OpenGSL: A Comprehensive Benchmark for Graph Structure Learning

Zhiyao Zhou[1], **Sheng Zhou**[2], **Bochao Mao**[3], **Xuanyi Zhou**[1], **Jiawei Chen**[1]
Qiaoyu Tan[4], **Daochen Zha**[5], **Can Wang**[1], **Yan Feng**[1], **Chun Chen**[1]
[1]College of Computer Science, Zhejiang University, Hangzhou, China
[2]School of Software Technology, Zhejiang University, Ningbo, China
[3]East China Normal University [4]New York University Shanghai [5]Rice University
{zjucszzy, zhousheng_zju, sleepyhunt, wcan, fengyan, chenc}@zju.edu.cn
10195102465@stu.ecnu.edu.cn qiaoyu.tan@nyu.edu daochen.zha@rice.edu

Abstract

Graph Neural Networks (GNNs) have emerged as the *de facto* standard for representation learning on graphs, owing to their ability to effectively integrate graph topology and node attributes. However, the inherent suboptimal nature of node connections, resulting from the complex and contingent formation process of graphs, presents significant challenges in modeling them effectively. To tackle this issue, Graph Structure Learning (GSL), a family of data-centric learning approaches, has garnered substantial attention in recent years. The core concept behind GSL is to jointly optimize the graph structure and the corresponding GNN models. Despite the proposal of numerous GSL methods, the progress in this field remains unclear due to inconsistent experimental protocols, including variations in datasets, data processing techniques, and splitting strategies. In this paper, we introduce OpenGSL, the first comprehensive benchmark for GSL, aimed at addressing this gap. OpenGSL enables a fair comparison among state-of-the-art GSL methods by evaluating them across various popular datasets using uniform data processing and splitting strategies. Through extensive experiments, we observe that existing GSL methods do not consistently outperform vanilla GNN counterparts. However, we do observe that the learned graph structure demonstrates a strong generalization ability across different GNN backbones, despite its high computational and space requirements. We hope that our open-sourced library will facilitate rapid and equitable evaluation and inspire further innovative research in the field of GSL. The code of the benchmark can be found in `https://github.com/OpenGSL/OpenGSL`.

1 Introduction

Graph Neural Networks (GNNs) [4, 13, 8, 32] have emerged as the dominant approach for learning on graph-structured data, thanks to their exceptional ability to leverage both the graph topology structure and node attributes [35]. Considerable endeavors have been recently made to enhance the performance of GNNs by refining the architectures of GNN models, such as neural message passing [32, 37, 36, 7, 2] and transformer based methods [14, 39, 28]. However, these *model-centric* methods overlook the potential flaws of the underlying graph structure, which can result in suboptimal performance. Notably, extensive evidence from previous studies [24] confirms that real-world

Submitted to the 37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks. Do not distribute.

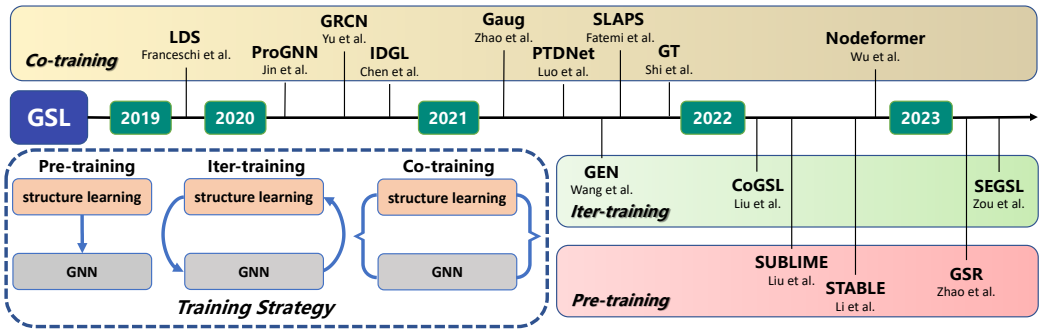

Figure 1: Timeline of GSL research. Bottom left corner: illustrations of the three training strategies.

graphs often exhibit subpar characteristics, such as the absence of valuable links and the presence of spurious connections among nodes.

To improve the graph quality, Graph Structure Learning (GSL) [48], a family of *data-centric* graph learning methods [42, 41, 23], has recently attracted considerable research interest. These methods target optimizing the graph structure and the corresponding GNN representations jointly [19, 16, 11, 5, 33]. By refining the graph structure, GSL methods can potentially empower GNNs to learn better representations with improved performance. GSL has been successfully applied to disease analysis [3] and protein structure prediction [12].

Despite the plethora of GSL methods proposed in recent years, as illustrated in Figure 1, there is no comprehensive benchmark for GSL, which significantly impedes the understanding and progress of GSL in several aspects. *i)* The use of different datasets, data processing approaches, data splitting strategies, and GNN backbones in the previous work makes many of the results incomparable. *ii)* There is a lack of understanding of the learned structure itself, particularly regarding its homophily and generalizability to other GNN backbones. *iii)* Apart from accuracy, understanding each method's computation and memory costs is imperative, yet often overlooked in the literature.

To bridge this gap, we introduce OpenGSL, the first comprehensive benchmark for GSL. OpenGSL implements a wide range of GSL algorithms through unified APIs, while also adopting consistent data processing and data splitting approaches for fair comparisons. Through benchmarking the existing GSL methods on various datasets, we make the following contributions:

- **Comprehensive benchmark.** OpenGSL enables a fair comparison among twelve state-of-the-art GSL methods by unifying the experimental settings across ten popular datasets of diverse types and scales. Surprisingly, the empirical results reveal that GSL methods do not consistently outperform the vanilla GNNs.
- **Multi-dimensional analysis.** We conduct a systematic analysis of GSL methods from various dimensions, encompassing the homophily of the learned structure, the generalizability of the learned structure across GNN backbones, and the time and memory efficiency of the existing methods. **Our key findings:** *i)* Contrary to the common belief in the homophily assumption, increasing the homophily of the structure does not necessarily translate into improved performance. *ii)* The learned structures by GSL methods exhibit strong generalizability. *iii)* Most GSL methods are time- and memory-inefficient, some of which require orders of magnitudes more resources than vanilla GNNs, highlighting the pressing need for more efficient GSL approaches.
- **Open-sourced benchmark library and future directions**: We have made our benchmark library publicly available on GitHub, aiming to facilitate future research endeavors. We have also outlined potential future directions based on our benchmark findings to inspire further investigations.

To summarize, in this paper, we aim to create a comprehensive benchmark that facilitates the regular evaluation of graph structure learning algorithms, encourages new research, and ultimately advances the efficacy of the field as a whole.

2 Formulations and Background

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be a graph, where $\mathcal{V}$ is the set of N nodes and $\mathbf{A} \in R^{N \times N}$ is the adjacent matrix. $\mathcal{E}$ denotes the edge set and $\mathbf{X} \in \mathcal{R}^{N \times d}$ represents the corresponding feature matrix with dimension d. Typically, a GNN model is often parameterized by a mapping function $f : (\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{H} \in \mathcal{R}^{N \times l}$, which maps each node $v \in \mathcal{V}$ into a l-dimensional embedding vector $\mathbf{h}_v$. In the semi-supervised setting, some nodes $v_i \in \mathcal{V}_{train}$ are often associated with labels y_i to guide the GNNs training. Please note that in the traditional GNNs, the adjacent matrix $\mathbf{A}$ only serves as input and is not updated along with the training of GNNs. In the GSL methods, a gradually updated structure $\mathbf{S} \in \mathcal{R}^{N \times N}$ is learned to replace the original structure $\mathbf{A}$ and optimize with respect to the training process. This is the major difference between traditional GNNs and GSL methods.

Timeline of GSL methods through the lens of training procedures. To provide a global understanding of the literature, we provide a high-level overview of the existing GSL methods based on their *training procedure*. Specifically, we identify two key components in GSL methods: structure learning component and GNN component. Based on the interaction between these two components, we partition existing GSL methods into three categories, depicted in bottom left corner of Figure 1: (a) pre-training, (b) co-training, and (c) iter-training. Pre-training involves a two-stage learning process, where the structure is learned through pre-training and then used to train GNNs in downstream tasks [19, 16, 43]. In co-training methods [6, 1, 21], neural networks that generate the graph structure are optimized together with GNNs. The iterative methods [33, 49, 31] involve training the two components iteratively; they learn the structure from predictions or representations generated by an optimized GNN and use it to train a new GNN model for the subsequent iteration.

