98 | 50.05 ± 2.09 | 58.18 ± 3.08 |
| $G^2GNN_{\emptyset}$ | 81.18 ± 7.24 | 81.77 ± 7.37 | 66.83 ± 2.85 | 72.24 ± 3.96 | 45.04 ± 2.56 | 82.33 ± 8.08 | 59.48 ± 3.38 | 73.02 ± 5.75 |
| $G^2GNN_e$(w/o kNN) | 77.11 ± 6.30 | 77.84 ± 6.24 | 66.31 ± 2.69 | 72.02 ± 3.81 | 41.70 ± 3.32 | 72.07 ± 9.60 | 59.57 ± 3.27 | 72.76 ± 4.85 |
| $G^2GNN_n$(w/o kNN) | 79.56 ± 7.44 | 80.16 ± 7.43 | 66.29 ± 2.55 | 72.30 ± 3.16 | 41.00 ± 5.51 | 70.76 ± 13.84 | 61.16 ± 2.94 | 75.67 ± 4.94 |
| $G^2GNN_e$ | 80.37 ± 6.73 | 81.25 ± 6.87 | 67.70 ± 2.96 | 73.10 ± 4.05 | 43.25 ± 3.91 | 77.03 ± 9.98 | 63.60 ± 1.57 | 72.97 ± 1.81 |
| $G^2GNN_n$ | 83.01 ± 7.01 | 83.59 ± 7.14 | 67.39 ± 2.99 | 73.30 ± 4.19 | 43.93 ± 3.46 | 79.03 ± 10.78 | 64.78 ± 2.86 | 74.91 ± 2.14 |

| Model | **PTC-MR** (9:81) | | **DHFR** (12:108) | | **REDDIT-B** (50:450) | | **Ave. Rank** | |
|---|---|---|---|---|---|---|---|---|
| | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro | F1-macro | F1-micro |
| GIN | 17.74 ± 6.49 | 20.30 ± 6.06 | 35.96 ± 8.87 | 49.46 ± 4.90 | 33.19 ± 14.26 | 36.02 ± 17.38 | 15.00 | 15.00 |
| $GIN_{us}$ | 44.78 ± 8.01 | 55.43 ± 14.25 | 55.96 ± 10.06 | 59.39 ± 6.52 | 66.71 ± 3.92 | 83.00 ± 5.18 | 7.71 | 7.00 |
| $GIN_{rw}$ | 36.96 ± 14.08 | 43.09 ± 20.01 | 55.16 ± 9.47 | 57.78 ± 6.69 | 45.17 ± 8.46 | 51.92 ± 12.29 | 11.86 | 11.86 |
| $GIN_{st}$ | 36.30 ± 11.45 | 40.04 ± 15.32 | 56.06 ± 9.60 | 58.48 ± 6.42 | 60.05 ± 4.14 | 73.59 ± 6.05 | 11.29 | 11.43 |
| InfoGraph | 25.85 ± 6.14 | 26.71 ± 6.50 | 50.62 ± 8.33 | 56.28 ± 4.58 | 57.67 ± 3.80 | 67.10 ± 4.91 | 13.00 | 13.14 |
| $InfoGraph_{us}$ | 44.29 ± 4.69 | 48.91 ± 7.49 | 59.49 ± 5.20 | 61.62 ± 4.18 | 67.01 ± 3.34 | 78.68 ± 3.71 | 7.14 | 7.29 |
| $InfoGraph_{rw}$ | 44.09 ± 5.62 | 49.17 ± 8.78 | 58.67 ± 5.82 | 60.24 ± 4.80 | 65.79 ± 3.38 | 77.35 ± 3.96 | 6.71 | 6.57 |
| GraphCL | 24.22 ± 6.21 | 25.16 ± 5.25 | 50.55 ± 10.01 | 56.31 ± 6.12 | 53.40 ± 4.06 | 62.19 ± 5.68 | 13.71 | 13.57 |
| $GraphCL_{us}$ | 45.12 ± 7.33 | 53.50 ± 13.31 | 60.29 ± 9.04 | 61.71 ± 6.75 | 62.01 ± 3.97 | 75.84 ± 3.98 | 7.57 | 7.71 |
| $GraphCL_{rw}$ | 44.75 ± 7.62 | 52.22 ± 13.24 | 60.87 ± 6.33 | 61.93 ± 5.15 | 62.79 ± 6.93 | 76.15 ± 9.15 | 7.29 | 7.57 |
| $G^2GNN_{\emptyset}$ | 45.92 ± 9.52 | 55.24 ± 14.78 | 59.07 ± 9.45 | 61.51 ± 6.17 | 68.19 ± 3.19 | 86.98 ± 4.53 | 3.57 | 3.14 |
| $G^2GNN_e$(w/o kNN) | 45.43 ± 10.50 | 57.70 ± 16.90 | 59.25 ± 7.48 | 61.00 ± 5.66 | 68.40 ± 3.34 | 84.96 ± 3.14 | 5.00 | 5.29 |
| $G^2GNN_n$(w/o kNN) | 46.18 ± 6.99 | 55.84 ± 14.71 | 57.34 ± 8.72 | 59.22 ± 6.98 | 67.01 ± 2.71 | 83.30 ± 3.52 | 5.86 | 5.29 |
| $G^2GNN_e$ | 46.40 ± 7.73 | 56.61 ± 13.72 | 61.63 ± 10.02 | 63.61 ± 6.05 | 68.39 ± 2.97 | 86.35 ± 2.27 | 2.14 | 2.71 |
| $G^2GNN_n$ | 46.61 ± 8.27 | 56.70 ± 14.81 | 59.72 ± 6.83 | 61.27 ± 5.40 | 67.52 ± 2.60 | 85.43 ± 1.80 | 2.14 | 2.43 |

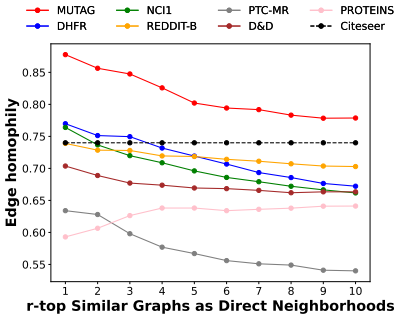


**Figure 7: Edge homophily of constructed kNN GoG on each dataset.**

## D.2 Influence of Imbalance Ratio

We vary the imbalance ratio from 1:9 to 9:1 by fixing the total number of training and validation graphs as 25%/25% of the whole dataset as before and gradually changing the number of graphs from different classes, which exhausts the imbalance scenarios from being balanced (5:5) to the extremely imbalanced (1:9 or 9:1) scenarios.

## E DETAILED EXPERIMENTAL RESULTS

Table 4 presents the F1-macro and F1-micro scores along with their standard deviation for all variants of our model and other baselines. We emphasize that the larger standard deviation (std) in our setting compared with small std in traditional full supervised graph classification where 90% graphs are used for training [37] is normal. This is because the natural imbalance will cause the training data during each running to be significantly different across different runnings, which leads to different performance and hence has such a larger standard deviation. We further argue that this standard deviation cannot be reduced by only increasing the number of runs due to the imbalance nature of the problem.