Overcoming Pitfalls in Graph Contrastive Learning Evaluation: Toward Comprehensive Benchmarks

Qian Ma[1], Hongliang Chi[1], Hengrui Zhang[2], Kay Liu[2], Zhiwei Zhang[3], Lu Cheng[2], Suhang Wang[3], Philip S. Yu[2], Yao Ma[1]

[1]Rensselaer Polytechnic Institute, [2]University of Illinois Chicago, [3]Pennsylvania State University

maq5@rpi.edu, chih3@rpi.edu, hzhan55@uic.edu, zliu234@uic.edu, zbz5349@psu.edu, lucheng@uic.edu, szw494@psu.edu, psyu@uic.edu, may13@rpi.edu

ABSTRACT

The rise of self-supervised learning(SSL), which operates without the need for labeled data, has garnered significant interest within the graph learning community. This enthusiasm has led to the development of numerous self-supervised learning methods, such as Graph Contrastive Learning (GCL) techniques, all aiming to create a versatile graph encoder that leverages the wealth of unlabeled data for various downstream tasks. However, the current evaluation standards for GCL approaches are flawed due to the need for extensive hyper-parameter tuning during pre-training and the reliance on a single downstream task for assessment. These flaws can skew the evaluation away from the intended goals, potentially leading to misleading conclusions. In our paper, we thoroughly examine these shortcomings and offer fresh perspectives on how GCL methods are affected by hyper-parameter choices and the choice of downstream tasks for their evaluation. Additionally, we introduce an enhanced evaluation framework designed to more accurately gauge the effectiveness, consistency, and overall capability of GCL methods. Our code implementation is available to ease reproducibility on `https://github.com/GraphTL-Bench/BGPM`.

1. INTRODUCTION

Graph Neural Networks (GNNs) [17; 8; 27] have emerged as a powerful tool for learning representations from graph-structured data, demonstrating remarkable success across various fields including social network analysis [14; 5; 9], molecular biology [4; 34; 13], recommendation systems [23; 33; 20], and traffic prediction [29; 21; 16]. By leveraging the rich relational information inherent in graphs, GNNs can capture complex patterns that traditional neural network architectures struggle to process. However, the efficacy of GNNs is largely contingent upon the availability of task-dependent labels to learn meaningful representations [10; 31; 37; 18]. Unlike labeling in more common modalities such as images, videos, texts, and audio, annotating graphs presents significant challenges due to the complex and often domain-specific nature of graph data [18; 11; 3]. This has led to a growing interest in self-supervised learning methods as a means to circumvent the limitations imposed by the need for extensive labeled datasets. Among the various categories of self-supervised learning, graph contrastive learning (GCL) has emerged as a particularly promising approach [28; 43; 44; 36; 26; 22; 39].

The primary goal of GCL is to pre-train an encoder capable of generating high-quality graph representations without relying on label information, with the hope that these representations can be effectively utilized across a wide array of downstream tasks [12; 2; 7; 18]. However, the current evaluation protocols for GCL methods exhibit significant shortcomings that fail to align with these fundamental goals as detailed below:

- GCL methods typically consist of multiple hyper-parameters due to their unique designs in augmentation methods or contrastive objectives. In existing evaluation protocols, these hyper-parameters are often tuned for each graph dataset, which plausibly involves the use of a validation set from a downstream task. However, in practice, this process contradicts the premise of pre-training without task-specific labels. This would be particularly problematic if the GCL methods are highly sensitive to hyper-parameter configurations.
- In existing evaluation procedures, the evaluation of GCL methods is predominantly focused on a single downstream task, usually node classification, conducted on the same dataset used for encoder pre-training. Such a limited evaluation framework may not accurately reflect the encoder's versatility across diverse tasks, potentially leading to misleading comparisons among different GCL methods.

In our study, we conduct a thorough empirical analysis focusing on two crucial elements: (a) how GCL methods' performance is affected by hyper-parameter adjustments in the pre-training phase, and (b) the extent to which a single downstream task can accurately reflect the overall efficacy of GCL methods. Our investigation, detailed in Section 4, confirms that the current evaluation systems are indeed compromised by these concerns. Particularly, we observe that some GCL methods are highly sensitive to the settings of hyper-parameters. Although these methods can perform well when hyper-parameters are optimally tuned, their effectiveness can substantially decrease with less-than-ideal settings, making them less practical for pre-training situations where adapting hyper-parameters for each specific downstream task is impractical. Despite this, such methods may still show strong results within the current evaluation models, which do not fully account for the challenges posed by their sensitivity to hyper-parameters [39; 43]. Additionally, evaluating these methods based on a single downstream

task often does not provide a complete picture of their capabilities, as the performance of GCL methods can vary significantly across different tasks. This inconsistency highlights the limitations of current evaluation approaches that focus on singular tasks, thus failing to offer a comprehensive assessment of GCL methods' overall performance.

Hence, to address the identified issues, we propose a new evaluation protocol aimed at a more comprehensive and accurate assessment of GCL methods. This improved protocol incorporates a comprehensive analysis across diverse hyper-parameter configurations and extends the evaluation to include multi-label dataset evaluation. Through these enhancements, our protocol seeks to mitigate the impact of hyper-parameter sensitivity and broaden the scope of evaluation beyond a single task, thereby offering a more comprehensive understanding of GCL method performance.