Homophily and Heterophily. Homophily [24] and heterophily are two mutually exclusive measurement based on similarity between connected node pairs, where two nodes are considered similar if they share the same node label. The homophily of graph $\mathcal{G}$ can be formulated as:

$$\text{homo}(\mathcal{G}) = \frac{1}{|\mathcal{E}|} |\{(v_i, v_j) | (v_i, v_j) \in \mathcal{E}, y_i = y_j\}| \tag{1}$$

where $|\mathcal{E}|$ is the number of observed edges. Correspondingly, the heterophily of graph $\mathcal{G}$ is defined as $1 - \text{homo}(\mathcal{G})$. The homophily of graphs have been widely assumed to be key motivation of designing GNNs, although a few recent works [22, 20] have argued on it. Although some GSL methods [44, 33] also claim to learn a graph structure with high homophily, the homophily of the graph structure learned by GSL methods have not been studied.

3 Benchmark Design

We begin by introducing the datasets utilized in our benchmarking process, along with the algorithm implementations. Then, we outline the research questions that guide our benchmarking study.

3.1 Datasets and Implementations

Datasets. In order to provide a comprehensive evaluation of existing GSL methods, we collect 10 graph datasets that have been widely used in the GSL literature. These selected datasets come from different domains and exhibit different characteristics, enabling us to evaluate the generalizability of existing methods across a range of scenarios. Specifically, we use three classic citation datasets [29], namely Cora, Citeseer, Pubmed, as well as two representative social network datasets BlogCatalog and Flickr [10]. Additionally, we include five datasets that have been proposed recently in [27], due to their ability to overcome the drawbacks of the commonly used heterophilous datasets[25]. Table 1 shows the statistics of these datasets, which are divided into two groups according to whether their edge homophily are higher than 0.5.

The data splitting methods in different GSL works are not consistent, bringing difficulties in conducting fair comparisons. We investigate various GSL works and choose the data splits that are most commonly used. For three citation datasets, we use the classic split from [38, 13]. For BlogCatalog

Table 1: Overview of the datasets used in this study.

group	Dataset	# Nodes	# Edges	# Feat.	Avg. # degree	# Classes	# Homophily
homophilous	Cora	2,708	5,278	1,433	3.9	7	0.81
	Citeseer	3,327	4,552	3,703	2.7	6	0.74
	Pubmed	19,717	44,324	500	4.5	3	0.80
	Questions	48,921	153,540	301	6.3	2	0.84
	Minesweeper	10,000	39,402	7	7.9	2	0.68
heterophilous	BlogCatalog	5,196	171,743	8,189	66.1	6	0.40
	Flickr	7,575	239,738	12,047	63.3	9	0.24
	Amazon-Ratings	24,492	93,050	300	7.6	5	0.38
	Roman-Empire	22,662	32,927	300	2.9	18	0.05
	Wiki-cooc	10,000	2,243,042	100	448.6	5	0.34

and Flickr, we follow the split in [10, 44]. For five heterophilous datasets, we follow the original split in [27]. More details about these datasets can be found in the supplemental materials.

Implementations. We consider a collection of state-of-the-art algorithms, including ProGNN [11], IDGL [1], GRCN [40], GAug [44], SLAPS [5], GEN [33], GT [30], Nodeformer [34], CoGSL [17], SUBLIME [19], STABLE [16], and SEGSL [49]. We rigorously reproduced all methods according to their papers and source codes. To ensure a fair evaluation, we perform hyperparameter tuning with the same search budget on the same dataset for all methods. More details on these algorithms and implementations can be found in the supplemental materials and the GitHub repository.

3.2 Research Questions

We carefully design the OpenGSL to systematically evaluate existing methods and inspire future research. Specifically, we aim to answer the following research questions.

RQ1: How much progress has been made by existing GSL methods?

Motivation. Previous research in GSL has been hindered by the use of different data preprocessing, and splits, which make it difficult to fairly evaluate and compare the performance of different methods. Given the fair comparison environment provided by OpenGSL, the first research question is to revisit how much progress has been made by existing GSL methods. By answering this question, we aim to gain a deeper understanding of the strengths and weaknesses of existing methods, and identify areas that offer potential for further enhancements.

Experiment Design. Following the experimental setting of existing GSL methods, we conduct the node classification experiments on all the datasets. For each method and dataset, we report the mean performance and standard deviation of 10 runs. For binary classification datasets, we report ROC AUC metric, while for other datasets, we report accuracy metric. Since most of the implemented GSL methods have used GCN as backbone, we compare the performances of GSL methods with vanilla GCN to verify the enhancement of learning structures.

RQ2: Does GSL benefit from learning graph structures with higher homophily?

Motivation. The homophily assumption has been a fundamental motivation of modern GNNs designs, which has also been brought to the GSL scenarios. More specifically, some existing GSL methods have attempted to learn the structure with higher homophily by introducing explicit homophily-oriented objectives [44, 33]. However, the validity of these claims is questionable since they have been evaluated only on a limited set of toy datasets [33], with little examination of their generalizability. As researchers have started to question the homophily assumption on GNNs [22], it becomes imperative to re-evaluate the significance of GSL methods in learning more homophilous graph structures.

Experiment Design. To answer this question, we first compare the homophily of the structure learned by GSL methods with the original one. Then we determine whether the reported performance

improvements stem from a more homophilous graph structure by examining the correlation between the homophily and node classification performance.

RQ3: Can the learned structures generalize to other GNN backbones?

Motivation. Although node classification tasks have been a popular means of evaluating GNNs optimized by GSL methods, the quality of the learned graph structure has not been thoroughly evaluated. It can be assumed that the learned graph structure is superior to the original one, which could potentially benefit other GNN models. Therefore, it is crucial to conduct experiments using both task-dependent and task-agnostic methods to evaluate the generalizability and expand the impact of the learned graph structure.

Experiment Design. To answer this research question, we use the learned structure and original features to create a new graph data $\mathcal{G}' = (\mathbf{S}, \mathbf{X})$, and train a new GNN method from scratch. By comparing the performance of the GNNs on the original graph and the new graph, we can evaluate the generalizability of the learned structure. To further verify the effectiveness of learned structure, we also include two non-GNN methods, Label Propagation [47, 46] and LINK [45], that only take graph structure as input for node classification.

RQ4: Are existing GSL methods efficient in terms of time and space?

Motivation. As the GSL methods simultaneously optimize the GNNs and graph structure, they naturally consume more computational complexity and space than the GNNs. However, the efficiency of GSL methods have been largely overlooked by existing methods. Although introducing the structure learning may benefit the GNNs, the extra computational consumption has posed significant requirements of trade-off between performance and efficiency. It is critical to understand the trade-off for deploying the GSL in the practical applications.

Experiment Design. To answer this research question, we evaluate the efficiency of each GSL method in terms of time and space. Specifically, we record the wall clock running time and peak GPU memory consumption during each method's training process. For a fair comparison, all experiments in this part were conducted on a single NVIDIA A800 GPU.

4 Experiment Results and Analyses

4.1 Performance Comparison (RQ1)

We present the performance of all methods on 10 datasets in Table 2 and Table 3. Below are the key findings from these tables.

① **Many GSL methods works on homophilous graphs, while most methods cannot handle imbalanced situations.** Table 2 demonstrates that most GSL methods outperform vanilla GCN on balanced homophilous graphs such as Cora, Citeseer, and Pubmed. Out of the 12 methods tested, 7 methods exceeded GCN on at least two datasets, and 9 methods outperformed GCN on at least one dataset. However, some methods were unable to surpass vanilla GCN, suggesting that structure learning might even degrade GNN performance. This result highlights the need to investigate the general effectiveness of GSL methods further. In contrast, the results were vastly different on datasets such as Questions and Minesweeper, where only a few methods showed an advantage over GCN. The imbalanced nature of these datasets limited the power of GSL, indicating that their effectiveness is constrained on such types of data. This fact suggests that as many real-world graphs are imbalanced, evaluating the effectiveness of GSLs on imbalanced datasets should receive more attention in the future research.

② **GSL methods can also be effective on heterophilous graphs.** Table 3 reveals that some GSL methods, including IDGL, GAug, GEN, and SUBLIME, have the potential to exceed vanilla GCN on heterophilous graphs such as BlogCatalog, Flickr, and Amazon-ratings. This finding is intriguing because homophily is not well-preserved on these graphs, inspiring us to investigate the correlation between homophily and structure learning. However, the results were different on Roman-empire

Table 2: Node classification results on Cora, Citeseer, Pubmed, Questions and Minesweeper. Shown is the mean ± s.d. of 10 runs with different random seeds. Highlighted are the top **first**, **second**, and **third** results. – denotes out of memory or time limit exceeded.

