GraphFM: A Comprehensive Benchmark for Graph Foundation Model

Yuhao Xu[1] **Xinqi Liu**[1] **Keyu Duan**[2] **Yi Fang**[1] **Yu-Neng Chuang**[3]
Daochen Zha[3] **Qiaoyu Tan**[1]
[1]Department of Computer Science, New York University Shanghai
[2]Department of Computer Science, National University of Singapore
[3]Department of Computer Science, Rice University
{yx3534, xl4600, yf2722, qiaoyu.tan}@nyu.edu
k.duan@u.nus.edu, {ynchuang, daochen.zha}@rice.edu

Abstract

Foundation Models (FMs) serve as a general class for the development of artificial intelligence systems, offering broad potential for generalization across a spectrum of downstream tasks. Despite extensive research into self-supervised learning as the cornerstone of FMs, several outstanding issues persist in Graph Foundation Models that rely on graph self-supervised learning, namely: 1) **Homogenization**. The extent of generalization capability on downstream tasks remains unclear. 2) **Scalability**. It is unknown how effectively these models can scale to large datasets. 3) **Efficiency**. The training time and memory usage of these models require evaluation. 4) **Training Stop Criteria**. Determining the optimal stopping strategy for pre-training across multiple tasks to maximize performance on downstream tasks. To address these questions, we have constructed a rigorous benchmark that thoroughly analyzes and studies the generalization and scalability of self-supervised Graph Neural Network (GNN) models. Regarding generalization, we have implemented and compared the performance of various self-supervised GNN models, trained to generate node representations, across tasks such as node classification, link prediction, and node clustering. For scalability, we have compared the performance of various models after training using full-batch and mini-batch strategies. Additionally, we have assessed the training efficiency of these models by conducting experiments to test their GPU memory usage and throughput. Through these experiments, we aim to provide insights to motivate future research. The code for this benchmark is publicly available at https://github.com/NYUSHCS/GraphFM.

1 Introduction

Foundation Models (FMs) represent an emerging paradigm of AI, focused on pre-training models on large datasets and subsequently adapting them to various downstream tasks [1]. FMs have already made significant strides in the field of Natural Language Processing (NLP), driven by the remarkable success of Large Language Models (LLMs) [2, 3, 4, 5, 6, 7]. Inspired by their success in NLP, FMs have naturally emerged as prominent research subjects across various other domains, such as computer vision [8, 9], time series analysis [10, 11], and recommender systems [12].

Graph learning is also evolving towards Graph FMs, propelled by advancements in Graph Self-Supervised Learning (GSSL) [13, 14, 15]. In GSSL, models are trained by solving auxiliary tasks, using supervision signals derived directly from the data itself without the need for human annotations. Consequently, GSSL is an effective approach to realizing Graph FMs by pre-training graph models on large unlabeled graphs. Existing GSSL methods typically follow two paradigms: contrastive models

Preprint. Under review.

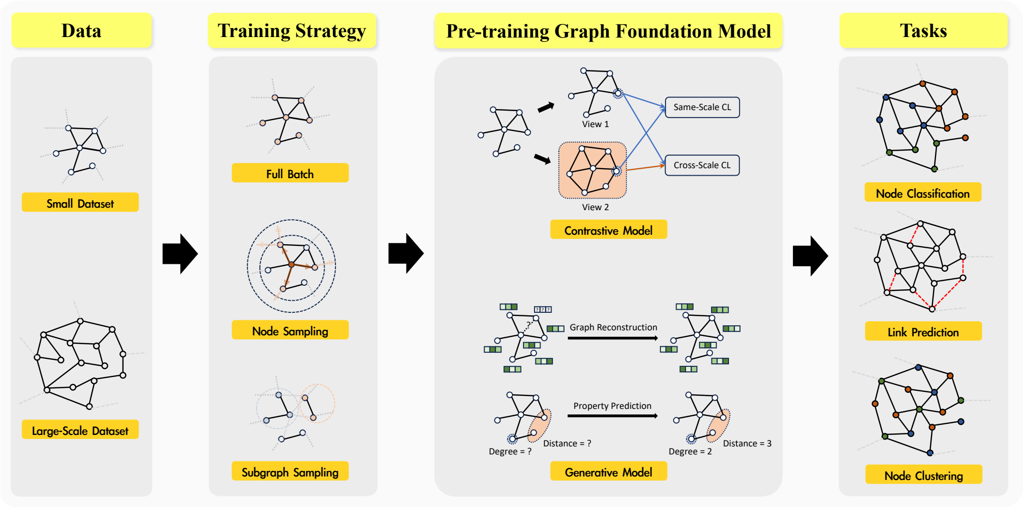

Figure 1: An overview of GraphFM. We perform a comprehensive benchmark of state-of-the-art self-supervised GNN models through four key aspects: dataset scale, training strategies, GSSL methods for Graph FMs, and adaptability to different downstream tasks.

and generative models. Contrastive models generate two graph views through data augmentation and employ graph neural networks (GNNs) to learn representations by optimizing a contrastive objective [16]. Generative models parameterize the encoder using GNNs [17, 18] and train the model by reconstructing observed edges [19, 20, 21] or node attributes [22, 23].

However, despite the plethora of proposed GSSL methods, it remains unclear how much progress we have made towards Graph FMs. *(i) There is no clear understanding of the homogenization [13], or generalization across different downstream tasks, of existing GSSL methods.* The majority of GSSL algorithms predominantly concentrate on node classification tasks, with limited evaluation on other downstream tasks [24, 25, 26, 18, 27, 28, 29]. Conversely, some are exclusively tailored to address link prediction tasks [30] or clustering tasks [31]. Thus, there is a lack of evaluation to understand how each GSSL method performs on all tasks. *(ii) Existing GSSL methods are evaluated under different settings, leading to results that are not directly comparable.* For example, S2GAE [32] is evaluated by an SVM classifier to do node classification task, while GraphMAE [24] uses MLP. For hyperparameters, CCA-SSG [27] searches for the learning rate in [5e-4, 1e-3, 5e-3], while GrapMAE2 [25] explores [2.5e-3, 2e-3, 1e-3]. Such critical details can have a substantial impact on performance, yet they are not thoroughly addressed in the existing literature. *(iii) There is a deficiency in evaluating the performance of GSSL methods across datasets of varying scales using different sampling strategies.* Some methods have only been evaluated on small datasets, lacking experimental validation on large-scale data [33, 27, 26], where full-batch training is often impractical, necessitating mini-batch training with specific sampling strategies. In this case, the selection of sampling strategies can significantly impact performance, underscoring the need for a more comprehensive evaluation.

To bridge this gap, we introduce GraphFM, the first comprehensive benchmark for building Graph FMs based on GSSL. An overview of GraphFM is depicted in Figure 1. GraphFM rigorously evaluates combinations across four key aspects: dataset scale, training strategies, various GSSL methods for Graph FMs, and adaptations to different downstream tasks. For a fair comparison, we implement all GSSL methods within a unified framework and employ consistent data processing and splitting methods for both training and evaluation. Additionally, we conduct hyperparameter searches with the same search budgets for all methods. In summary, our contributions include:

- **Comprehensive benchmark.** GraphFM enables a fair comparison among eight representative GSSL methods under a unified experimental setup across six popular datasets with varying scales.
- **Multi-dimensional Analysis.** GraphFM employs both full-batch and mini-batch training strategies and utilizes the trained node representations to perform three downstream tasks: node classification, link prediction, and node clustering. We systematically analyze the performance and efficiency

under various settings. Furthermore, we investigate the influence of using performances from different downstream tasks or alternative metrics as early stopping criteria to train Graph FMs.

- **Openness.** We have open-sourced our code and the pre-trained models on GitHub to facilitate future research. Based on our benchmark findings, we also outline potential future research directions to inspire further studies.