2. PRELIMINARIES

Graph Contrastive Learning (GCL) focuses on learning superior node representations by distinguishing between node pairs that share similar semantics and those that do not. An anchor node, which can be any selected node in augmented graphs, is paired with semantically similar nodes or graphs (positive examples) to create positive pairs and with dissimilar nodes or graphs (negative examples) to establish negative pairs. GCL models aim to map graph elements such as nodes or graphs into an embedding space where positive pairs are pulled closely together while negative pairs are pushed far apart. To accomplish this, GCL methods are designed with a variety of components. As well-summarized in a prior work [41], GCL methods are normally designed differently in the following three key components:

- **Data Augmentation:** Data augmentation in GCL aims to create variations of a graph that preserve its major semantic information, thereby helping models to create positive and negative examples. This involves two main approaches: topology transformations and feature transformations. Topology augmentations adjust the graph structure through methods like edge dropping etc. Feature augmentations alter node features by masking to generate diverse representations. The common hypeter-parameters derived from these components are *drop edge rate* and *drop feature rate*.

- **Contrastive Mode:** In GCL, contrasting modes define how the similar and dissimilar samples of an anchor node are selected for comparison. There are three primary modes: *local-local* contrasts nodes at the same level, *global-global* contrasts entire graph embeddings, and *global-local* contrasts nodes with their graph-level representation. The choice of mode depends on the task, with node-focused tasks typically using local-local and global-local modes.

- **Contrastive Objective:** Contrastive objectives quantify the similarity between positive pairs and the difference from negative pairs. Common objectives include Information Noise Contrastive Estimation (InfoNCE) and Jensen-Shannon Divergence (JSD), which require explicit negative sampling. Other objectives like Bootstrapping Latent Loss (BL) and Barlow Twins (BT) do not require negatives, focusing instead on enhancing positive pair similarity and feature diversity. The temperature hyper-parameter τ in InfoNCE is typically adjustable in GCL methods utilizing it as the contrastive objective.

Investigated Methods: Building on the general paradigm of Graph Contrastive Learning (GCL), we summarize the key hyper-parameters influencing graph representation across various methods investigated in our work with a brief introduction as follows.

- **1. DGI:** As a pioneer work of GCL, DGI [28] maximizes mutual information between global graph embeddings and local node embeddings, leveraging JSD as its contrastive loss. Under its design, it has two hyper-parameters to vary, *hidden dimension* and *layer number* of the backbone GNN used to extract representations.

- **2. MVGRL:** [11] leverages multiple graph views to capture comprehensive structural patterns, applying different GNN configurations for each view to enrich node representations. MVGRL is with the same set of adjustable hyper-parameters as the DGI, but two backbone GNNs are applied on different augmented views(*i.e.,* there are two sets of *hidden dimension* and *layer number*).

- **3. GRACE:** [43] focuses on local node-level embedding contrast between two randomly augmented graph views. Each view is copied from the input graph first, next undergoing modifications where a certain percentage of edges and node features are removed and masked, determined by the specified *drop edge rate* and *drop feature rate*, respectively. Different from JSD loss adopted DGI and MVGRL, the adopted contrastive loss for GRACE is InfoNCE controlled by an additional parameter τ.

- **4. BGRL:** [26] utilizes Bootstrapping Latent loss for graph learning without negative samples, with similar two augmented graph views adjustable with *drop edge rate* and *drop feature rate*.

- **5. Graph Barlow Twins:** [1] aims to make the cross-correlation matrix of embeddings from two views as close to the identity matrix as possible, using different hyper-parameters *drop edge rate* and *drop node rate* for graph augmentations on two views.

- **6. CCA-SSG:** [36] first generates shared node representations from two augmented graph views with similar hyper-parameters mentioned before, then using Canonical Correlation Analysis to maximize correlation between views and decorrelate feature dimensions within each view. CCA-SSG has an additional hyper-parameter *loss reweighting factor* for the trade-off between losses.

- **7. SUGRL:** [22] simplifies the contrastive learning process by omitting graph augmentation and similarity determination steps, focusing instead on increasing inter-class differences and decreasing intra-class differences with a triplet loss and an upper bound loss. Given this specific design, there are four unique hyper-parameters for SUGRL, *loss reweighting factor*, *iteration number* for its unique negative samples generation module, *margin parameter*, a hyper-parameter of the triplet loss that penalizes a small gap in distances between a positive pair and a negative pair, and a *bound parameter*, a non-negative tuning parameter used in the upper bound loss.

- **8. COSTA:** [38] is proposed to address biases in graph augmentation by employing feature augmentation, allowing for better control over data distribution in the latent space. COSTA has τ for its InfoNCE contrastive loss and *drop edge rate* and *drop node rate* for two different augmented views.

- **9. SFA:** [39] proposed spectral feature augmentation for GCL, which is designed to re-balance the feature spectrum by iteratively removing low-rank information from the feature matrix. Therefore, apart from *drop edge rate* and *drop node rate*, there are two more hyper-parameters for SFA: k as the number of iterations used in its spectral feature augmentation, and again τ in its adopted InfoNCE loss.

3. DATASETS AND EVALUATION METRICS

3.1 Datasets

In this study, we utilize a diverse collection of datasets to evaluate the performance of GCL models across different domains and tasks:

Cora (Cora), Citeseer (Cite), Pubmed (Pub): Provided by [32], these citation networks include articles as nodes with bag-of-words features, edges as citations, and node labels indicating article types. **Amazon Computers (Am-Com) and Amazon Photo (Am-Ph):** From [24], these are part of the Amazon co-purchase graph, where nodes represent goods, edges indicate co-purchases, features are from bag-of-words reviews, and labels denote product categories. **Coauthor CS (Co-CS) and Coauthor Physics (Co-Phy):** From the Microsoft Academic Graph, these networks depict authors as nodes and co-authorship as edges. Features are derived from keywords in publications, with labels reflecting research areas. **Protein-Protein Interaction (PPI):** As cited in [45], this dataset includes networks for human tissues, nodes with multiple gene ontology labels, and features such as positional gene sets and immunological signatures. **PCG, HumLoc, and EukLoc:** Curated by [40] for multi-label classification, focusing on protein phenotype prediction (PCG) and subcellular localization in humans (HumLoc) and eukaryotes (EukLoc). The detailed statistics of datasets can be found in Table 1.