Strategy	Model	Cora	Citeseer	Pubmed	Questions	Minesweeper
–	GCN	81.95 ± 0.62	71.34 ± 0.48	78.98 ± 0.35	**75.80 ± 0.51**	**78.28 ± 0.44**
Co-training	ProGNN	80.27 ± 0.48	71.35 ± 0.42	79.39 ± 0.29	–	51.43 ± 2.22
Co-training	IDGL	**84.19 ± 0.61**	**73.26 ± 0.53**	**82.78 ± 0.44**	50.00 ± 0.00	50.00 ± 0.00
Co-training	GRCN	**84.61 ± 0.34**	72.34 ± 0.73	79.30 ± 0.34	**74.50 ± 0.84**	72.57 ± 0.49
Co-training	GAug(O)	**83.43 ± 0.53**	72.79 ± 0.86	78.73 ± 0.77	–	**77.93 ± 0.64**
Co-training	SLAPS	72.29 ± 1.01	70.00 ± 1.29	70.96 ± 0.99	–	50.89 ± 1.72
Co-training	GT	80.79 ± 1.14	68.50 ± 2.07	77.91 ± 0.58	**75.36 ± 0.92**	**89.70 ± 0.28**
Co-training	Nodeformer	78.81 ± 1.21	70.39 ± 2.04	78.38 ± 1.94	72.61 ± 2.29	77.29 ± 1.71
Iter-training	GEN	81.66 ± 0.91	**73.21 ± 0.62**	78.49 ± 3.98	–	79.56 ± 1.09
Iter-training	CoGSL	81.46 ± 0.88	**72.94 ± 0.71**	–	–	–
Iter-training	SEGSL	81.04 ± 1.07	71.57 ± 0.40	79.26 ± 0.67	–	–
Pre-training	SUBLIME	83.33 ± 0.73	72.44 ± 0.89	**80.56 ± 1.32**	67.21 ± 0.99	49.93 ± 1.36
Pre-training	STABLE	83.25 ± 0.86	70.99 ± 1.19	**81.46 ± 0.78**	–	70.78 ± 0.27

Table 3: Node classification results on BlogCatalog, Flickr, Amazon-ratings, Roman-empire and Wiki-cooc. Shown is the mean ± s.d. of 10 runs with different random seeds. Highlighted are the top **first**, **second**, and **third** results. – denotes out of memory or time limit exceeded.

Strategy	Model	BlogCatalog	Flickr	Amazon-ratings	Roman-empire	Wiki-cooc
–	GCN	76.12 ± 0.42	61.60 ± 0.49	45.24 ± 0.29	**70.41 ± 0.47**	**92.03 ± 0.19**
Co-training	ProGNN	73.38 ± 0.30	52.88 ± 0.76	–	56.21 ± 0.58	89.07 ± 5.59
Co-training	IDGL	89.68 ± 0.24	**86.03 ± 0.25**	45.87 ± 0.58	47.10 ± 0.65	90.18 ± 0.27
Co-training	GRCN	76.08 ± 0.27	59.31 ± 0.46	**50.06 ± 0.38**	44.41 ± 0.41	90.59 ± 0.37
Co-training	GAug(O)	76.92 ± 0.34	61.98 ± 0.67	48.42 ± 0.39	52.74 ± 0.48	**91.30 ± 0.23**
Co-training	SLAPS	**91.73 ± 0.40**	83.92 ± 0.63	40.97 ± 0.45	**65.35 ± 0.45**	89.09 ± 0.54
Co-training	GT	70.70 ± 5.62	43.19 ± 6.53	**48.55 ± 0.34**	**76.49 ± 0.80**	90.26 ± 1.24
Co-training	Nodeformer	44.53 ± 22.62	67.14 ± 6.77	41.33 ± 1.25	56.54 ± 3.73	54.83 ± 4.43
Iter-training	GEN	**90.48 ± 0.99**	**84.84 ± 0.81**	**49.17 ± 0.68**	–	**91.15 ± 0.49**
Iter-training	CoGSL	–	–	–	–	–
Iter-training	SeGSL	75.03 ± 0.28	60.59 ± 0.54	–	–	–
Pre-training	SUBLIME	**95.29 ± 0.26**	**88.74 ± 0.29**	44.49 ± 0.30	63.93 ± 0.27	76.10 ± 1.12
Pre-training	STABLE	71.84 ± 0.56	51.36 ± 1.24	48.36 ± 0.21	41.00 ± 1.18	80.46 ± 2.44

and Wiki-cooc datasets, where only a few methods demonstrated an improvement over GCN. This observation suggests that the heterophilous datasets may have informative structural patterns that current GSL methods targeting homophily undermine. For more in-depth analysis, please refer to section 4.2.

4.2 Correlation between Homophily and Structure Learning (RQ2)

To analyze the correlation between homophily and structure learning, we plot the homophily of the learned structure and the node classification performance in the same figure. Figure 2 and Figure 3 show the results, from which we have the following observations:

③ **The homophily of the learned structures varies on homophilous and heterophilous datasets.** The results presented in Figure 2 are intriguing. They demonstrate that on homophilous datasets, the homophily of the learned structures is scarcely different from the original structure, and in some cases, even lower. This is counterintuitive, considering that methods like GEN explicitly aim to increase the homophily of the learned structures. However, on heterophilous datasets in Figure 3, the homophily of the learned structures is significantly improved in most cases, possibly due to the starting level of graph homophily. Because most GSL methods are trained with limited supervision signals, the number of edges that can be recovered or removed by GSL is limited. Thus, on heterophilous datasets where most edges do not fit the homophily assumption, these limited edges are more likely to be updated. On the other hand, they are already satisfied on homophilous datasets, explaining the lack of improvement there.

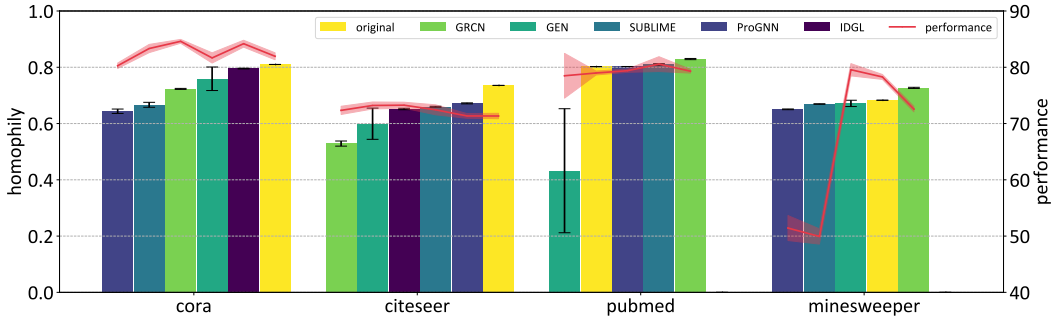

Figure 2: Homophily of learned structures and performances on homophilous datasets. The methods are ordered by homophily.

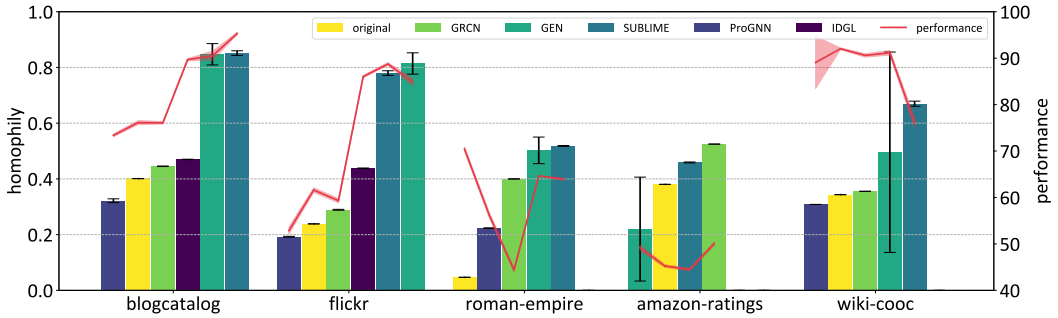

Figure 3: Homophily of learned structures and performances on heterophilous datasets. The methods are ordered by homophily.

④ **Homophily is not a proper guidance for structure learning.** As shown in both Figure 2 and Figure 3, homophily is only significantly positively correlated with performance on two datasets, i.e., BlogCatalog and Flickr. In most cases, we do not observe positive correlation between the performance and the homophily. These findings suggest that homophily of structure is not a proper guidance for GSL in all cases, which may break the common assumption and expect novel learning objective. This can be somhow explained by the viewpoint that certain heterophilous structural patterns can be leveraged by GNNs [22, 20], thereby guiding structure learning with homophily may break these patterns and leading to suboptimal results.