2 Preliminaries

Notations and Problem Formulation. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ be a graph, where $\mathcal{V}$ is the set of N nodes, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. $\mathcal{E}$ denotes the edge set and $\mathbf{X} \in \mathbb{R}^{N \times d}$ represents the corresponding feature matrix with dimension d. Typically, a GNN model is parameterized by a mapping function $f : (\mathbf{A}, \mathbf{X}) \rightarrow \mathbf{H} \in \mathbb{R}^{N \times l}$, which maps each node $v \in \mathcal{V}$ into a l-dimensional embedding vector $\mathbf{h}_v \in \mathbb{R}^l$, where $\mathbf{h}_v$ is the v-th row of $\mathbf{H}$. Once we obtain $\mathbf{H}$, we can adapt them with a head to perform downstream tasks. The objective of Graph FMs is to train a model that can generate high-quality $\mathbf{H}$, typically with GSSL methods, such that the adapted models can perform well across various downstream tasks.

Homogenization of Graph FMs. Homogenization means the generalization capability of a FM to different downstream tasks [13]. In the context of Graph FMs, we focus on three common tasks, including node classification, link prediction, and node clustering.

Scalability and Training Strategies. To train Graph FMs, it is often crucial to employ GSSL methods on large graphs. Standard GNNs typically operate in a full-batch setting, retaining the entire graph structure during forward and backward propagation to facilitate message passing (MP) among nodes. However, as the graph size increases, full-batch training becomes impractical due to significant memory usage and extensive computation time [34]. In this scenario, mini-batch training strategies can be adopted, using sampled subgraphs as mini-batches to approximate full-batch message passing, thereby significantly reducing memory consumption. Specifically, MP with K layers can be expressed as follows:

$$\mathbf{X}^{(K)} = \mathbf{A}^{(K-1)}\sigma\left(\mathbf{A}^{(K-2)}\sigma\left(\cdots\sigma\left(\mathbf{A}^{(0)}\mathbf{X}^{(0)}\mathbf{W}^{(0)}\right)\cdots\right)\mathbf{W}^{(K-2)}\right)\mathbf{W}^{(K-1)}$$

where σ is an activation function (e.g. ReLU) and $\mathbf{A}^{(l)}$ is the weighted adjacency matrix at the l-th layer. In the full-batch setting, $\mathbf{A}^{(l)}$ encompasses all nodes in the graph, while in the mini-batch setting, $\mathbf{A}^{(l)}$ only covers a subset of the nodes, resulting in $\mathbf{A}^{(l)}$ being a sub-matrix of the full adjacency matrix. The choice of sampling strategy plays an important role; in this work, we focus on two commonly used sampling strategies: node sampling [35] and subgraph sampling [36].

Early stopping criteria. When pre-training GNN models, we commonly employ early stopping and save the best model based on the performance of a specified metric on the validation set. Subsequently, we evaluate this saved model on the test set. This process is straightforward when focusing on a single task and evaluation metric, as often seen in the GSSL literature. However, training Graph FMs requires achieving good performance across various downstream tasks and metrics, such as accuracy and AUC. The impact of early stopping criteria on this objective has not been fully explored.

3 Benchmark Design

We begin by introducing the datasets used in our benchmarking process, along with the algorithm implementations. Then, we pose the research questions to guide our benchmarking study.

3.1 Dataset and Implementations

Datasets. To conduct a comprehensive evaluation of existing GSSL methods, we selected six widely used graph node classification datasets from the GSSL literature. Table 1 shows the statistical data of datasets, these datasets vary in size, allowing us to assess the generalization capabilities of current methods across different data scales. Specifically, we utilized three classic citation datasets: Cora, Citeseer, and Pubmed [37]. Additionally, we included two popular social network datasets:

Table 1: An overview of the datasets used in this study.

Dataset	# Nodes	#Edges	# Feat.	Avg. # degree	# Classes
Cora	2,708	5,278	1,433	3.9	7
Citeseer	3,327	4,552	3,703	2.7	6
Pubmed	19,717	44,324	500	4.5	3
Flickr	89,250	899,756	500	10.09	7
Reddit	232,965	11,606,919	602	493.56	41
ogbn-arxiv	169,343	1,166,243	128	13.7	40

Flickr [38] and Reddit [35], along with the arxiv citation dataset from the Open Graph Benchmark (OGB) [39]. We provide more details in Appendix A.1.

Implementations. We consider a collection of state-of-the-art GSSL methods. For contrastive methods, we include BGRL [28], CCA-SSG [27], GCA [33], GBT [26] and GraphECL [40]. For generative methods, we consider GraphMAE [24], GraphMAE2 [25] and S2GAE [32]. 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 about the implementations and the hyperparameter search process are in Appendix A.2.

3.2 Research Questions

We carefully design the `GraphFM` to systematically evaluate existing methods to motivate future research. Specifically, our aim is to address the following research questions.

RQ1: How do existing GSSL methods perform in terms of node classification performance?

Motivation: Node classification stands as the most commonly used task in GSSL literature. Our first research question aims to reassess existing papers within this standard task, employing consistent evaluation methods to ensure a fair comparison.

Experiment Design: We conduct experiments following standard settings, wherein the models are trained on the Cora, Citeseer, and Pubmed datasets using a full-batch training strategy. Early stopping is based on accuracy for the node classification task, and performance is evaluated using the same criterion. More details can be found in Appendix B.1.

RQ2: How do pre-trained Graph FMs perform in terms of performance on other downstream tasks such as link prediction and node clustering?

Motivation: To evaluate the homogenization of GSSL methods, experiments across various downstream tasks are necessary to understand each method's generalization performance.

Experiment Design: After obtaining pre-trained Graph FMs, we utilize the node representations post-training to conduct node classification, link prediction, and node clustering tasks. For link prediction tasks, we employ area under the curve (AUC) and average precision score (AP), while for node clustering tasks, we use normalized mutual information (NMI) and adjusted rand index (ARI), which are all the standard metrics. More details can be found in Appendix B.2.

RQ3: How do various training strategies (i.e., full batch, node sampling, or subgraph sampling) influence the performance of Graph FMs? How efficient are these strategies, particularly when dealing with large-scale graphs?

Motivation: RQ1 and RQ2 focus on small datasets, while for large-scale datasets, full-batch training strategies may not be feasible. Hence, examining model performance and efficiency under mini-batch training strategies is essential to assess scalability.

Experiment Design: We train the GSSL models on the Flickr, Reddit, and Arxiv datasets using two mini-batch training strategies: node sampling and subgraph sampling. Tasks include node classification, link prediction, and node clustering tasks. Additionally, to understand the training speed and memory usage of the GSSL methods using different sampling strategies, we report throughput and actual memory usage during training. More details can be found in Appendix B.3.

Table 2: The performance of node classification for full batch in Cora, Citeseer and Pubmed. Averaged results with 5 different random seeds are reported. Highlighted are the top first, second, and third results.

Paradigm	Models	cora	citeseer	pubmed
Contrastive	BGRL	81.27±0.95	71.35±0.65	**86.19±0.17**
	CCA-SSG	**86.50±0.03**	**73.36±0.75**	85.14±0.05
	GBT	81.81±0.95	67.24±0.94	78.83±0.61
	GCA	**85.87±0.49**	71.88±0.42	**86.22±0.75**
	GraphECL	84.26±0.06	70.73±0.68	**86.04±0.07**
Generative	GraphMAE	**85.78±0.69**	**73.41±0.35**	84.28±0.13
	GraphMAE2	84.84±0.22	**72.26±0.25**	84.93±0.01
	S2GAE	82.90±0.31	69.34±0.19	81.57±0.06

RQ4: Will using performances from different downstream tasks or alternative metrics as early stopping criteria impact the effectiveness of Graph FMs?

Motivation: In the aforementioned RQs, we save the best-performing model in node classification tasks and subsequently test it on the test set. However, the model obtained in this way may not necessarily perform well in other downstream tasks. Thus, it is essential to investigate the impact of different early stopping criteria.

Experiment Design: We explore the viability of saving pre-trained models based on their results across different downstream tasks, such as link prediction and node clustering, and subsequently evaluate their effectiveness across various training strategies and downstream tasks. More details can be found in Appendix B.4.