Table 1: Datasets statistics. Kindly note that PCG, HumLoc, EukLoc and PPI are multi-label node classification dataset while the others are multi-class node classification datasets.

Dataset	#nodes	#edges	#features	#class
PCG	3k	37k	32	15
HumLoc	3.10k	18k	32	14
EukLoc	7.70K	13K	32	22
PPI	56,944	818,716	50	121
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3
CS	18,333	163,788	6,805	15
Physics	34,493	495,924	8,415	5
Computers	13,752	491,722	767	10
Photo	7,650	238,162	745	8

3.2 Evaluation Metrics

Accuracy is the most commonly used metric for evaluating multi-node classification tasks. It measures the proportion of correctly predicted instances over the total instances. Specifically, accuracy is calculated as the ratio of the sum of true positives and true negatives to the total number of cases, making it a straightforward and intuitive measure of model performance.

F1 score, which is essential for assessing model performance in multi-label node classification tasks, harmonizes precision and recall into a single metric. Below is a breakdown for Macro and Micro F1 Score: *Macro F1 Score* Computes F1 scores separately for each label and then averages them. Ensures equal importance is given to each label, which is especially useful for datasets with imbalanced label distributions. *Micro F1 Score* Aggregates true positives, false positives, and false negatives across all labels to compute a collective F1 score. Focuses on overall performance, weighting more heavily towards labels that appear more frequently.

4. ISSUES OF EXISTING EVALUATION PROTOCOL

The primary objective of graph contrastive learning is to pre-train an encoder capable of producing high-quality graph representations without label information, with the anticipation that these representations can be efficiently applied to a diverse range of downstream tasks. However, the existing evaluation protocol for graph contrastive learning methods deviates from the aforementioned goals, thus inadequately assessing their efficacy. Specifically, existing evaluation protocols have the following deficiencies:

- As elucidated in Section 2, GCL methods typically involve numerous hyper-parameters during the pre-training stage. In the current evaluation protocol, these hyper-parameters are typically optimized for each graph dataset [43; 39]. The hyper-parameter selection is plausibly carried out with a validation set of a downstream task—an approach that is ideally not assumed during the pre-training phase of the encoder. This is essentially problematic especially if the encoders are sensitive to the hyper-parameters.

- The existing evaluation protocol predominantly concentrates on a single downstream task, i.e., node classification on the same dataset the encoder is trained. Such a constrained assessment framework may inadequately capture the encoder's performance across the entire task space, rendering comparisons between different GCL methods potentially misleading. In particular, if the downstream task does not effectively represent the diverse range of possible applications, the basis for comparing GCL methods becomes flawed, as it fails to reflect the true generality and adaptability of the encoder's learned representations.

In this section, we undertake empirical experiments to substantiate the aforementioned concerns. Our experiments seek to investigate the following two research questions:

- What is the extent of sensitivity of GCL methods to hyper-parameters during pre-training stage?

- Is a single downstream task representative enough to comprehensively assess the performance of GCL methods?

4.1 Sensitivity to Hyper-Parameters

In this section, we aim to investigate the sensitivity of the GCL methods to the hyper-parameters in the pre-training stage. Specifically, we intend to commence our investigation by undertaking a hyper-parameter tuning process for each GCL method, utilizing the validation set of the downstream task by the conventional model selection procedure. This approach will not only facilitate gaining a deeper understanding of the capabilities inherent within GCL methods but also establish baselines for assessing their sensitivity to hyper-parameters—we will compare their performance under optimally tuned hyper-parameters against the outcomes derived from suboptimal hyper-parameters.

4.1.1 Settings

In this part, we delineate the general experimental settings for our investigation. Following existing works [42; 39; 22; 44], the pre-training is conducted on the same dataset used for downstream task evaluation without access to labels.

Search Space of Hyper-Parameters. As explicated in Section 2, various GCL methods typically involve distinct numbers of hyper-parameters, thereby yielding diverse hyper-parameter search spaces for these GCL methods. Therefore, given the computational expense associated with GCL methods, it is impractical and inequitable to tune all hyper-parameters exhaustively. Instead, we advocate to randomly sample 20 combinations of hyper-parameters from the search space of each method. Subsequently, the "optimal" set of hyper-parameters is selected from these sampled combinations for each method. Thus, the selected hyper-parameters in this work differ from the ones reported in existing works [43; 39]. These 20 combinations of hyper-parameters can be found along with our code implementation [1]. We also outline hyper-parameters search space within typical range in Appendix A. For all methods, the backbone model is GCN [15]. Specifically, throughout the pre-training phase of each encoder, we save encoder versions at predetermined epochs, namely at $50, 100, 500, 1,000$, and $10,000$ epochs. This systematic approach enables the evaluation of the encoder's performance at different stages of training without imposing additional computational burdens, requiring only minimal storage overhead.

Pipeline Our evaluation procedure adheres to the pipeline employed in previous works [28; 43; 44; 42]. Specifically, we fix the pre-trained encoder and tune a linear task head for the downstream node classification task *e.g.,* a linear multi-class classifier utilizing the node representations from the encoder as input. The performance of the downstream node classification task is considered as the encoder's performance *i.e.,* the performance of the GCL method.

4.1.2 Results analysis

We conduct experiments on 7 multi-class node classification datasets, which are introduced in Section 3. We present and analyze the results as follows.