4.3 Generalizability (RQ3)

We show the performance of several GNN models and simple non-GNN models using the structures learned by GSL methods as inputs in Table 4 and 5, from which we have the following observations:

⑤ **The structures learned by GSL methods exhibit strong generalizability.** The results in Table 4 and 5 demonstrate that many GNN backbones show improved performance on the learned structure, marked in green, as compared to the original structure. This observation underscores the generalizability of the learned structure and its potential to enhance numerous GNN methods. Additionally, we observed that the structures learned by GSL methods also help to improve the performance of two simple non-GNN methods - LPA and LINK - and in some instances, they even outperform GNNs. The promising results strengthen the notion of the learned structure's generalizability, even without considering node features as input. In conclusion, the experimental results provide strong evidence for the generalizability of the learned structure and call for further exploration and applications of GSL methods.

Table 4: Generalizability on Cora. Improvements over the original structure are marked green.

Structure source	GCN	SGC	JKNet	APPNP	GPRGNN	LPA	LINK
A	81.95 ± 0.62	80.00 ± 0.14	80.40 ± 1.13	83.33 ± 0.33	83.51 ± 0.67	60.30 ± 0.00	50.06 ± 2.46
ProGNN	82.58 ± 0.47	82.10 ± 0.44	82.24 ± 0.85	83.38 ± 0.57	83.36 ± 0.40	75.85 ± 0.39	78.64 ± 0.22
IDGL	83.01 ± 0.56	83.44 ± 0.27	82.22 ± 0.59	84.34 ± 0.61	84.76 ± 0.66	77.31 ± 0.58	75.27 ± 1.66
GRCN	83.98 ± 0.35	84.66 ± 0.33	84.09 ± 0.46	83.35 ± 0.47	84.41 ± 0.81	79.03 ± 0.56	81.83 ± 0.24
GEN	81.74 ± 0.97	81.82 ± 1.27	81.92 ± 0.77	82.21 ± 0.97	82.16 ± 0.93	80.97 ± 1.47	79.12 ± 3.31
SUBLIME	82.69 ± 0.49	82.32 ± 0.48	81.98 ± 0.65	82.79 ± 0.61	83.59 ± 0.76	78.42 ± 1.08	81.25 ± 1.23

Table 5: Generalizability on BlogCatalog. Improvements over the original structure are marked green.

Structure source	GCN	SGC	JKNet	APPNP	GPRGNN	LPA	LINK
A	76.12 ± 0.42	75.37 ± 0.08	74.17 ± 0.42	93.72 ± 0.10	93.82 ± 0.16	57.53 ± 0.00	64.47 ± 0.30
ProGNN	71.73 ± 0.88	75.20 ± 0.12	73.73 ± 0.69	93.71 ± 0.27	94.70 ± 0.26	53.66 ± 0.12	66.00 ± 0.79
IDGL	89.35 ± 0.23	75.20 ± 0.12	89.77 ± 0.32	94.90 ± 0.17	95.39 ± 0.17	73.84 ± 0.30	81.25 ± 0.17
GRCN	75.69 ± 0.34	74.77 ± 0.29	75.62 ± 0.28	93.45 ± 0.14	93.53 ± 0.24	59.68 ± 0.24	66.76 ± 0.54
GEN	90.01 ± 1.62	89.31 ± 2.97	90.23 ± 1.09	90.37 ± 1.16	90.40 ± 1.06	86.04 ± 9.31	86.29 ± 8.73
SUBLIME	95.01 ± 0.32	94.95 ± 0.34	73.73 ± 0.69	95.35 ± 0.20	94.51 ± 0.36	89.04 ± 1.09	86.45 ± 1.42

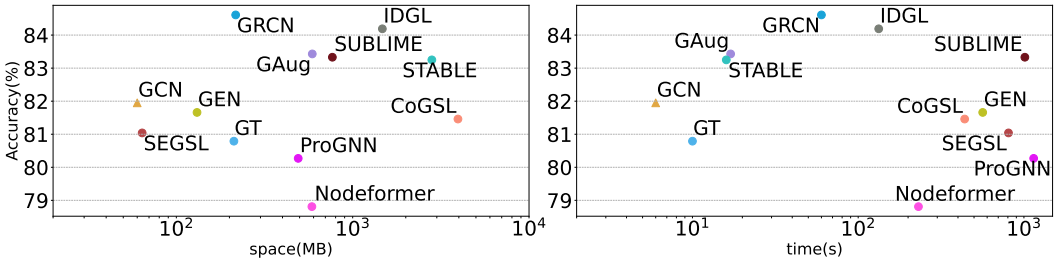

Figure 4: Time and space consumption of different methods on Cora

4.4 Time and Memory Efficiency (RQ4)

The efficiency of all methods on Cora are presented in Figure 4. For complete statistics on other datasets, please refer to the supplemental materials.

⑥ **Most GSL methods have large time and space consumptions.** The Figure 4 clearly demonstrate that the current state-of-the-art GSL methods is struggling to achieve a satisfactory balance between performance and efficiency. In particular, most existing GSL methods suffer from significant efficiency issues, with many taking up to ten times longer to run than the GCN method. According to the results, ProGNN is the slowest, requiring 190 times longer than GCN. Likewise, most GSL methods consume an excessive amount of memory, with CoGSL consuming up to 66 times more. The efficiency problem of GSL methods is especially pronounced on larger datasets, as discussed in the Supplemental Materials. According to Table 2, several GSL methods run out of memory when performing graph structure learning on the Questions dataset. Given these findings, it is vital to address the efficiency problem to ensure that GSL methods can be deployed successfully in a wide spectrum of real-world scenarios.

5 Future Direrctions

Drawing upon our empirical analyses, we point out some promising future directions for GSL and discuss future work for the refinement of OpenGSL.

Rethinking the necessity of homophily in GSL. While current GSL methods commonly pursue homophily within the refined graph structure, observations ③ and ④ suggest that performance enhancements in GSL do not necessarily originate from increased homophily. Consequently, it is essential to rethink the necessity of homophily in GSL and to explore alternative factors contributing to the effectiveness of GSL.

Designing adaptive GSL methods for diverse datasets. Observations ① and ② indicate that current GSL method do not universally work well across diverse datasets. Thus, there is an evident opportunity for creating innovative GSL method that adapt effectively to diverse datasets. In pursuit of this goal, two critical questions arise: 1) What characteristics should learned structures exhibit for disparate datasets? Our observation ④ highlights that a sole focus on homophily may not lead to substantial advancements in certain datasets, suggesting the need to explore more crucial properties. 2) How can we incorporate these characteristics into the structure learning? Some properties might be hard to evaluate or optimize, hence surrogate evaluators warrant further investigation. By answering these questions, we can develop more effective and efficient GSL methods that are better suited for a variety of real-world applications.

Developing task-agnostic GSL methods. Existing research efforts on GSL are mainly task-motivated. However, real-world scenarios often necessitate the refinement of a graph structure without accessing the downstream task. Although there are a few preliminary work on task-agnostic GSL [19, 16, 43], they exhibit certain limtations on preserving the naive characteristics such as proximity which is not sufficient in downstream tasks like graph clustering. The core challenge is to extract semantic information from the graph data and to define optimality of the structure in the absence of explicit labels.

Improving the efficiency of GSL methods. Observation ⑦ exposes the issue of efficiency in GSL, requiring further attention in GSL research. The practical utility of current GSL methodologies is often hampered by these efficiency issues. Although some attempts have been made to address this problem, they commonly compromise the expressiveness of GSL. These compromises include limiting the weights on pre-existing edges [5] or employing anchor points [1]. Drawing inspiration from the successful adoption of sampling strategies in Graph Neural Networks (GNNs) for acceleration [18], it would be promising to devise sophisticated sampling methodologies specifically tailored for GSL.

6 Conclusion and Future Work

This paper introduces the first comprehensive graph structure learning (GSL) benchmark, OpenGSL, by applying and comparing twelve cutting-edge methods across ten renowned datasets covering varying types and scopes. This unbiased comparison and comprehensive analysis unearthed several key findings about this promising research topic. Firstly, we reevaluated the correlation between homophily and structure learning, recommending innovative learning objectives. Second, we established the robustness and broad applications of the acquired structure. Lastly, we highlighted the efficiency predicaments of the existing methods, emphasizing the need for essential attention. We believe that this benchmark will have an extensive positive impact on this emerging research domain. We have released the code to the academic community and welcome further contributions of diverse datasets and ground-breaking methods.