4 Experiments Results and Analyses

4.1 Performance Comparison in Node Classification (RQ1)

We report the performance of all methods on 3 small datasets with full batch training strategy in Table 2. We made several key observations from the table.

① **Thanks to the standardized settings, our reproduced results on full-batch training are generally comparable to or sometimes even higher than those in the original paper.** `GraphFM` utilizes Optuna [41] to aid in hyperparameter search for achieving optima model performance. As shown in Table 2, the top-performing datasets exhibit higher accuracy results than those reported in the original literature. Notably, the node classification results on the PubMed dataset exceed the previous benchmarks by as much as 4 percentage points. This improvement is likely due to Optuna identifying more suitable hyperparameters for the model after standardizing the settings. The only exception is that, for S2GAE, the performance is worse compared to the original paper. The possible reason is that the node classification task in the original study was conducted using an SVM classifier, whereas `GraphFM` employs an MLP head to all methods so that the results are comparable.

② **The performance gap between leading contrastive and generative paradigms on node classification is marginal.** Although the learning processes of contrastive and generative-based GSSL models differ, they exhibit similar performance on Cora, CiteSeer, and PubMed as shown in Table 2. It is noteworthy that traditional beliefs regarding generative models (e.g., GAE [19]) suggest that they cannot perform comparably to contrastive-based methods on node classification tasks. However, as observed in Table 2, advanced generative models (such as GraphMAE, GraphMAE2, and S2GAE) achieve highly competitive results in classification. Particularly in the Cora and Citeseer tasks, the average performance of the generative approach even surpasses that of the contrastive models.

4.2 Performance Comparison in Link Prediction and Node Clustering (RQ2)

In this section, we investigate the homogenization capability of pre-trained graph FMs across various tasks. Specifically, in our experiments, `GraphFM` saves the pre-trained models based on the highest accuracy achieved in the node classification task. Subsequently, it performs three downstream tasks:

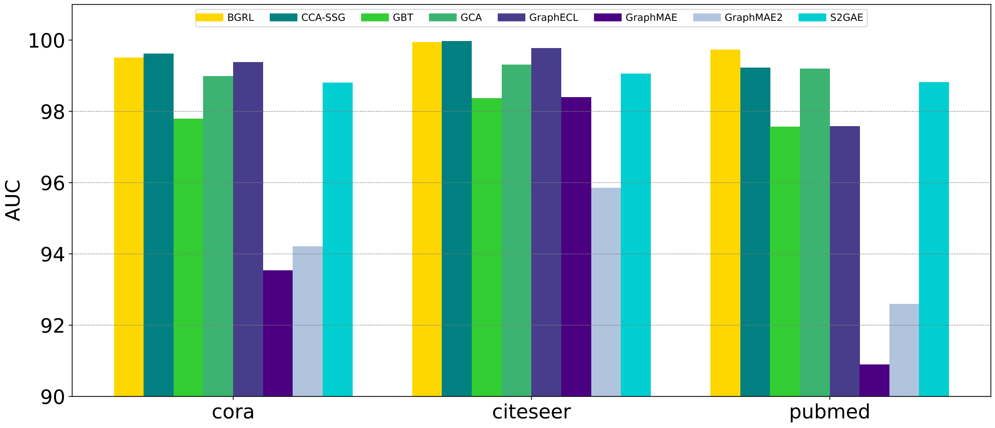

Figure 2: Link Prediction results on Cora, Citeseer, Pubmed based on full batch training.

node classification, link prediction, and node clustering. Since the node classification task has already been discussed in RQ1, here, we analyze the pre-trained models' generalization ability on link prediction and node clustering tasks. Figures 2 and Figure 4 (in Appendix C.1) present the results of link prediction and node clustering, respectively, from which we made the following observations:

③ **Generative models (GraphMAE and GraphMAE2) perform poorly on link prediction tasks.** While advanced generative models like GraphMAE and GraphMAE2 demonstrate competitive performance on node classification tasks, as depicted in Table 2, they underperform other methods on link prediction tasks, as illustrated in Figure 2. The possible reason for this discrepancy is that these models solely concentrate on reconstructing node features and neglect the conventional reconstruction of network structure, which is essential for inferring missing links.

④ **Although generative models fall short in link prediction, they outperform other baselines in node clustering tasks.** As depicted in Figure 4 in the Appendix C.1, except for CCA-SSG, GraphMAE and GraphMAE2 consistently surpass all contrastive-based methods on the Cora dataset and outperform all comparative methods in other scenarios. These findings, combined with their performance in node classification, underscore the advantages of node feature reconstruction as a general node-level learning objective

⑤ **Both contrastive and generative models demonstrate strong homogenization capability on small datasets.** As illustrated in Figures 2 and 4, we can see that although the pre-trained models were saved according to their performance in node classification tasks, they still perform quite well across other downstream tasks in general, exhibiting good homogenization capabilities on small datasets.

4.3 Performance and Efficiency Comparison in Large-Scale Dataset (RQ3)

In this section, we conduct experiments across three downstream tasks to evaluate the performance of GSSL methods on large scale datasets w.r.t. different training strategies. The training results of node sampling are recorded in Table 8, 10, 12, while the training results of subgraph sampling are recorded in Table 9, 11, 13. All these tables can be found in Appendix C.2.

⑥ **On small datasets, the mini-batch version of existing GSSL methods generally yields lower performance across the three downstream tasks compared to their full batch counterparts.** From Tables 8 and 9, we observe that in the node classification task, the performance of almost all models decreases compared to the full batch variants, except for GBT, which shows an improvement. From Tables 10 and 11, GraphMAE exhibits a significant improvement over its full batch version in the link prediction task, although contrastive models still generally perform better. From Tables 12 and 13, in the node clustering task, the performance gap between contrastive and generative models diminishes compared to the full batch training scenarios. Tables 3 and 4 record the training results of GraphFM with mini-batch on the PubMed dataset. From the tables, we can see that, overall, the performance of mini-batch training is slightly lower compared to full-batch training. The generative models exhibit a more significant performance drop than the contrastive models.

Table 3: The result of GraphFM in Pubmed dataset with node sampling training strategy.

Models	Node Classification	Link Prediction	Node Clustering
BGRL	83.70(↑ 2.43)	99.60(↓ 0.13)/99.52(↓ 0.13)	0.1139(↓ 0.2025)/0.0588(↓ 0.2130)
CCA-SSG	83.51(↓ 2.99)	99.58(↑ 0.35)/99.49(↑ 0.59)	0.1246(↑ 0.0324)/0.0625(↑ 0.0580)
GBT	84.23(↑ 2.42)	99.40(↑ 1.83)/99.31(↑ 0.41)	0.0453(↓ 0.1342)/-0.0084(↓ 0.0132)
GCA	82.29(↓ 3.58)	99.17(↓ 0.02)/99.08(↑ 0.06)	0.0708(↓ 0.1087)/0.0150(↓ 0.1398)
GraphECL	83.10(↓ 1.16)	95.28(↓ 2.30)/94.67(↓ 2.01)	0.1056(↑ 0.1676)/0.0199(↑ 0.2350)
GraphMAE	83.60(↓ 2.18)	96.04(↑ 5.15)/94.72(↑ 3.75)	0.1125(↓ 0.2107)/0.0250(↓ 0.2731)
GraphMAE2	80.76(↓ 4.08)	84.73(↓ 7.86)/85.37(↓ 6.59)	0.2770(↓ 0.0557)/0.2594(↓ 0.0541)
S2GAE	81.40(↓ 1.50)	88.77(↓ 10.05)/86.33(↓ 12.51)	0.2647(↑ 0.1671)/0.2757(↑ 0.2016)

Table 4: The result of GraphFM in Pubmed dataset with subgraph sampling training strategy.