The Capability of GCL Methods. We illustrate the capability of GCL methods with their corresponding "optimal" hyper-parameters selected from the pre-defined 20 combinations. The "optimal" performance of 9 representative methods across 7 datasets are shown in Figure 1.

[1] https://github.com/GraphTL-Bench/BGPM

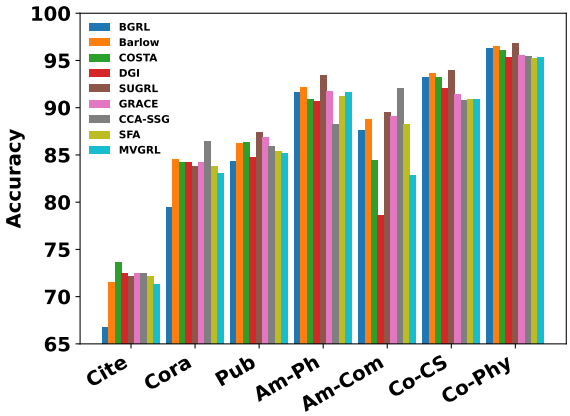


Figure 1: Performance with optimal hyper-parameters.

As shown in Figure 1, we observe that with the "optimal" hyper-parameters, all methods are capable of achieving reasonable performance that is close to the reported results in the original papers [36; 39; 38] across all datasets. However, the rankings between them occasionally diverge from the rankings reported in existing works. This is mainly due to the different hyper-parameters adopted, which to some extent underscore the sensitivity of the GCL methods to the hyper-parameters. Note that in this experiment, our objective is not to replicate the results from previous studies but rather to demonstrate that all GCL methods can achieve robust performance when appropriately tuned.

The Selected Hyper-parameters. In order to investigate how the GCL methods are sensitive to hyper-parameters, we first analyze the selected optimal set of hyper-parameters for different methods across various datasets. In particular, we

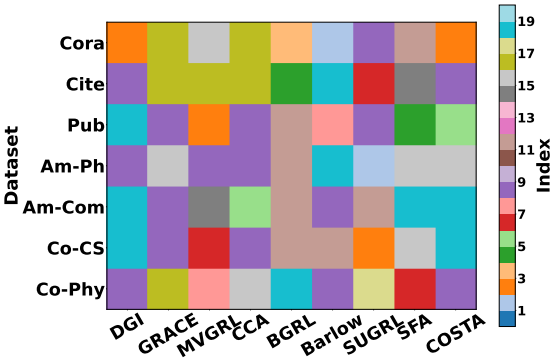


Figure 2: Index of the optimal hyperparameter set for different methods on different datasets. Indices range from 0 to 19.

document the indices of the selected sets of optimal hyper-parameters across all datasets for each method. We illustrate the hyper-parameter selection process for all datasets in Figure 2, where various indices are represented by different colors in the heatmap.

As depicted in Figure 2, for all methods, there is no dominating set of hyper-parameters valid for all datasets, i.e., different datasets prefer different hyper-parameters. These findings underscore the sensitivity of GCL methods to hyper-parameters, indicating the impracticality of employing identical hyper-parameter settings across all datasets, even for the same method. Such observations are consistent with existing works [39; 43], which have shown that GCL methods often require varied hyper-parameter configurations when applied to different datasets.

Variance Across Different Hyper-parameters. The preceding results also indicate that the methods are sensitive to hyper-parameters. In this part, we aim to further understand how significantly the GCL methods are affected by the hyper-parameters. In particular, for each method, we assess the downstream performance for the 20 pre-trained encoders corresponding to the 20 combinations of hyper-parameters.

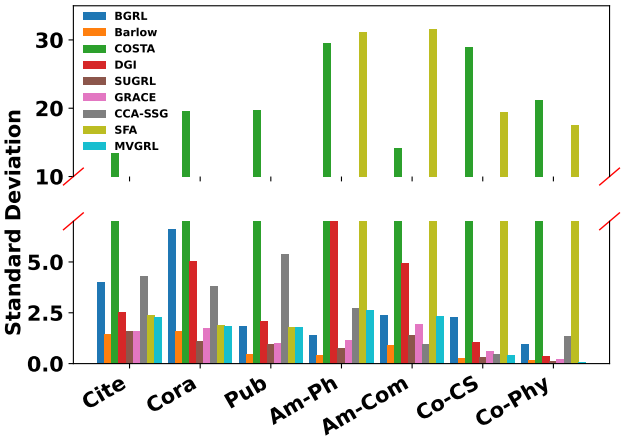


Figure 3: Standard deviation across 20 sets of hyper-parameters.

Subsequently, we compute the standard deviation of these downstream task performances for each method across all datasets. The results are presented in Figure 3, where the key observations emerge.

- The standard deviation is contingent upon the method, indicating varying degrees of sensitivity among different methods. Notably, COSTA and SFA demonstrate considerably higher instability compared to other methods, while Barlow and SUGRL consistently yield stable results.
- The standard deviation is influenced by the dataset utilized for the evaluation, suggesting varying levels of stability across datasets for the same method. For instance, SFA exhibits stability on datasets such as Citesser, Cora, and Pubmed but displays pronounced instability on other datasets. Conversely, Barlow exhibits greater stability on Coauthor-CS and Physics datasets, albeit with relatively lower stability on Citeseer and Cora datasets.