Despite conducting a comprehensive benchmark for GSL methods, OpenGSL plans to further enhance its benchmarking process. We intend to achieve this goal by incorporating several improvements in the following ways. To begin with, we plan to widen our range of datasets to establish a more impartial comparison. Specifically, we aim to cater to heterogeneous information networks that are used extensively in practical applications. Secondly, in view of the efficiency limitations associated with current GSL methods, we are looking to integrate structure learning on large-scale datasets such as OGB [9]. Thirdly and more importantly, we believe that it is crucial to provide universal solutions such as subgraph sampling to develop new and more efficient GSL methods. Lastly, we intend to expand OpenGSL's support to other graph learning tasks as most of the current research only focuses on node classification. We will update our repository to reflect all the potential new tasks, datasets, as well as any improvement or correction. We are also open to suggestions and welcome any comments to improve our benchmark and elevate its usability.

References

[1] Y. Chen, L. Wu, and M. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020.

[2] E. Chien, J. Peng, P. Li, and O. Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*.

[3] L. Cosmo, A. Kazi, S.-A. Ahmadi, N. Navab, and M. M. Bronstein. Latent patient network learning for automatic diagnosis.(2020). 2020.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[5] B. Fatemi, L. El Asri, and S. M. Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. *Advances in Neural Information Processing Systems*, 34:22667–22681, 2021.

[6] L. Franceschi, M. Niepert, M. Pontil, and X. He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR, 2019.

[7] J. Gasteiger, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations*.

[8] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

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

[10] X. Huang, J. Li, and X. Hu. Label informed attributed network embedding. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 731–739, 2017.

[11] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 66–74, 2020.

[12] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[13] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[14] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[15] J. Leskovec and A. Krevl. Snap datasets: Stanford large network dataset collection, 2014.

[16] K. Li, Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, and Q. He. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 925–935, 2022.

[17] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, and C. Shi. Compact graph structure learning via mutual information compression. In *Proceedings of the ACM Web Conference 2022*, pages 1601–1610, 2022.

[18] X. Liu, M. Yan, L. Deng, G. Li, X. Ye, and D. Fan. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, 9(2):205–234, 2021.

[19] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, pages 1392–1403, 2022.

[20] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup. Revisiting heterophily for graph neural networks. In *Advances in Neural Information Processing Systems*.

[21] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787, 2021.

[22] Y. Ma, X. Liu, N. Shah, and J. Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*.

[23] M. Mazumder, C. Banbury, X. Yao, B. Karlaš, W. G. Rojas, S. Diamos, G. Diamos, L. He, D. Kiela, D. Jurado, et al. Dataperf: Benchmarks for data-centric ai development. *arXiv preprint arXiv:2207.10062*, 2022.

[24] M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[25] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*.

[26] O. Platonov, D. Kuznedelev, A. Babenko, and L. Prokhorenkova. Characterizing graph datasets for node classification: Beyond homophily-heterophily dichotomy. *arXiv preprint arXiv:2209.06177*, 2022.

[27] O. Platonov, D. Kuznedelev, M. Diskin, A. Babenko, and L. Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *The Eleventh International Conference on Learning Representations*, 2023.

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

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

[30] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization.

[31] Z. Song, Y. Zhang, and I. King. Towards an optimal asymmetric graph structure for robust semi-supervised node classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1656–1665, 2022.

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

[33] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, and X. Xie. Graph structure estimation neural networks. In *Proceedings of the Web Conference 2021*, pages 342–353, 2021.

[34] Q. Wu, W. Zhao, Z. Li, D. P. Wipf, and J. Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

[35] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[36] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*.

[37] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pages 5453–5462. PMLR, 2018.

[38] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[39] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[40] D. Yu, R. Zhang, Z. Jiang, Y. Wu, and Y. Yang. Graph-revised convolutional network. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*, pages 378–393. Springer, 2021.

[41] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, and X. Hu. Data-centric ai: Perspectives and challenges. In *SDM*, 2023.

[42] D. Zha, Z. P. Bhat, K.-H. Lai, F. Yang, Z. Jiang, S. Zhong, and X. Hu. Data-centric artificial intelligence: A survey. *arXiv preprint arXiv:2303.10158*, 2023.

[43] J. Zhao, Q. Wen, M. Ju, C. Zhang, and Y. Ye. Self-supervised graph structure refinement for graph neural networks. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 159–167, 2023.

[44] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah. Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pages 11015–11023, 2021.

[45] E. Zheleva and L. Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pages 531–540, 2009.

[46] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16, 2003.

[47] X. Zhu. *Semi-supervised learning with graphs*. Carnegie Mellon University, 2005.

[48] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang. Deep graph structure learning for robust representations: A survey. *arXiv preprint arXiv:2103.03036*, 14, 2021.

[49] D. Zou, H. Peng, X. Huang, R. Yang, J. Li, J. Wu, C. Liu, and P. S. Yu. Se-gsl: A general and effective graph structure learning framework through structural entropy optimization. In *Proceedings of the ACM Web Conference 2023*, pages 499–510, 2023.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] .
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This work does not present any foreseeable societal impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The training detailed are presented in the Supplementary Materials.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Type of resources are described in the Supplementary Materials, though the total amount of compute is note estimated.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] Given in the Supplementary Materials.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] No human data collected.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] No human data collected.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Details on Benchmark

A.1 Datasets

Cora, Citeseer and Pubmed [29] are three citation networks commonly used in prior GSL works [1, 11, 44, 33, 34, 19], with nodes representing papers and edges representing papers' citation relationships. Node feature are bag-of-words feature vectors. The label of each node is its category of research topic.

BlogCatalog [10] is a social network created from an online community where bloggers can follow each other. The features associated with each user are generated based on the keywords in their blog description, while the labels are chosen from a set of predefined categories based on interests of bloggers.

Flickr [10] is a platform for sharing images and videos where users can follow each other, forming a social network. In this dataset, user-specified interest tags are used to generate features, with the groups they have joined serving as labels.

Five datasets from [27] are listed below.

Amazon-ratings is based on the Amazon product co-purchasing network dataset [15]. In this dataset, nodes represent products while edges connect products that are frequently bought together. Node features are the mean of FastText embeddings for words present in the product description. The ratings of products are divided into five distinct classes as labels.

Roman-empire is based on the Roman Empire article from English Wikipedia. In this dataset, each node corresponds to a single word in the text. When two words are consecutive in the text or are connected by a dependency tree, an edge is created between them. Node features are FastText word embeddings. The label of a node is its syntactic role.

Questions is based on a question-answering website. In this dataset nodes represent users, and two users are connected if one user answered the other user's question during a given time interval. Node features are the mean of FastText embeddings for words in the user's description. The label of a node is whether the user remained active on the website. Notably, the dataset is highly imbalanced with 97% of users belonging to the active class.

Minesweeper is a synthetic dataset based on the Minesweeper game. Each node represents a square in a 100×100 grid. Two nodes are connected if they are adjacent in the grid. 20% of the nodes are randomly designated as mines for prediction. Node features are one-hot-encoded numbers of neighboring mines, while for 50% of the selected nodes the features are set unknown to increase task difficulty, indicated by a separate binary feature. Notably, the dataset is also imbalanced.

Wiki-cooc is based on the English Wikipedia. In this dataset, nodes represent unique words and edges connect frequently co-occurring words. Node features are FastText word embeddings. The label of a node is its part of speech.

A.2 Algorithms

Here we introduce all the GSL algorithms implemented in OpenGSL.

ProGNN [11] optimizes the adjacency matrix directly, which is set as an $n \times n$ parameter matrix. Three properties, namely sparsity, low-rankness, and smoothness, are introduced to guide structure learning.

IDGL [1] models the structure as a weighted cosine function of node representations. To increase efficiency for larger graphs, the paper proposes an anchor-based structure learning method.

GRCN [40] incorporates two GNNs, one for node classification and the other one for computing node representations that are used to derive structure via a metric function. Both GNNs are optimized simultaneously to minimize the task loss.

GAug [44] shares a similar architecture with GRCN, differing in that GAug employs a graph auto-encoder to learn the structure, while structure learning is guided by an edge-prediction loss in addition to the task loss.

SLAPS [5] explores the scenario where original structure is absent. It leverages a MLP to obtain node representations to generate structure through a metric function. The overall architecture of SLAPS is similar to GRCN, with an additional auto-denoising loss to guide structure learning. Specifically, the corrupted features and the learned structure are fed into another GNN, and the output is expected to reconstruct the original features.

GEN [33] assumes that the optimal structure is generated by an SBM model, and further assumes that the similarity matrices of the node representations at different levels are observations of the optimal structure. The EM algorithm is employed to learn the expected optimal structure given a well-optimized GNN. GEN performs structural learning and task learning in an iterative way.