Models	Node Classification	Link Prediction	Node Clustering
BGRL	84.52(↓ 1.67)	99.47(↓ 0.26)/99.39(↓ 0.26)	0.2272(↓ 0.0892)/0.1830(↓ 0.0888)
CCA-SSG	84.83(↓ 0.31)	99.69(↑ 0.46)/99.62(↑ 0.72)	0.2482(↑ 0.1560)/0.2214(↑ 0.2169)
GBT	82.61(↑ 3.78)	98.70(↑ 1.13)/98.83(↑ 1.68)	0.0638(↓ 0.1157)/-0.0018(↓ 0.1566)
GCA	84.43(↓ 1.79)	98.85(↓ 0.34)/98.80(↓ 0.22)	0.0888(↓ 0.1844)/0.0182(↓ 0.2367)
GraphECL	84.59(↓ 1.45)	96.23(↓ 1.35)/95.69(↓ 0.99)	0.3442(↑ 0.2473)/0.3057(↑ 0.2878)
GraphMAE	84.85(↑ 0.57)	95.86(↑ 4.97)/94.64(↑ 3.67)	0.3211(↓ 0.0021)/0.2893(↓ 0.0142)
GraphMAE2	80.24(↓ 4.96)	83.99(↓ 8.60)/84.79(↓ 7.17)	0.2773(↓ 0.0544)/0.2598(↓ 0.0537)
S2GAE	79.51(↓ 2.06)	89.72(↓ 1.91)/88.09(↓ 3.54)	0.3001(↑ 0.2025)/0.2865(↑ 0.2124)

⑦ **On large datasets, the performance of existing GSSL methods varies significantly across different tasks.** In the node classification task, both contrastive and generative-based methods exhibit similar performance across the three datasets. However, in the node clustering task, generative models consistently outperform contrastive models in all cases. It is noteworthy that some results display negative values in this task. This arises from the calculation formula of the Adjusted Rand Index (ARI), which ranges from [-1, 1]. Thus, these negative values fall within the expected range. Table 5 shows the results of GraphFM in Flickr dataset.

For training efficiency, ⑧ **generative models do not encounter out-of-memory issues, providing them with a notable advantage in scalability on large-scale datasets.** Comparing the two mini-batch methods, ⑨ **subgraph sampling exhibits the lowest memory usage and the fastest training speed,** as evidenced in Table 14, Table 15 (in Appendix C.3), and Figure 3, particularly on larger datasets. In summary, considering the variable performance of mini-batch variants, exploring the design of effective self-supervised training architectures or objectives within the mini-batch framework represents a promising avenue for future research.

4.4 Performance Using Alternative Early Stopping Criterion (RQ4)

Since all models perform well with both full batch and mini-batch methods on small datasets, our experiment will focus on large-scale dataset. In this section, we focus on using link prediction and node clustering as the early stop criteria. Specifically, we save pre-trained models based on their performance in the downstream task and test there performance across three different downstream tasks on large dataset. Table 6 reports the results on Flickr dataset, and more results can be found in Appendix C.4.

⑩ **GSSL methods can achieve better performance on a downstream task when using the same task as the early stop criteria.** According to Table 6, it is evident that compared to previous experiments, the performance of contrastive models in link prediction has significantly improved. Conversely, no substantial performance enhancement is observed in generative methods. These findings underscore the impact of early stopping criteria on various self-supervised training methodologies, especially for contrastive-based approach. Similar observations can be made when employing node clustering as the early stoping criterion (see Appendix C.4 for details).

5 Future Directions

Drawing upon our empirical analyses, we point out some promising future directions for GraphFM.

Table 5: The result of `GraphFM` in Flickr dataset. " - " means out of memory.

Training Strategy	Models	Node Classification	Link Prediction	Node Clustering
Node Sampling	BGRL	47.37±0.05	87.88/88.24	0.0054/0.0145
	CCA-SSG	51.59±0.14	76.45/14.44	0.0622/0.0397
	GBT	52.11±0.08	86.69/87.93	0.0179/0.0175
	GCA	-	-	-
	GraphECL	-	-	-
	GraphMAE	49.25±0.13	50.00/50.00	0.0154/0.0197
	GraphMAE2	46.07±0.83	49.94/49.98	0.0157/0.0097
	S2GAE	43.90±0.17	49.95/49.93	0.0067/0.0054
Subgraph Sampling	BGRL	47.14±0.07	86.92/87.57	0.0052/0.0145
	CCA-SSG	50.95±0.20	50.00/50.00	0.0181/0.0125
	GBT	51.00±0.17	50.00/50.00	0.0453/0.0176
	GCA	51.76±0.08	50.00/50.00	0.0343/0.0155
	GraphECL	46.23±0.09	51.06/51.12	0.0128/0.0142
	GraphMAE	45.30±0.85	50.00/50.00	0.0090/0.0137
	GraphMAE2	46.25±0.90	49.94/49.98	0.0157/0.0068
	S2GAE	43.97±0.20	49.91/49.91	0.0039/0.0033

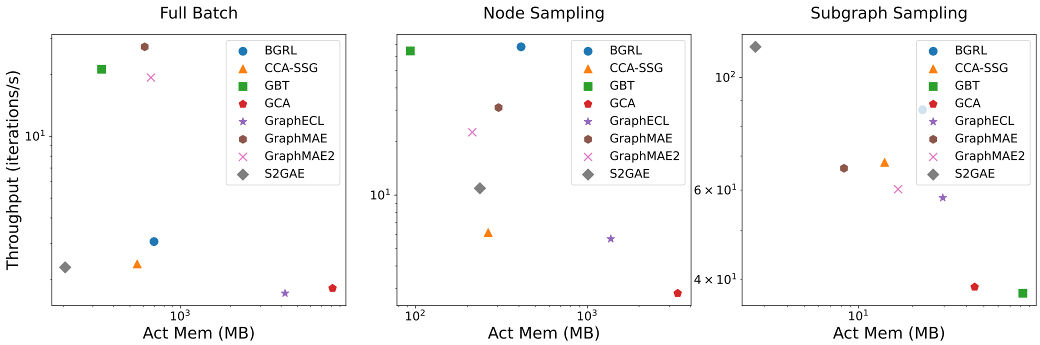

Figure 3: Time and space consumption of different methods and training strategy on Pubmed.

Reconsidering the homogeneity of contrastive models and generative models is imperative. Homogeneity is a significant characteristic of FMs and should be given high priority. However, based on the results from `GraphFM`, current contrastive and generative models face substantial challenges in achieving homogeneity. These challenges arise from various factors such as the datasets, node-level or edge-level downstream tasks.

Exploring an effective early stop strategy for GNN pre-training. Based on the above experiments, no single early stopping criterion currently enhances model performance across various downstream tasks, contradicting the original intention of the foundation model. Future research should focus on exploring more effective early stopping criteria.

How to extend graph foundation model to textual attributed graphs. Presently, our training primarily revolves around conventional graph datasets, where where node features are numerical vectors. Nonetheless, in real-world graph applications, nodes are often characterized by textual descriptions, such as social media posts on Twitter, formally referred to as textual attribute graphs. Whether GraphFM can be extended to accommodate such graphs, and how it should be extended, remains an open research question.

6 Conclusion

This paper introduces `GraphFM`, a comprehensive benchmark for Graph Foundation Models. We reimplement and compare 8 leading GSSL methods across diverse datasets, providing a fair compar-

Table 6: The result of `GraphFM` in Flickr dataset by saving valid model with the best performance in link prediction. " - " means out of memory.