Summary. In conclusion, the effectiveness of most GCL methods is notably influenced by the selection of hyper-parameters. Additional evidence on the sensitivity and preference to hyper-parameter of graph SSL methods designed for graph-level tasks, which further demonstrates the critical role hyper-parameters play in determining model performance. When the hyper-parameters are properly optimized, most GCL methods are capable of delivering satisfactory performance for a given downstream task. However, the practical challenge arises from the fact that tuning GCL methods hyper-parameters to perfection is often not viable, primarily because downstream tasks are not predefined during the pre-training phase. Therefore, *when evaluating the efficacy of a GCL method, it is crucial to consider its sensitivity to hyper-parameters.* Specifically, GCL methods that exhibit minimal sensitivity to changes in hyper-parameters are generally more desirable. To achieve acceptable performance on downstream tasks, such methods necessitate a reduction in the workload associated with hyper-parameter selection during the pre-training stage.

4.1.3 *Further Verification on Graph Prompting*

Building on the 'pre-training and fine-tuning' paradigm of Graph Contrastive Learning (GCL), recent studies have explored an extended framework that integrates 'pre-training, prompting, and fine-tuning' [25; 6; 19], where prompting introduces a small set of learnable tokens or structures into the input graph to reformulate downstream tasks so they better align with the pre-training objective. However, as long as 'pre-training' remains a component, the challenges discussed earlier persist *i.e.,* the sensitivity of methods to hyper-parameters may impact evaluation outcomes.

To investigate this, we conducted experiments using the All-in-One framework [25], following the experimental settings outlined in Section 4 to assess hyper-parameter sensitivity[2]. In accordance with the original implementation, we evaluated two GNN backbones, GAT [27] and GCN [15],pre-trained with GraphCL [35] and SimGRACE [30].

Our primary observation from Figure 4 is that, when properly tuned, models with the same backbone but different pre-training methods achieve comparable performance. For example, on the Cora and CiteSeer datasets, GraphCL does not demonstrate a clear advantage over SimGRACE.

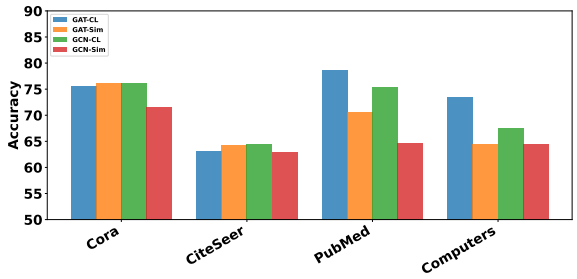


Figure 4: All-in-One performance with optimal hyper-parameters. The suffix "CL" denotes models pre-trained with GraphCL, while "Sim" represents models pre-trained with SimGRACE.

However, failing to account for hyper-parameter sensitivity may lead to an incomplete evaluation of benchmark performance. As shown in Figure 5, SimGRACE exhibits more stable performance on the Cora and CiteSeer datasets, as reflected by a lower standard deviation. If only the "op-

[2]https://github.com/VAN-QIAN/ProG/tree/downstream

timal" performance is considered while ignoring sensitivity, an important insight is overlooked: SimGRACE may be a more reliable choice than GraphCL on the CiteSeer dataset.

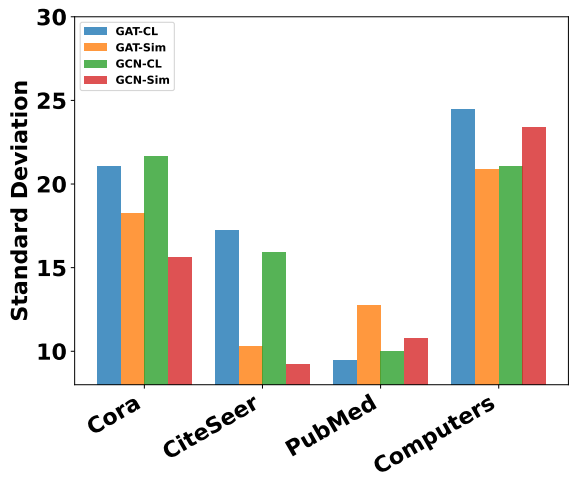


Figure 5: All-in-One standard deviation across 20 sets of hyper-parameters. The suffix "CL" denotes models pre-trained with GraphCL, while "Sim" represents models pre-trained with SimGRACE.

4.2 Representativeness of Downstream Tasks

In this section, we explore the adequacy of using a single downstream task to evaluate GCL methods. Our objective is to conduct experiments comparing the performance of various GCL methods across multiple downstream tasks and to assess whether the relative rankings of these methods remain consistent across different tasks. This investigation seeks to elucidate the representativeness of a single task as a benchmark for assessing the effectiveness of different GCL methods.

Pipeline. Following the aforementioned schedule, for the PPI dataset with 121 labels, we have 121 binary classification tasks. Subsequently, we utilize the evaluation setting as described in Section *4.1.1* to select the best set of hyper-parameters for each task. Thus, after the model selection, for each GCL method, we will have 121 selected "optimal" encoders corresponding to the 121 tasks. We then proceed to compare the GCL methods on each task with their corresponding "optimal" encoders specific to this task. Specifically, for each task, we rank the performance of the 9 methods. Therefore, we compile a total of 121 ranking lists, with each list comprising 9 GCL methods.

4.2.1 Results Analysis

We present key statistics extracted from the 121 ranking lists in Table 2. For each GCL method, the "Min" and "Max" denote its minimum and maximum rankings obtained among the 121 tasks, respectively. Additionally, the "Mean" and "Std_dev" indicate the mean rank and standard deviations of the rank among the 121 tasks, respectively.

The results in table 2 reveal that the rankings of all GCL methods vary across different downstream tasks. Each method can exhibit significant performance disparities, achieving high rankings on certain tasks while performing poorly on others, as evidenced by the "Min" and "Max" rankings. Moreover, the standard deviation of each method underscores the prevalence of this phenomenon. These findings indicate that no single downstream task is representative enough for the entire downstream task space. Relying solely on a single downstream task for evaluation may lead to an inaccurate assessment of the effectiveness.