GT[30] is a transformer-based approach that performs structure learning before message aggregation in each layer. Edge weights are computed in a similar way as in Transformer, and only weights on pre-existing edges are calculated to reduce complexity.

Nodeformer [34] is a model that allows for layer-wise edge reweighting on all node pairs by utilizing a kernelized Gumbel-Softmax operator, which reduces complexity from quadratic to linear with respect to the number of nodes. To guide structure learning, an extra edge-level regularization is implemented.

CoGSL [17] extracts two fundamental views from the original graph and refines them using a view estimator. An adaptive fusion strategy is employed to obtain the final view. CoGSL maintains the performance of three views while reducing the mutual information between every two views to achieve a "minimal sufficient structure."

SUBLIME [19] presents an approach for unsupervised structure learning. It proposes a structure bootstrapping contrastive learning framework, where an anchor structure is set up to provide supervision signals for the learner structure. Specifically, SUBLIME utilizes a GNN-based encoder to obtain node representations from both views and optimizes the GNN encoder through a node-level contrastive loss. During the training process, the anchor structure is updated every several epochs as the interpolation between the anchor structure and the learner structure.

STABLE [16] obtains reliable node representations leveraging contrastive learning. The new structure is computed as the similarity matrix of node representations. In addition, an advanced GCN is proposed to enhance the robustness of the ordinary GCN.

SEGSL [49] introduces the concept of graph structural entropy. It starts by enhancing the structure guided by the one-dimensional structural entropy maximization strategy. Then, an encoding tree is constructed to capture the hierarchical information of the graph structure. Finally, SEGSL reconstructs the graph structure from the encoding tree. The method performs structure learning and task learning in an iterative way.

Table 6: Hyper-parameter search space of all implemented methods.

Defense	Hyper-parameter	Search Space
General Settings	learning rate	1e-1, 1e-2, 1e-3, 1e-4
	weight decay	5e-4, 5e-5, 5e-6, 5e-7, 0
GCN [13]	number of layers	2, 3, 4, 5
	hidden size	16, 32, 64, 128
	dropout	0, 0.2, 0.5, 0.8
GRCN [40]	K for nearest neighbors	5, 50, 100, 200
	learning rate for graph	1e-1, 1e-2, 1e-3, 1e-4
SLAPS [5]	learning rate of DAE	1e-2, 1e-3
	dropout_adj1	0.25, 0.5
	dropout_adj2	0.25, 0.5
	k for nearest neighbors	10, 15, 20, 30
	λ	0.1, 1, 10, 100, 500
	ratio	1,5,10
	nr	1, 5
GT [30]	number of layers	2, 3, 4, 5
	hidden size	16, 32, 64, 128
	number of heads	1, 2, 4, 8
	dropout	0, 0.2, 0.5, 0.8
Nodeformer [34]	number of layers	2, 3, 4, 5
	hidden size	16, 32, 64, 128
	number of heads	1, 2, 4, 8
	dropout	0, 0.2, 0.5, 0.8
	number of samples for gumbel softmax sampling	5, 10, 20
	weight for edge reg loss	1, 0.1, 0.01
GEN [33]	k for nearest neighbors	7, 8, 9, 10
	tolerance for EM	1e-2, 1e-3, 1e-4
	threshold for adding edges	0.5, 0.6, 0.7, 0.8
SEGSL [49]	K	2, 3, 4
	se	2, 3, 4
ProGNN [11]	learning rate for adj	1e-1, 1e-2, 1e-3, 1e-4
GAug [42]	α for interpolation	0, 0, 0.1, 0.3, 0.5, 0.7, 0.9, 1
	temperature	0.3, 0.6, 0.9, 1.2
	epochs for warm up	0, 10, 20
IDGL [1]	number of anchors	300, 500, 700
	number of heads in structure learning	2, 4, 6, 8
	λ_1 for interpolation	0.7, 0.8, 0.9
	λ_2 for interpolation	0.1, 0.2, 0.3
SUBLIME [19]	dropout rate for edge	0, 0.25, 0.5
	τ for bootstrapping	0.99, 0.999, 0.9999
STABLE [16]	threshold for consine similarity	0.1, 0.2, 0.3
	k for nearest neighbors	1, 3, 5, 7

B Additional Details on Experimental Settings

Running Experiments. Our experiments are mostly conducted on a Linux server with an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz, 125 GB RAM, and an NVIDIA GTX 2080 Ti GPU (12GB). However, as some methods may exceed GPU memory, we run partial experiments (including all experiments related to efficiency) on another Linux server with an Intel(R) Xeon(R) Platinum 8358 CPU @ 2.60GHz, 1008 GB RAM, and an NVIDIA A800 GPU (80GB).

We have developed Open Graph Structure Learning (OpenGSL)[1], which offers a comprehensive and unbiased platform for evaluating GSL algorithms and facilitating future researches in this domain.

[1] https://github.com/OpenGSL/OpenGSL

For more information about the involved packages, please refer to the README file in the Github repository.

General Experimental Settings. We strive to adhere to the original implementation of various GSL methods provided in their respective papers or source codes. To achieve this, we have integrated different options into a standardized framework that includes feature normalization, parameter initialization, training strategies, and so on. We choose not to augment the models with LayerNorm as in [27], since this tchnique has not been adopted by GSL algorithms. Consequently, there may be some discrepancies observed in the results for the five datasets from [27].

Hyperparamter. We conduct comprehensive hyperparameter tuning to ensure a thorough and impartial evaluation of these GSL methods. When the paper and source code of a particular method do not provide information on hyperparameter settings, we tune the corresponding hyperparameters. Hyperparameter tuning is conducted via bayesian search using NNI[2]. The hyperparameter search spaces of all methods are presented in Table 6. More detalis on hyperparameter settings for all the methods can be found in our Github repository [3].

C Additional Results

C.1 Additional Results on the Properties of Learned Structure

In Section 4.2, we analyze the correlation between homophily and performance. The results show that homophily is not always positively correlated with performance and therefore not a suitable guidance in all scenarios. Here, we present additional results on two new measures named "adjusted homophily" and "label informativeness", introduced by [26].

Adjusted homophily is proposed to overcome the sensitivity of commonly used homophily measures to the number of classes and their balance. Specifically, the adjusted homophily of a graph $\mathcal{G}$ is defined as follows:

$$\text{adj\_homo}(\mathcal{G}) = \frac{\text{homo}(\mathcal{G}) - \sum_{k=1}^{C} p(k)^2}{1 - \sum_{k=1}^{C} p(k)^2} \tag{2}$$

where $p(k) = \frac{\sum_{v_i:y_i=k} d(v_i)}{2|E|}$,$d(v_i)$ is the degree of node v_i.

Label informativeness measures the overall informativeness of neighbor's labels for central node's labels in a graph. This metric, which goes beyond the homophily-heterophily dichotomy, can further describe the degree of regularity in the connectivity patterns of heterophilous graphs. The label informativeness of a graph $\mathcal{G}$ is defined as follows:

$$\text{LI}(\mathcal{G}) = 2 - \frac{\sum_{c_1,c_2} p(c_1,c_2) \log p(c_1.c_2)}{\sum_{c} p(c) \log p(c)} \tag{3}$$

where $p(c_1, c_2) = \frac{|(u,v)|(u,v)\in\mathcal{E}, y_u=c_1, y_v=c_2|}{2|\mathcal{E}|}$.

The adjusted homophily and label informativeness of structures learned by GSL methods on various datasets are presented in Fig 5 and Fig 6, respectively.

The figures reveal a similarity in the patterns of adjusted homophily, label informativeness and homophily for structures learned by GSL methods. Notably, adjusted homophily and label informativeness show significant positive correlations with performance only on blogcatalog and flickr, which is the same as homophily. These results imply that the newly proposed measures are still unable to effectively gauge structure quality and unsuitable for guiding structure learning on real data, despite that they are found to be effective on synthetic data [26]. Further exploration is needed in this aspect.

[2]https://github.com/microsoft/nni/

[3]https://github.com/OpenGSL/OpenGSL/paper/configs

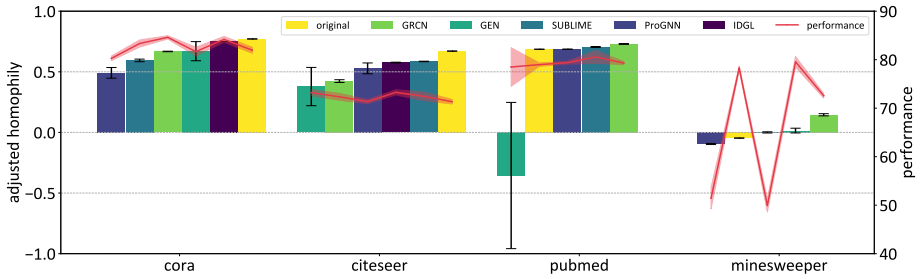

Figure 5: Adjusted homophily of learned structures and performances on homophilous datasets. The methods are ordered by homophily. Left to right on minesweeper: ProGNN, original, SUBLIME, GEN, GRCN.