Training Strategy	Models	Node Classification	Link Prediction	Node Clustering
Node Sampling	BGRL	46.07±0.56	86.60/87.31	0.0073/0.0195
	CCA-SSG	51.03±0.03	98.98/98.90	0.0555/0.0538
	GBT	51.66±0.17	85.39/86.64	0.0727/0.0386
	GCA	-	-	-
	GraphECL	-	-	-
	GraphMAE	45.86±0.05	50.00/50.00	0.0128/0.0082
	GraphMAE2	45.80±0.18	50.08/50.04	0.0257/0.0090
	S2GAE	45.13±0.32	50.65/50.41	0.0246/0.0175
Subgraph Sampling	BGRL	46.09±1.31	86.36/87.06	0.0074/0.0211
	CCA-SSG	50.92±0.05	86.36/87.50	0.0827/0.0645
	GBT	52.14±0.06	52.24/57.72	0.0423/0.0134
	GCA	51.94±0.72	72.84/65.80	0.0744/0.0315
	GraphECL	47.09±0.34	51.16/50.92	0.0043/0.0022
	GraphMAE	49.63±0.42	57.67/54.80	0.0201/0.0124
	GraphMAE2	45.86±0.09	50.10/50.05	0.0249/0.0109
	S2GAE	44.76±0.43	50.25/50.30	0.0068/0.0049

ison and insightful analysis into this burgeoning research field. Our empirical observations reveal variations in performance between full-batch and mini-batch training scenarios. Furthermore, we find that existing self-supervised GNN pre-training efforts may not effectively serve as foundation models on graphs, as they often struggle to generalize well across key graph reasoning tasks (node classification, link prediction, and node clustering) simultaneously. Notably, we highlight the significant impact of early stopping criteria in GNN pre-training on model generalization capability, a critical issue previously overlooked by the research community. We believe that this benchmark will have a positive impact on this emerging research domain. Our code is publicly available, and we encourage contributions of new datasets and methods. In the future, we aim to extend the applicability of `GraphGLM` to text-attributed graphs and broaden its support for various graph-level learning tasks and heterogeneous graphs, enhancing its versatility and comprehensiveness.

References

[1] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[6] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[7] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

[8] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[9] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023.

[10] Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023.

[11] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

[12] Shijie Geng, Juntao Tan, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. Vip5: Towards multimodal foundation models for recommendation. *arXiv preprint arXiv:2305.14302*, 2023.

[13] Jiawei Liu, Cheng Yang, Zhiyuan Lu, Junze Chen, Yibo Li, Mengmei Zhang, Ting Bai, Yuan Fang, Lichao Sun, Philip S Yu, et al. Towards graph foundation models: A survey and beyond. *arXiv preprint arXiv:2310.11829*, 2023.

[14] Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*, 45(2):2412–2429, 2022.

[15] Lirong Wu, Haitao Lin, Cheng Tan, Zhangyang Gao, and Stan Z Li. Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 35(4):4216–4235, 2021.

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

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

[18] Qiaoyu Tan, Ninghao Liu, and Xia Hu. Deep representation learning for social network analysis. *Frontiers in big Data*, 2:2, 2019.

[19] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[20] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

[21] Yucheng Shi, Yushun Dong, Qiaoyu Tan, Jundong Li, and Ninghao Liu. Gigamae: Generalizable graph masked autoencoder via collaborative latent space reconstruction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2259–2269, 2023.

[22] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017.

[23] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. Co-embedding attributed networks. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 393–401, 2019.

[24] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022.

[25] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *Proceedings of the ACM Web Conference 2023*, pages 737–746, 2023.

[26] 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.

[27] 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.

[28] 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.

[29] Yi Fang, Dongzhe Fan, Daochen Zha, and Qiaoyu Tan. Gaugllm: Improving graph contrastive learning for text-attributed graphs with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2024.

[30] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[31] Yue Liu, Ke Liang, Jun Xia, Sihang Zhou, Xihong Yang, Xinwang Liu, and Stan Z Li. Dink-net: Neural clustering on large graphs. In *International Conference on Machine Learning*, pages 21794–21812. PMLR, 2023.

[32] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. S2gae: self-supervised graph autoencoders are generalizable learners with graph masking. In *Proceedings of the sixteenth ACM international conference on web search and data mining*, pages 787–795, 2023.

[33] 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.

[34] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Advances in Neural Information Processing Systems*, 35:5376–5389, 2022.

[35] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[36] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

[37] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

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

[39] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[40] Teng Xiao, Huaisheng Zhu, Zhiwei Zhang, Zhimeng Guo, Charu C Aggarwal, and Suhang Wang. Graphecl: Towards efficient contrastive learning for graphs. 2023.

[41] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. pages 2623–2631, 2019.

[42] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[43] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pages 1073–1080, 2009.

[44] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

A Additional Details on Benchmark

A.1 Datasets

Cora, Citeseer and Pubmed [37] are three citation networks commonly used in prior GSSL works [28, 27, 40, 24, 25, 32]. In these datasets, nodes represent academic papers, and edges denote citation relationships between the papers. Each node's features are represented as bag-of-words vectors, and the label assigned to each node corresponds to its research topic category.

Flickr [38] is a social network dataset where nodes represent users and edges represent interactions between users (such as comments and likes). Node features are metadata attributes derived from users' photos. The label of each node is not predefined, making it suitable for tasks like link prediction and node clustering.

Reddit [35] is a social network dataset where nodes represent posts and edges represent comments linking the posts. Node features are 602-dimensional vectors representing various attributes of the posts, such as word embeddings. The label of each node corresponds to the community or subreddit to which the post belongs, with 41 different classes in total.

Ogbn-arxiv [39] is a citation network dataset from the Open Graph Benchmark (OGB) suite. Nodes represent papers from the arXiv repository, and edges represent citation relationships between papers. Node features are 128-dimensional vectors representing word2vec embeddings of paper abstracts. The label of each node is the subject area of the paper, with 40 different categories in total.

A.2 GSSL models

BGRL [28] is a contrastive learning model that focuses on learning node representations by maximizing agreement between different views of the same graph. It leverages bootstrapping techniques to create positive and negative samples, ensuring robust and informative embeddings.

CCA-SSG [27] applies canonical correlation analysis to graph data for self-supervised learning. The method aims to find representations that maximize the correlation between two sets of views from the graph, promoting the extraction of common features and enhancing the quality of node embeddings.

GBT [26] is a self-supervised learning model specifically designed for graph-structured data. Inspired by the Barlow Twins framework from computer vision, GBT aims to learn meaningful node representations by maximizing the similarity between different augmented views of the same graph while minimizing redundancy between feature dimensions.

GCA [33] is a graph contrastive learning method that generates augmented views of the graph and uses these views to learn node representations. It optimizes the agreement between the embeddings of the original and augmented graphs, helping the model to generalize better across different tasks.

GraphECL [40] is an advanced contrastive learning model that enhances the basic framework by incorporating additional graph structural information. It improves the quality of learned embeddings by leveraging both node attributes and structural features, making it effective for various graph-based tasks.

GraphMAE [24] is inspired by the success of masked autoencoders in NLP. It masks a portion of the graph data (such as node features or edges) and trains the model to reconstruct the masked parts. This approach helps in learning robust and informative node representations without relying on labeled data.

GraphMAE2 [25] builds on the original GraphMAE, introducing enhancements to the masking and reconstruction mechanisms. It may involve more sophisticated masking strategies, improved network architectures, or additional training objectives to further enhance the quality of learned embeddings.

S2GAE [32] is a generative model that uses autoencoders for graph data. It employs self-supervised learning techniques to train the autoencoder to reconstruct the graph from its latent representation. This process helps in capturing the underlying structure and features of the graph, making the embeddings useful for downstream tasks like node classification and clustering.

Table 7: Hyper-parameter search space of all implemented methods.