Table 2: Optimal Performance Ranking Based on Accuracy

Model	Min	Max	Mean	Std_dev
Barlow	1	9	3.058	1.489
BGRL	1	9	3.462	2.691
CCA-SSG	2	9	5.935	1.245
COSTA	1	9	5.496	1.687
DGI	4	9	8.545	1.113
GRACE	1	8	3.025	1.474
MVGRL	1	8	5.033	2.147
SFA	2	9	7.074	1.337
SUGRL	1	9	3.371	2.378

Given the impracticality of hyper-parameter selection for each GCL method, we adopt an alternative experimental approach that involves averaging the performance across 20 sets of hyper-parameters. This methodology aims to mitigate the bias introduced by hyper-parameter sensitivity. Specifically, for each downstream task, we compute the average performance of a GCL method over these 20 sets of hyper-parameters. Subsequently, we apply this average performance metric to rank the GCL methods for each of the 121 tasks, resulting in 121 distinct ranking lists. These results are presented in Table 3. Similar observations as in Table 2 can be made from Table 3, further confirming that a single task is not sufficiently representative.

Table 3: Average Performance Ranking Based on Accuracy

Model	Min	Max	Mean	Std_dev
Barlow	1	9	4.876	2.039
BGRL	1	9	4.694	3.149
CCA-SSG	3	9	6.479	1.472
COSTA	1	9	3.843	2.045
DGI	4	9	5.157	2.077
GRACE	1	8	1.901	1.576
MVGRL	1	8	5.570	1.983
SFA	2	9	7.694	1.757
SUGRL	1	9	4.785	2.205

Summary. It is evident that relying on a single downstream task for comparing different GCL methods is not sufficient to obtain a comprehensive understanding of their performance. Therefore, *when evaluating the efficacy of GCL methods, it is critical to conduct comprehensive experiments on multiple downstream tasks.*

Discussion. To incorporate multiple downstream tasks, a straightforward idea is to include tasks of varying levels simultaneously, such as graph-level (graph classification), edge-level (link prediction), and node-level (node classification) tasks. However, handling all these tasks with a single pre-trained model is typically infeasible due to the limitations of both *dataset perspective* and *model architecture perspective.*

From the **dataset perspective**, node classification datasets like Cora and Coauthor typically consist of a single graph with labeled nodes, making it impractical to conduct graph classification tasks on such datasets. Additionally, performing edge-level tasks like link prediction using pre-trained models poses additional challenges, as it requires removing a portion of the edges and using them as a test set, necessitating the pre-training of another model on a modified dataset with the removed edges. It is critical to remove a portion of the edges (i.e., split the dataset) in a manner that ensures balanced data for training and testing. However, determining a reasonable way to split the data remains a significant challenge, particularly since none of the methods we reviewed incorporate edge-level tasks for evaluation.

From the **model structure perspective**, most GCL models designed for node classification are inherently unsuited for graph classification tasks, as they are primarily intended for learning node representations. The majority of the methods we reviewed lack a dedicated pooling module, making them nativly unsuitable for direct evaluation on graph-level tasks like graph classification.

Therefore, we propose simulating multiple downstream tasks by framing a multi-label classification task as a series of independent binary classification tasks, which we detail in the next section.

5. THE IMPROVED EVALUATION PROTOCOL

To address the aforementioned issues, we introduce an improved protocol for a more comprehensive evaluation. This protocol is designed to address the limitations of hyper-parameter sensitivity and the narrow focus on single downstream tasks. The improved evaluation framework consists of the following components.

- **Comprehensive Analysis Across Diverse Hyper-parameter Configurations.** We advocate for evaluating GCL methods based on their performance across a variety of hyper-parameter configurations. This approach includes analyzing both the average performance and the variability (or variance) of performances. By considering the variance, we gain insights into the stability of the GCL methods under different hyper-parameter settings, providing a more holistic view of their effectiveness.

- **Extension to Multi-label Dataset Evaluation.** Our protocol expands the scope of evaluation to include multi-label datasets, which allows us to simulate an environment with multiple downstream tasks. In particular, conducting a multi-label classification task can be regarded as solving a set of K binary classification tasks with K denoting the number of labels in the multi-label classification task. Evaluating GCL methods in this context offers a richer perspective on their capacity to generalize and adapt to diverse downstream tasks.

General Settings. Our evaluation encompasses both multi-class and multi-label datasets to ensure a comprehensive analysis of GCL methods. For multi-class, accuracy serves as the primary evaluation metric, offering a straightforward measure of a model's predictive performance. To address the inherent variability and sensitivity of GCL methods to hyper-parameter selection, we undertake an extensive analysis based on the average and variance of performance metrics across 20 randomly sampled sets of hyper-parameters. For multi-label, our evaluation framework employs both Micro-F1 and Macro-F1 scores as metrics as introduced in Section 3. Adhering to a similar methodology as applied in the multi-class dataset evaluation, we also derive insights from the average and variance of these two metrics across 20 randomly sampled sets of hyper-parameters for each GCL method.