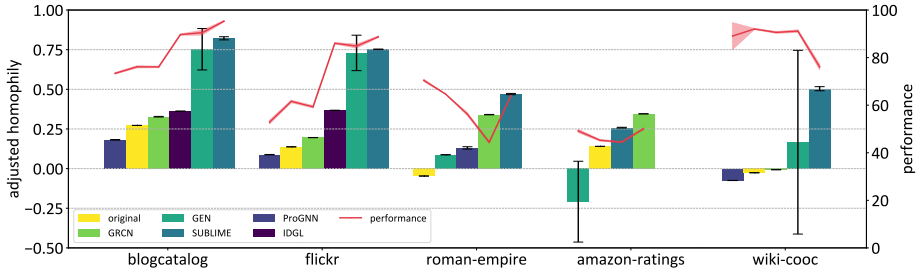

Figure 6: Adjusted homophily of learned structures and performances on heterophilous datasets. The methods are ordered by homophily. Left to right on wiki-cooc: ProGNN, original, GRCN, GEN, SUBLIME.

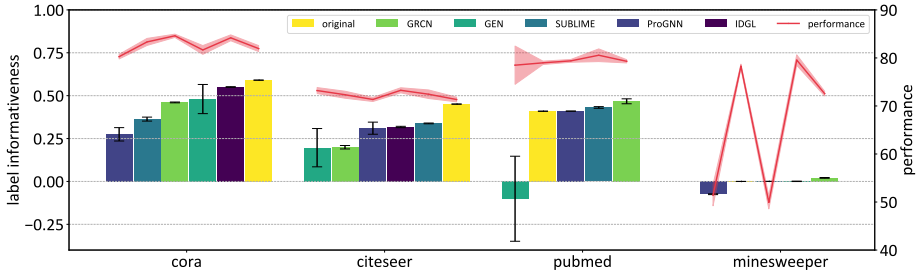

Figure 7: Label informativeness of learned structures and performances on homophilous datasets. The methods are ordered by homophily. Left to right on minesweeper: ProGNN, original, SUBLIME, GEN, GRCN.

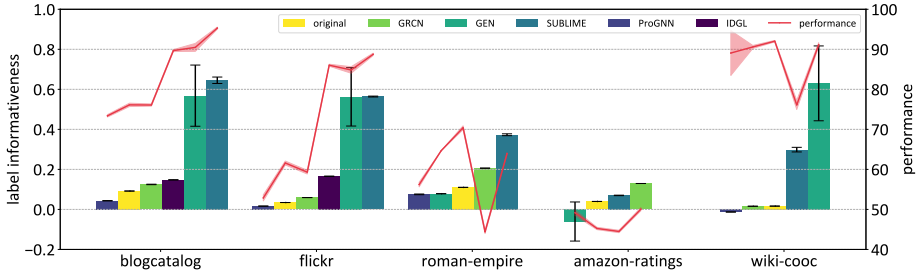

Figure 8: Label informativeness of learned structures and performances on heterophilous datasets. The methods are ordered by homophily.

C.2 Additional Results on Generalizability

In addition to Section 4.3, we also assessed the generalizability of structures learned by various GSL methods on other datasets where GSL methods have made progress. The results are presented in Table 7 - 9.

These additional results are consistent with the conclusions drawn in Section 4.3. In most cases, the learned structures exhibit strong generalizability. These learned structures frequently outperform the original structures and impart additional advantages to various GNN models.

Table 7: Generalizability on Citeseer. Improvements over the original structure are marked green.

Structure source	GCN	SGC	JKNet	APPNP	GPRGNN	LPA	LINK
A	71.34 ± 0.48	71.20 ± 0.00	68.26 ± 0.97	71.86 ± 0.35	71.14 ± 0.34	26.20 ± 0.00	34.70 ± 1.09
ProGNN	70.00 ± 0.66	69.02 ± 0.96	68.33 ± 1.39	71.14 ± 0.53	70.45 ± 0.98	64.46 ± 2.11	57.12 ± 1.71
IDGL	73.03 ± 0.82	73.20 ± 0.48	72.21 ± 1.01	73.82 ± 0.63	72.96 ± 1.05	64.17 ± 1.13	64.87 ± 0.93
GRCN	70.90 ± 0.38	71.44 ± 0.35	71.03 ± 1.49	73.33 ± 0.53	73.25 ± 0.51	70.43 ± 0.30	69.10 ± 0.88
GEN	72.83 ± 0.84	73.02 ± 0.79	72.67 ± 1.33	73.24 ± 0.70	73.27 ± 0.81	65.55 ± 5.35	68.20 ± 3.81
SUBLIME	71.22 ± 0.77	70.76 ± 0.57	68.17 ± 1.21	72.60 ± 0.66	71.64 ± 0.59	53.50 ± 0.79	55.00 ± 1.48

Table 8: Generalizability on Pubmed. Improvements over the original structure are marked green.

Structure source	GCN	SGC	JKNet	APPNP	GPRGNN	LPA	LINK
A	78.98 ± 0.35	78.96 ± 0.05	78.00 ± 0.71	80.10 ± 0.27	79.05 ± 0.44	24.70 ± 0.00	44.71 ± 0.67
GEN	79.50 ± 0.81	79.23 ± 0.82	79.14 ± 0.96	79.83 ± 0.44	79.76 ± 0.93	51.61 ± 17.37	64.79 ± 12.11
ProGNN	78.74 ± 0.65	78.70 ± 0.11	76.97 ± 1.44	80.33 ± 0.29	79.10 ± 0.64	24.80 ± 0.00	44.23 ± 1.24
GRCN	79.15 ± 0.37	79.12 ± 0.09	77.92 ± 0.50	80.28 ± 0.28	79.41 ± 0.51	44.82 ± 0.30	44.57 ± 0.73
SUBLIME	79.24 ± 1.24	79.67 ± 1.10	78.90 ± 1.06	80.99 ± 0.80	80.49 ± 0.93	30.60 ± 1.72	49.58 ± 1.27

Table 9: Generalizability on Flickr. Improvements over the original structure are marked green.

Structure source	GCN	SGC	JKNet	APPNP	GPRGNN	LPA	LINK
A	61.60 ± 0.49	60.72 ± 0.02	57.69 ± 0.64	83.58 ± 0.29	82.31 ± 0.30	39.22 ± 0.00	42.60 ± 0.07
ProGNN	47.28 ± 0.80	60.79 ± 0.24	61.63 ± 0.41	84.16 ± 0.49	86.14 ± 0.20	40.14 ± 0.19	50.15 ± 0.24
GRCN	61.14 ± 0.33	58.86 ± 0.57	60.52 ± 0.36	83.14 ± 0.24	82.11 ± 0.57	44.95 ± 0.49	50.79 ± 0.61
IDGL	85.98 ± 0.21	86.43 ± 0.18	84.97 ± 0.26	87.98 ± 0.33	88.75 ± 0.24	71.18 ± 0.28	39.95 ± 0.42
GEN	85.94 ± 0.55	85.93 ± 0.59	85.86 ± 0.72	84.16 ± 0.49	85.92 ± 0.66	84.15 ± 1.23	84.47 ± 1.02
SUBLIME	88.65 ± 0.26	88.20 ± 0.25	88.00 ± 0.36	89.55 ± 0.23	89.55 ± 0.31	77.74 ± 0.40	71.86 ± 0.51

C.3 Addtional Results on Efficiency

We record the efficiency of existing GSL methods on more datasets, and the results are displayed in Fig 9- 13.

Based on these results, it is evident that the current GSL methods faces challenges on efficiency, and the problem is more severe on large-scale graphs. Henceforth, there is a pressing demand for the development of more efficient GSL methods capable of processing large-scale graph data.