Models	Hyper-parameter	Search Space
General Settings	lr	[1e-6, 1e-2]
	weight_decay	[1e-6, 1e-2]
	batch_size	512, 1024, 2048, 4096, 10000, 20000
	decode_channels_lp	128, 256, 512, 1024
	decode_layers_lp	1, 2, 4, 8
BGRL [28]	drop_edge_p_1	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	drop_edge_p_2	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	drop_feat_p_1	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	drop_feat_p_2	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
CCA-SSG [27]	dfr	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	der	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	hid_dim	128, 256, 512, 1024
GBT [33]	emb_dim	128, 256, 512, 1024
	lr_base	[1e-6, 1e-2]
	p_x	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	p_e	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
GCA [26]	num_hidden	128, 256, 512, 1024
	drop_edge_rate_1	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	drop_edge_rate_2	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	drop_feature_rate_1	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
	drop_feature_rate_2	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6
GraphECL [40]	hid_dim	128, 256, 512, 1024, 2048
	n_layers	[1, 4]
	temp	0.4, 0.5, 0.6, 0.7, 0.8
	lam	[1e-6, 1e-2]
GraphMAE [24]	num_heads	1, 2, 4, 8
	num_hidden	256, 512, 1024
	attn_drop	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	in_drop	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	negative_slope	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	mask_rate	0.4, 0.5, 0.6, 0.7, 0.8
	drop_edge_rate	0.0, 0.05, 0.15, 0.20
	$alpha_l$	1, 2, 3
GraphMAE2 [25]	num_heads	1, 2, 4, 8
	num_hidden	256, 512, 1024
	attn_drop	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	in_drop	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	negative_slope	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	mask_rate	0.4, 0.5, 0.6, 0.7, 0.8
	drop_edge_rate	0.0, 0.05, 0.15, 0.20
	$alpha_l$	1, 2, 3
	replace_rate	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
	lam	0.0, 0.1, 0.2, 0.3, 0.4, 0.5
S2GAE [32]	dim_hidden	128, 256, 512, 1024
	decode_channels	128, 256, 512, 1024
	decode_layers	[1, 8]
	mask_ratio	0.4, 0.5, 0.6, 0.7, 0.8

B Additional Experimental Details

B.1 RQ1

General Experimental Settings. We strive to adhere to the original implementation of various GSSL models provided in their provided in their respective papers or source codes. To achieve this, we have integrated different options into a standardized framework as shown in Figure 1. To ensure fairness and consistency, we have standardized the optimizer as well as the evaluation methods for node classification, link prediction, and node clustering. Additionally, we have adopted the method of splitting edges for link prediction and adhered to the data splitting approach used in PyG [42].

Hyperparameter. We conduct comprehensive hyperparameter tuning through Optuna [41] to ensure a thorough and impartial evaluation of these GSSL models. The hyperparameter search spaces of all models are presented in Table 7, the notation "[]" indicates the range for hyperparameter tuning, while the absence of brackets denotes specific values used in the search. For detailed meanings of these hyperparameters, please refer to their original papers.

B.2 RQ2

In our link prediction tasks, we use AUC (Area Under the Curve) and AP (Average Precision) as metrics, while for node clustering, we employ NMI (Normalized Mutual Information) and ARI (Adjusted Rand Index) [43]. These metrics are widely recognized as effective for these respective tasks [21]. The following are the details for these metrics.

B.2.1 AUC

AUC measures the ability of the model to distinguish between positive and negative edges. It is calculated as the area under the Receiver Operating Characteristic (ROC) curve.

$$\text{AUC} = \int_0^1 \text{TPR}(FPR)\, d(\text{FPR}), \tag{1}$$

where TPR (True Positive Rate) and FPR (False Positive Rate) are defined as:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

B.2.2 AP

AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight.

$$\text{AP} = \sum_n (R_n - R_{n-1}) P_n, \tag{2}$$

where P_n and R_n are the precision and recall at the n-th threshold.

B.2.3 NMI

NMI measures the similarity between the clustering of the nodes and the ground truth labels. It is defined as:

$$\text{NMI}(U, V) = \frac{I(U; V)}{\sqrt{H(U)H(V)}}, \tag{3}$$

where $I(U; V)$ is the mutual information between the cluster assignments U and V, and $H(U)$ and $H(V)$ are the entropies of U and V, respectively.

$$I(U;V) = \sum_{u \in U} \sum_{v \in V} p(u,v) \log \frac{p(u,v)}{p(u)p(v)}$$
$$H(U) = -\sum_{u \in U} p(u) \log p(u)$$

B.2.4 ARI

The ARI measures the similarity between two data clusterings, corrected for chance. It is defined as:

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{0.5 \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}, \tag{4}$$

where n_{ij} is the number of elements in both cluster i of the true clustering and cluster j of the predicted clustering, a_i is the number of elements in cluster i, b_j is the number of elements in cluster j, and n is the total number of elements.

B.3 RQ3

In our experiments, we selected Node Sampling [35] and Subgraph Sampling [36] two sampling strategies. The formula mentioned in Section 2 is the general formula for message passing. In the mini-batch setting, the function will be as follows:

$$\mathbf{X}_{\mathcal{B}_0}^{(k)} = \tilde{\mathbf{A}}_{\mathcal{B}_1}^{(k-1)} \sigma \left(\tilde{\mathbf{A}}_{\mathcal{B}_2}^{(k-2)} \sigma \left(\cdots \sigma \left(\tilde{\mathbf{A}}_{\mathcal{B}_K}^{(0)} \mathbf{X}_{\mathcal{B}_K}^{(0)} \mathbf{W}^{(0)} \right) \cdots \right) \mathbf{W}^{(K-2)} \right) \mathbf{W}^{(K-1)}$$

where $\mathcal{B}_l$ is the set of sampled nodes for the l-th layer, and $\tilde{\mathbf{A}}^{(l)}$ is the adjacency matrix for the l-th layer sampled from the full graph. The key difference among different sampling methods is how $\{\mathcal{B}_0, \ldots, \mathcal{B}_{K-1}, \mathcal{B}_K\}$ are sampled, the following are the details for these two methods.

B.3.1 Node Sampling

$\mathcal{B}_{l+1} = \bigcup_{v \in \mathcal{B}_l} \{u \mid u \sim Q \cdot \mathbb{P}_{\mathcal{N}(v)}\}$, where $\mathbb{P}$ is a uniform distribution; $\mathcal{N}(v)$ is the sampling space, i.e., the $1-$hop neighbors of v; and Q denotes the number of samples.

B.3.2 Subgraph Sampling

$\mathcal{B}_K = \mathcal{B}_{K-1} = \cdots = \mathcal{B}_0 = \{u \mid u \sim Q \cdot \mathbb{P}_{\mathcal{G}}\}$. In the subgraph-wise sampling, all layers share the same subgraph induced from the entire graph G based on a specific sampling strategy $\mathbb{P}_{\mathcal{G}}$, such that the sampled nodes are confined in the subgraph. ClusterGCN [36] first partitions the entire graph into clusters based on some graph partition algorithms, e.g., METIS [44], and then selects several clusters to form a batch.

B.4 RQ4

To explore the impact of different metrics as early stopping criteria on model performance, we conducted experiments by replacing the usual accuracy metric in node classification with AUC for link prediction and NMI for node clustering.

C Additional Results

Running Experiments. Our experiments are mostly conducted on a Linux server with Lenovo SR670, and an NVIDIA RTX8000 GPU (48G).

C.1 Result of Full Batch

In Section 4.2, we analyzed the outcomes of various downstream tasks that differ from saving the best validation model. The results of node clustering are recorded in Figure 4, and the analysis is detailed in Section 4.2.

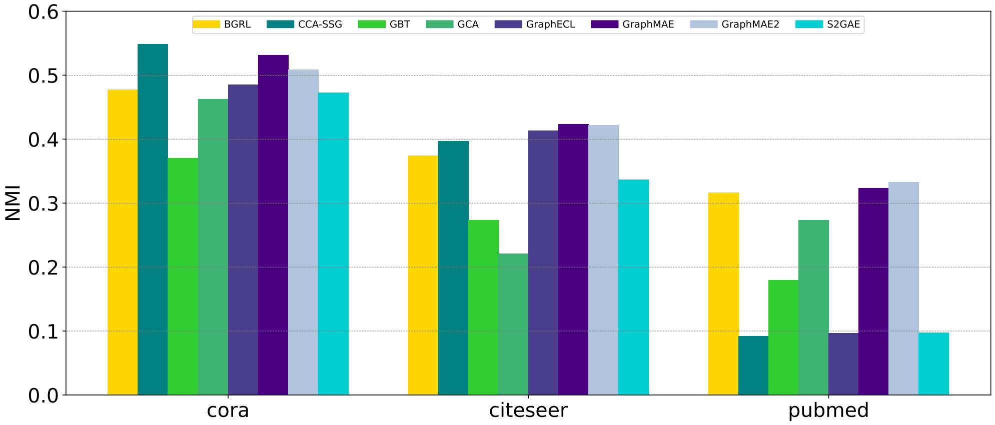

Figure 4: Node Clustering results on Cora, Citeseer, Pubmed based on full batch training.