5.1 Evaluation with Multi-class Classification

We conduct evaluation procedures for each GCL method on all 7 multi-class node classification datasets introduced in Section 3.1. With our proposed evaluation protocol, we aim to assess the general performance and stability of each method on different datasets. Due to space limit, we aggregate the results of all the datasets into Figure 6 by averaging them *i.e.,* the x and y axis indicates the average values of the method's accuracy and the standard deviation across 7 datasets, respectively. The performance here specifically denotes accuracy as detailed in Section 3.2. Here, a higher mean performance signifies superior effectiveness, whereas a lower mean standard deviation points to enhanced stability. Notably, in Figure 6, COSTA has been excluded due to its notable instability, which is consistent with previously reported findings [39; 38]. Barlow and SUGRL stand out by achieving the highest rankings in both average performance and stability metrics, showing their effectiveness and robustness. This finding aligns with the effectiveness reported in their respective papers. Surprisingly, DGI demonstrates a more competitive performance than might be anticipated, considering it as an earlier work often reported to underperform in other Graph Contrastive Learning (GCL) methods. Next, we delved deeper into the discrepancies between our results and those previously published in the literature [38; 39]. SFA [39] reported that SFA consistently surpassed GRACE and Barlow when utilizing a specific set of hyper-parameters, with SUGRL classified as a moderately performing method. In Section *4.1.2*, SFA is also observed with comparably good results similar to GRACE. Also, COSTA [38] was often highlighted as achieving near-

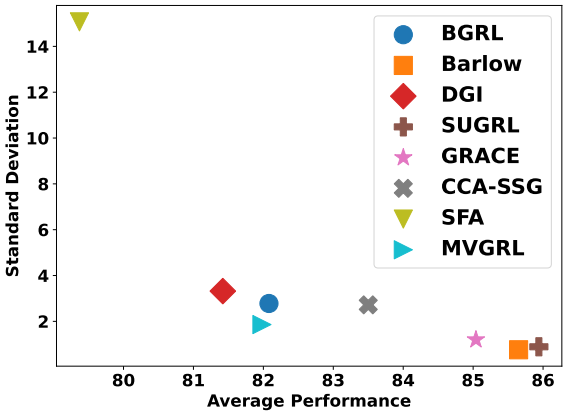
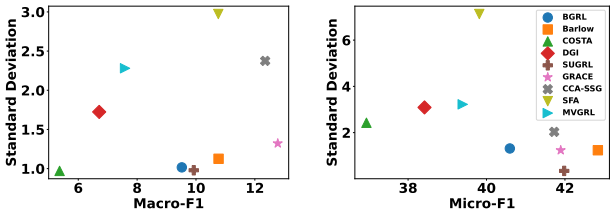


Figure 6: Results on multi-class classification datasets.

top performance, normally just behind SFA. This is also validated by the observation that COSTA performs relatively well in our results shown in Section *4.1.2*. However, when evaluating across our two aforementioned metrics, GRACE and Barlow not only outperform SFA and COSTA, showing superior effectiveness, but SUGRL also emerges as a top performer. Here, SFA's evaluation was based on dataset-specific hyper-parameters according to [39], while hyper-parameters for other baselines follow same ones in PyGCL[42]. Those new findings in our results demonstrate the advantage of our proposed evaluation protocol, which is designed to measure the overall performance and stability of GCL methods in a more realistic way. Overall, our new protocol enables a more comprehensive and robust understanding of a GCL method's capabilities.

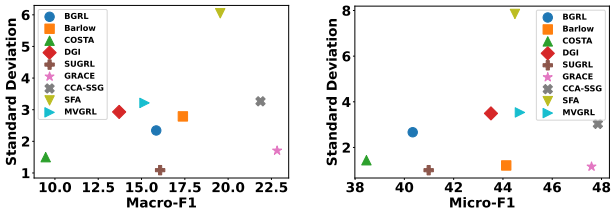
5.2 Evaluation with Multi-label Classification

Following our discussion on the outcomes of multi-class classification, we now shift our focus to the evaluation results obtained from multi-label classification tasks. We conduct the experiments on three datasets Eukloc, Humloc, and PCG, and their corresponding results are presented in Figures 7, 8, and 9, respectively.



(a) Macro F1 on Eukloc (b) Micro F1 on Eukloc

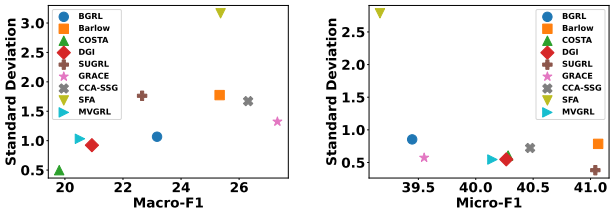
Figure 7: Results on Eukloc



(a) Macro F1 on Humloc (b) Micro F1 on Humloc

Figure 8: Results on Humloc

We make some general observations from these figures. Initially, we will discuss findings aligning with established benchmarks in multi-class classification, followed by an exploration of divergent results observed in multi-label classification. In multi-label classification, GRACE and Barlow maintain their strong performance and stability, underscoring their robust and consistent capabilities. DGI and BGRL also show a consistent pattern, displaying good stability and moderate performance in both classification contexts. Contrasting the two-fold consistencies mentioned in the prior models, the remaining models exhibit a similar pattern in



(a) Macro F1 on PCG (b) Micro F1 on PCG

Figure 9: Results on PCG

only one aspect: SUGRL demonstrates notable strength in performance across both evaluation settings, while SFA consistently keeps relatively large standard deviations across all evaluations. Also, CCA-SSG maintains a trend of moderately above-average performance.

Apart from consistent results, we also observe notable shifts in model performance or overall stability. SFA, previously underperforming in multi-class classification, shows improved results in the current multi-label classification tasks, ranking above average. Conversely, SUGRL, while dominating in multi-class classification, falls to moderate performance levels in the new tasks. In addition, MVGRL sees a slight increase in standard deviation, indicating a change in stability from lower to moderate levels in the multi-label classification. Moreover, CCA-SSG's stability decreases in multi-label classification, with some datasets like the Eukloc showing a notably higher standard deviation in the Macro F1 assessment.