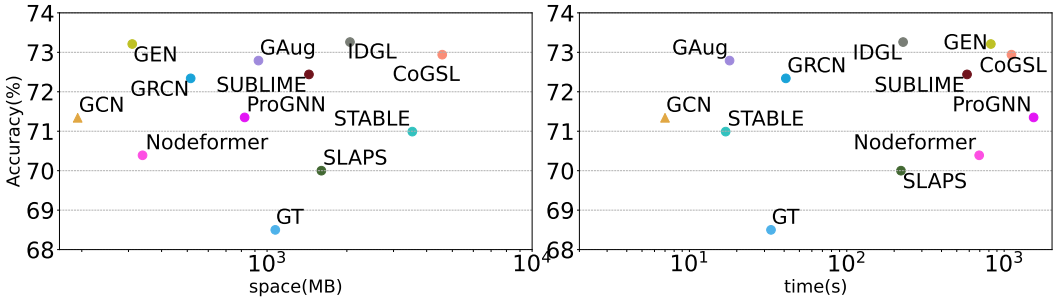

Figure 9: Time and space consumption of different methods on Citeseer

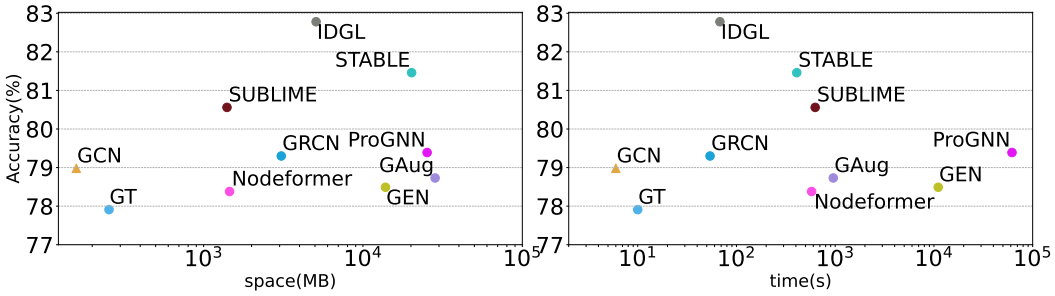

Figure 10: Time and space consumption of different methods on Pubmed

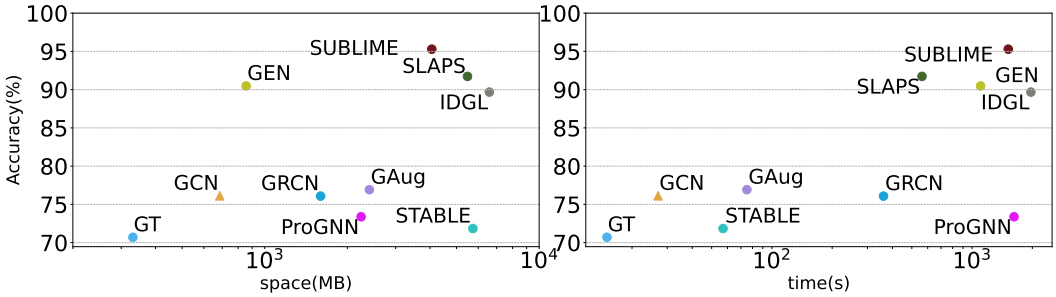

Figure 11: Time and space consumption of different methods on BlogCatalog

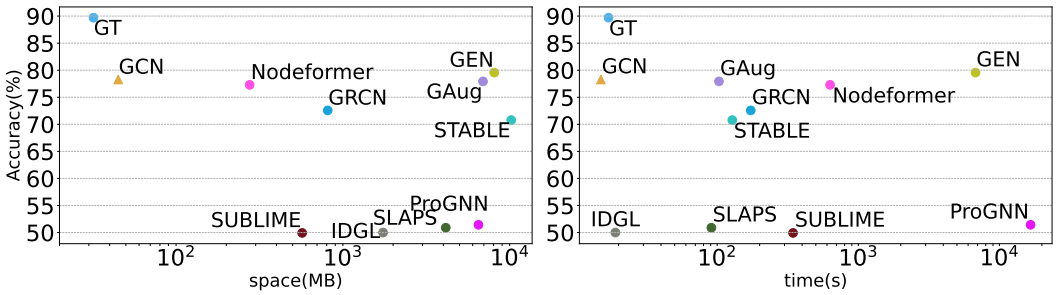

Figure 12: Time and space consumption of different methods on Minesweeper

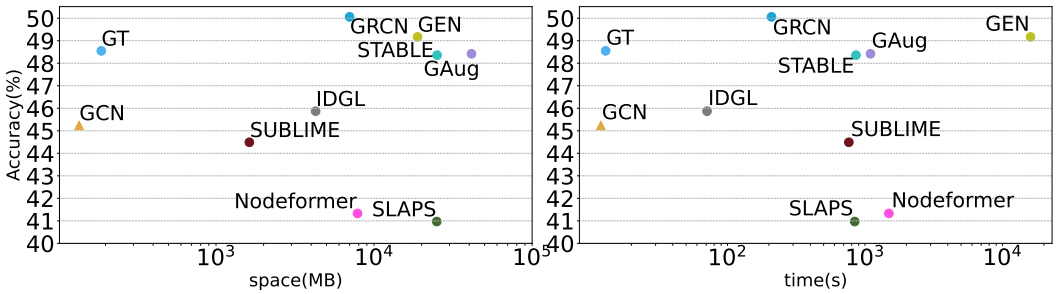

Figure 13: Time and space consumption of different methods on Amazon-ratings

C.4 Addtional Results on Robustness

GSL is capable of refining the original suboptimal graph structure, which theoretically gives it superior robustness compared to GCN. We conduct experiments to evaluate the robustness of some GSL methods. Specifically, we add noisy edges to the original graph to introduce varying degrees of perturbations. Homophily is defined as a measure of perturbations, and an edge with endpoints

belonging to different classes is defined as a noisy edge. Below are performances of some GSL methods on cora and citeseer with varying degrees of perturbations. Figure 14 demonstrates that some GSL methods show a slower rate of decline than GCN, with SUBLIME and GEN displaying superior levels of robustness amongst these methods.

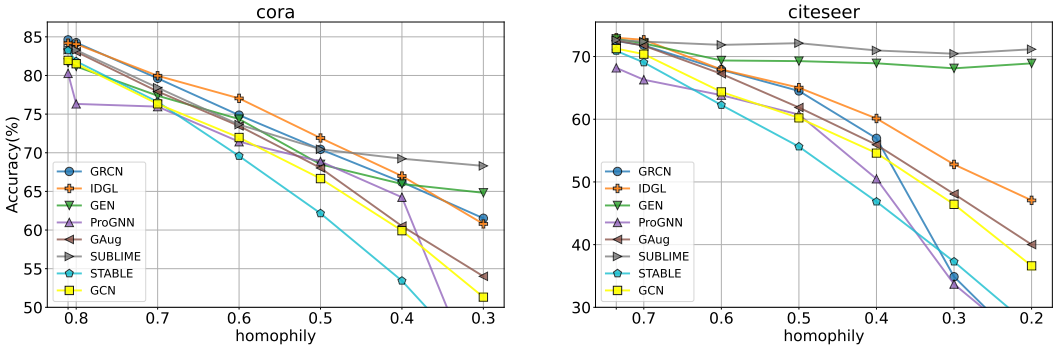

Figure 14: Robustness of GSL methods on Cora and Citeseer

D Reproducibility

All experimental results in OpenGSL are easily reproducible. We provide more detailed information on reproducibility in the following aspects.

Accessibility. All the datasets, algorithm implementations, and experimental configurations can be found in our open-sourced project at `https://github.com/OpenGSL/OpenGSL` without personal request.

Datasets. All the datasets are publicly available. Specifically, Cora, Citeseer, Pubmed, Blogcatalog and Flickr can be loaded with PyTorch Geometric[4]. The other five datasets can be found in the repository[5] of [27].

Code structure. We have organized the code structure to ensure fair experimental settings across algorithms and easy reproduction of the experimental results. OpenGSL consists of the following modules. The `config` module includes the `yaml` files that define the necessary hyperparameters and settings for training and evaluation. The `data` module is used to load datasets. The `method` module contains various GNN and GSL algorithms. The `ExpManager` module manages the learning process, allowing for multiple independent runs.

Documentation and uses. We have made a concerted effort to offer users a README, ensuring their seamless use of our library. We have also added necessary comments to ensure code readability. In addition, we provide the necessary files to reproduce the experimental results, while also serving as examples on how to use the library[6]. To run the code, the users simply need to run `.py` files with specifies arguments such as `data, method` and `gpu`.

License. We use an MIT license for our open-sourced project.

Code maintenance. We are committed to continuously updating our code while proactively addressing user issues and feedback on a regular basis. Furthermore, we enthusiastically welcome contributions from the community to enhance our library and benchmark algorithms. Nonetheless, we will implement strict version control measures to ensure reproducibility during the mainteance.

[4]https://www.pyg.org
[5]https://github.com/yandex-research/heterophilous-graphs
[6]https://github.com/OpenGSL/OpenGSL/tree/main/paper