C.2 Result of Mini-Batch

In Section 4.3, we analyzed the performance of GraphFM using the mini-batch training strategy across various datasets. Additional results are recorded in Tables 8, 9, 10, 11, 12 and 13.

From the table, we can observe that S2GAE, through its mini-batch training strategy, does not perform as well as other models in node classification on large-scale datasets. However, it typically performs better in link prediction, and its performance in node clustering tasks is not significantly different from other models. This indicates that S2GAE's model scalability is related to the downstream tasks, exhibiting stronger scalability in link prediction tasks.

Table 8: The result of GraphFM in Node classification with Node sampling. " - " means out of memory.

Models	cora	citeseer	pubmed	Flickr	Reddit	Arxiv
BGRL	83.69±0.20	70.12±0.53	83.70±0.07	47.37±0.05	93.16±0.03	65.20±0.57
CCA-SSG	86.29±0.18	71.58±0.13	83.51±0.14	51.59±0.14	93.51±0.86	67.16±0.18
GBT	82.71±0.64	68.56±1.32	84.23±0.19	52.11±0.08	92.17±0.09	61.60±0.25
GCA	86.05±0.20	72.92±0.52	82.29±0.14	-	-	-
GraphECL	78.11±0.25	65.52±0.23	83.10±0.04	-	-	-
GraphMAE	83.96±0.69	70.63±0.18	83.60±0.06	49.25±0.13	94.30±0.04	66.33±0.11
GraphMAE2	77.52±0.52	64.77±0.78	80.76±0.18	46.07±0.83	92.66±0.30	65.45±0.01
S2GAE	77.22±0.88	64.85±1.25	81.40±0.37	43.90±0.17	62.73±0.84	56.58±0.71

Table 9: The result of GraphFM in Node classification with Subgraph sampling. " - " means out of memory.

Models	Cora	Citeseer	Pubmed	Flickr	Reddit	Arxiv
BGRL	83.69±0.70	70.75±0.41	84.52±0.14	47.14±0.07	93.39±0.08	64.68±0.12
CCA-SSG	84.90±0.95	71.96±0.25	84.83±0.16	50.95±0.20	93.53±0.42	66.62±0.17
GBT	83.09±0.61	68.11±1.59	82.61±0.13	51.00±0.17	92.25±0.11	60.10±0.04
GCA	85.71±0.26	70.24±0.54	84.43±0.31	51.76±0.08	91.46±0.33	62.87±0.43
GraphECL	84.29±0.46	72.16±0.20	84.59±0.37	46.23±0.09	-	58.51±0.11
GraphMAE	85.72±0.77	72.51±0.50	84.85±0.17	45.30±0.85	94.41±0.14	67.33±0.05
GraphMAE2	78.32±1.01	64.45±0.35	80.24±0.25	46.25±0.90	91.86±0.58	65.71±0.01
S2GAE	81.12±0.71	64.68±0.19	79.51±0.66	43.97±0.20	62.19±1.12	46.45±0.08

C.3 Results of Efficiency

In Section 4.3, in addition to analyzing the performance of GraphFM under mini-batch conditions, we also examined the training efficiency of GraphFM. Further results are documented in table 14 and 15,

Table 10: The result of `GraphFM` in Link prediction with Node sampling. " - " means out of memory.

Models	Metrics	Cora	Citeseer	Pubmed	Flickr	Reddit	Arxiv
BGRL	AUC	98.53±1.07	99.41±0.61	99.60±0.04	87.88±0.24	42.72±3.84	96.98±0.42
	AP	98.75±0.74	99.53±0.38	99.52±0.05	88.24±0.18	59.22±0.95	96.08±0.40
CCA-SSG	AUC	99.64±0.11	99.89±0.10	99.58±0.11	76.45±14.44	20.17±0.42	45.28±0.52
	AP	99.63±0.14	99.87±0.12	99.49±0.16	73.96±17.05	45.62±0.72	58.44±0.41
GBT	AUC	99.22±0.10	99.74±0.10	99.40±0.10	86.69±0.42	46.86±0.44	57.33±0.64
	AP	99.08±0.34	99.72±0.16	99.31±0.15	87.93±0.31	50.96±1.49	60.61±0.70
GCA	AUC	98.80±0.22	99.18±0.18	99.17±0.08	-	-	-
	AP	98.58±0.30	99.11±0.26	99.08±0.12			
GraphECL	AUC	95.68±0.85	96.22±0.33	95.28±0.40	-	-	-
	AP	95.44±1.15	96.61±0.38	94.67±0.48			
GraphMAE	AUC	97.28±0.29	99.33±0.43	96.04±0.09	50.00±0.00	72.49±0.43	50.01±0.02
	AP	97.05±0.51	99.22±0.55	94.72±0.33	50.00±0.00	66.18±0.33	50.00±0.01
GraphMAE2	AUC	89.66±0.55	95.05±0.48	84.73±0.99	49.94±0.05	50.08±0.07	50.00±0.00
	AP	90.80±0.91	95.08±0.11	85.37±0.85	49.98±0.03	50.04±0.03	50.00±0.00
S2GAE	AUC	95.16±0.30	95.75±0.12	88.77±0.64	49.95±0.44	90.75±1.12	94.97±0.69
	AP	95.47±0.29	96.45±0.06	86.33±0.63	49.93±0.32	90.08±0.76	94.97±0.69

Table 11: The result of `GraphFM` in Link prediction with Subgraph sampling. " - " means out of memory.

Models	Metrics	Cora	Citeseer	Pubmed	Flickr	Reddit	Arxiv
BGRL	AUC	97.18±1.96	99.81±0.09	99.47±0.07	86.92±0.15	21.99±0.23	93.81±3.20
	AP	97.80±1.49	99.79±0.13	99.39±0.09	87.57±0.15	43.42±0.16	91.20±3.72
CCA-SSG	AUC	99.91±0.00	98.23±0.29	99.69±0.00	50.00±0.00	21.22±0.08	33.25±0.64
	AP	99.91±0.00	97.85±0.24	99.62±0.04	50.00±0.00	45.88±0.54	49.22±2.56
GBT	AUC	99.13±0.07	99.83±0.11	98.70±0.14	50.00±0.00	44.72±0.07	45.70±2.80
	AP	99.03±0.09	99.75±0.23	98.83±0.05	50.00±0.00	48.01±0.33	50.45±2.79
GCA	AUC	98.61±0.21	99.97±0.03	98.85±0.25	50.00±0.00	51.88±0.74	39.38±2.34
	AP	98.27±0.31	99.97±0.03	98.80±0.22	50.00±0.00	50.97±0.82	50.10±1.43
GraphECL	AUC	98.98±0.19	99.65±0.27	96.23±0.14	51.06±0.35	-	59.01±0.54
	AP	98.62±0.60	99.64±0.28	95.69±0.31	51.12±0.41		60.47±0.63
GraphMAE	AUC	96.63±0.79	99.65±0.20	95.86±0.25	50.00±0.00	77.07±1.42	91.50±0.56
	AP	96.50±0.80	99.64±0.21	94.64±0.49	50.00±0.00	68.59±0.82	88.07±0.64
GraphMAE2	AUC	89.62±0.54	94.92±0.57	83.99±1.02	49.94±0.05	50.00±0.00	99.01±0.00
	AP	90.76±0.91	94.98±0.57	84.79±0.88	49.98±0.03	50.00±0.00	98.85±0.02
S2GAE	AUC	95.92±1.28	96.76±0.49	89.72±0.37	49.91±0.23	80.17±0.93	79.00±1.31
	AP	94.76±1.71	96.22±0.70	88.09±0.44	49.91±0.28	78.99±1.32	78.53±1.31

Table 12: The result of `GraphFM` in Node clustering with Node sampling. " - " means out of memory.