The observed consistencies across evaluation methods suggest a good level of agreement, indicating that both evaluation methods can, to a certain extent, reflect the overall performance and stability of GCL methods. On the other hand, new observations in multi-label classification tasks highlight the value of incorporating this new multi-label classification evaluation approach, which broadens the evaluation scope and reveals deeper insights into the performance of GCL methods.

6. CONCLUSION

In this paper, we illustrate the intrinsic limitations of the existing evaluation protocol of GCL methods, highlighting its divergence from the overarching goals of self-supervised learning. We approach this assessment from two angles: the numerous hyper-parameters during the pre-training phase and the sole evaluation based on a single downstream task. We verify the prominent issues by investigating the sensitivities of GCL methods to hyper-parameters during the pre-training stage and the representativeness of the downstream task. Moreover, to rectify the aforementioned issues, we propose an improved evaluation protocol for better benchmarking to comprehensively evaluate the GCL methods.

7. ACKNOWLEDGEMENTS

The research is supported by the National Science Foundation (NSF) under grant numbers NSF2406647 and NSF-2406648.

8. REFERENCES

[1] Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *Knowledge-Based Systems*, 256:109631, 2022.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[3] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. Graph transfer learning via adversarial domain adaptation with graph convolution. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4908–4922, 2022.

[4] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

[5] Wenqi Fan, Yao Ma, Dawei Yin, Jianping Wang, Jiliang Tang, and Qing Li. Deep social collaborative filtering. In *Proceedings of the 13th ACM RecSys*, 2019.

[6] Taoran Fang, Yunchao Zhang, Yang Yang, Chunping Wang, and Lei Chen. Universal prompt tuning for graph neural networks. *Advances in Neural Information Processing Systems*, 36:52464–52489, 2023.

[7] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.

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

[9] Dmitri Goldenberg. Social network analysis: From graph theory to applications with python. *arXiv preprint arXiv:2102.10014*, 2021.

[10] William L Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.

[11] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pages 4116–4126. PMLR, 2020.

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[13] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30:595–608, 2016.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[16] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[18] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and S Yu Philip. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5879–5900, 2022.

[19] Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks. In *Proceedings of the ACM web conference 2023*, pages 417–428, 2023.

[20] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. *Advances in neural information processing systems*, 32, 2019.

[21] Qian Ma, Zijian Zhang, Xiangyu Zhao, Haoliang Li, Hongwei Zhao, Yiqi Wang, Zitao Liu, and Wanyu Wang. Rethinking sensors modeling: Hierarchical information enhanced traffic forecasting. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

[22] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7797–7805, 2022.

[23] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. *Advances in neural information processing systems*, 30, 2017.

[24] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

[25] Xiangguo Sun, Hong Cheng, Jia Li, Bo Liu, and Jihong Guan. All in one: Multi-task prompting for graph neural networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2120–2131, 2023.

[26] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.

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

[28] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *arXiv preprint arXiv:1809.10341*, 2018.

[29] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[30] Jun Xia, Lirong Wu, Jintao Chen, Bozhen Hu, and Stan Z Li. Simgrace: A simple framework for graph contrastive learning without data augmentation. In *Proceedings of the ACM web conference 2022*, pages 1070–1079, 2022.

[31] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[32] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[33] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 974–983, 2018.

[34] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *Advances in neural information processing systems*, 31, 2018.

[35] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[36] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. *Advances in Neural Information Processing Systems*, 34:76–89, 2021.

[37] Mengmei Zhang, Linmei Hu, Chuan Shi, and Xiao Wang. Adversarial label-flipping attack and defense for graph neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 791–800. IEEE, 2020.

[38] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Costa: covariance-preserving feature augmentation for graph contrastive learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2524–2534, 2022.

[39] Yifei Zhang, Hao Zhu, Zixing Song, Piotr Koniusz, and Irwin King. Spectral feature augmentation for graph contrastive learning and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 11289–11297, 2023.

[40] Tianqi Zhao, Ngan Thi Dong, Alan Hanjalic, and Megha Khosla. Multi-label node classification on graph-structured data. *arXiv preprint arXiv:2304.10398*, 2023.

[41] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*, 2021.

[42] Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*, 2021.

[43] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

[44] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021.

[45] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

APPENDIX

A. HYPER-PARAMETERS SEARCH SPACE

In our experiments, we performed a random hyper-parameter generation to examine the method's sensitivity to the hyper-parameters. The the search spaces are within typical range and details are outlined below:

- **Learning Rate**: The learning rate was searched over the values {0.01, 0.001, 0.0001, 0.00001}.
- **Drop Edge Rate**: The edge dropout rate was sampled from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Drop Feature Rate**: The node feature dropout rate was selected from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Hidden Dimension (`nhid`)**: The hidden dimension of the model was chosen from {128, 256, 512}.
- **Pnhid**: Hidden dimension for projection was searched over {128, 256, 512}.
- **Drop Edge Rate1 & Drop Edge Rate2**: The edge dropout rates for methods with two augmented view, each sampled dependently from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Drop Feature Rate1 & Drop Feature Rate2**: The feature dropout rates for methods with two augmented view, also sampled dependently from {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.
- **Loss Weight**: The loss weight factor of method SUGRL was searched over {5, 10, 20, 50, 100}.
- **NN**: The NN of method SUGRL indicating the number of node permutations was selected from {1, 3, 4, 5, 10}.
- **Margin1 & Margin2**: The margins used for Loss of Method SUGRL are dependently sampled from {0.4, 0.5, 0.8, 0.9} and {0.1, 0.2, 0.4, 0.5, 0.6, 0.9}, respectively.
- **K**: The K, indicating the number of iterations used in SFA k was chosen from {1, 2}.
- τ: The temperature of InfoNCE loss, is sampled over the range {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}.