Models	Cora	Citeseer	Pubmed	Flickr	Reddit	Arxiv
BGRL	0.3719/0.2217	0.1883/0.0628	0.1139/0.0588	0.0054/0.0145	0.5855/0.2043	0.2077/0.0472
CCA-SSG	0.5107/0.4398	0.3525/0.2607	0.1246/0.0625	0.0622/0.0397	0.5560/0.1701	0.2868/0.0769
GBT	0.3441/0.1497	0.1466/0.0180	0.0453/-0.0084	0.0179/0.0080	0.5832/0.1680	0.2029/-0.0058
GCA	0.4591/0.3781	0.3229/0.2906	0.0708/0.0150	-	-	-
GraphECL	0.4620/0.3262	0.3032/0.2123	0.1056/0.0199	-	-	-
GraphMAE	0.5348/0.4476	0.3510/0.3042	0.1125/0.0250	0.0154/0.0197	0.8139/0.7313	0.3899/0.1848
GraphMAE2	0.3923/0.3286	0.2697/0.2714	0.2770/0.2594	0.0157/0.0097	0.6408/0.5023	0.3739/0.1747
S2GAE	0.3457/0.2018	0.2179/0.1044	0.2647/0.2757	0.0067/0.0054	0.4464/0.2648	0.2800/0.1199

from the table, we can observe that compared to the other two training strategies, subgraph sampling requires less memory and provides faster training speeds. In terms of model comparison, S2GAE achieves better training efficiency across all benchmarked tasks.

Table 13: The result of `GraphFM` in Node clustering with Subgraph sampling (NMI/ARI). " - " means out of memory.

Models	Cora	Citeseer	Pubmed	Flickr	Reddit	Arxiv
BGRL	0.2589/0.1143	0.3102/0.1593	0.2272/0.1830	0.0052/0.0145	0.6227/0.1944	0.2123/0.0441
CCA-SSG	0.2165/0.1258	0.1591/0.0265	0.2482/0.2214	0.0181/0.0125	0.5441/0.1764	0.2959/0.0846
GBT	0.3657/0.1944	0.1373/0.0156	0.0638/-0.0018	0.0453/0.0176	0.6073/0.1610	0.2069/-0.0173
GCA	0.4609/0.3277	0.2509/0.2125	0.0888/0.0182	0.0343/0.0155	0.5330/0.1709	0.1732/0.0019
GraphECL	0.5568/0.5186	0.3891/0.3663	0.3442/0.3057	0.0128/0.0142	-	0.3290/0.1296
GraphMAE	0.5454/0.4424	0.4181/0.3975	0.3211/0.2839	0.0090/0.0137	0.7988/0.6419	0.4146/0.2035
GraphMAE2	0.3909/0.3271	0.2706/0.2695	0.2773/0.2598	0.0157/0.0068	0.4640/0.2320	0.2577/0.1149
S2GAE	0.4121/0.2735	0.2762/0.2074	0.3001/0.2865	0.0039/0.0033	0.3906/0.2112	0.2212/0.0666

Table 14: The memory usage of activations and the hardware throughput (higher is better).

	Batch Type	Cora		Citeseer		Pubmed	
		Act Mem. (MB)	Throughput (iteration/s)	Act Mem. (MB)	Throughput (iteration/s)	Act Mem. (MB)	Throughput (iteration/s)
BGRL	Full	115.02	57.80	200.28	33.97	695.89	3.07
	Node	101.48	51.02	126.90	34.68	411.47	28.17
	Subgraph	24.55	85.48	24.14	91.17	22.75	86.34
CCA-SSG	Full	123.70	4.36	263.72	2.24	552.28	2.39
	Node	69.49	6.42	80.91	3.65	90.34	2.16
	Subgraph	27.67	47.71	25.97	67.19	13.97	68.03
GBT	Full	49.54	50.76	66.43	23.64	338.26	21.18
	Node	65.04	58.82	76.14	51.02	93.29	64.51
	Subgraph	95.04	54.64	174.47	28.73	82.43	37.59
GCA	Full	430.84	17.94	354.45	9.40	8117.57	1.82
	Node	364.88	6.01	351.15	3.00	3356.76	2.81
	Subgraph	50.75	37.96	91.37	58.41	44.39	38.68
GraphECL	Full	155.88	43.69	265.64	6.77	3554.12	1.72
	Node	92.59	24.75	242.54	12.24	185.10	5.68
	Subgraph	29.93	55.91	48.11	61.34	29.49	57.95
GraphMAE	Full	142.08	48.08	370.15	29.88	577.61	27.24
	Node	103.78	32.26	175.52	37.04	96.23	21.01
	Subgraph	26.45	54.05	42.25	114.58	10.02	66.23
GraphMAE2	Full	146.55	39.37	345.08	24.60	667.16	19.30
	Node	57.17	37.04	86.15	23.20	63.95	22.52
	Subgraph	18.31	48.77	44.92	102.30	13.61	60.24
S2GAE	Full	28.09	26.18	35.43	1.38	205.03	2.30
	Node	7.63	80.00	6.93	70.92	237.02	10.92
	Subgraph	2.87	128.21	2.99	106.32	2.66	114.84

C.4 Results of Early Stop Criteria

In Section 4.4, we analyzed the performance of `GraphFM` when different downstream tasks were used as early stopping criteria. More experimental results are documented in Tables 16, 17 and 18. Due to tests on the Reddit dataset typically taking more than 24 hours, this study primarily conducts tests on the Flickr and Ogbn-arxiv datasets. We use AUC in link prediction and NMI in node clustering as the metrics to preserve the valid model, and the analysis is detailed in Section 4.4.

Table 15: The memory usage of activations and the hardware throughput. " - " means out of memory.

	Batch Type	Flickr		Reddit		Arxiv	
		Act Mem. (MB)	Throughput (iteration/s)	Act Mem. (MB)	Throughput (iteration/s)	Act Mem. (MB)	Throughput (iteration/s)
BGRL	Node	1379.81	6.64	2490.87	2.56	2476.28	4.47
	Subgraph	22.61	126.58	21.63	89.29	87.26	37.73
CCA-SSG	Node	612.13	2.27	1496.45	0.81	2841.26	1.12
	Subgraph	13.90	58.82	15.05	42.74	31.84	0.001
GBT	Node	1801.95	5.14	4335.30	1.74	3030.22	1.72
	Subgraph	219.11	37.87	271.24	8.25	224.10	24.75
GCA	Node	-	-	-	-	-	-
	Subgraph	44.80	35.34	36.82	4.75	264.96	0.79
GraphECL	Node	-	-	-	-	-	-
	Subgraph	22.14	34.97	-	-	20.16	19.65
GraphMAE	Node	638.83	5.54	1293.80	1.57	399.42	6.71
	Subgraph	10.36	63.69	12.38	47.39	6.49	21.50
GraphMAE2	Node	1039.38	3.29	2525.84	1.63	906.31	6.24
	Subgraph	16.29	48.54	22.80	46.73	14.14	22.27
S2GAE	Node	165.50	17.15	380.07	1.42	179.94	2.92
	Subgraph	2.62	135.14	0.87	136.99	2.61	43.29

Table 16: The result of `GraphFM` in Flickr dataset by saving valid model with the best performance in node clustering. " - " means out of memory.

Training Strategy	Models	Node Classification	Link Prediction	Node Clustering
Node Sampling	BGRL	46.33±0.94	86.56/87.33	0.0094/0.0181
	CCA-SSG	51.75±1.22	90.08/90.56	0.0850/0.0583
	GBT	51.70±0.86	97.21/97.39	0.0453/0.0196
	GCA	-	-	-
	GraphECL	-	-	-
	GraphMAE	46.68±0.74	50.00/50.00	0.0296/0.0282
	GraphMAE2	46.54±0.68	50.00/50.00	0.0266/0.0081
	S2GAE	45.13±0.32	50.65/50.41	0.0246/0.0175
Subgraph Sampling	BGRL	46.32±0.78	86.86/87.52	0.0082/0.0183
	CCA-SSG	49.51±1.03	55.39/52.86	0.0986/0.0579
	GBT	52.36±0.32	91.08/91.70	0.0699/0.0328
	GCA	49.87±0.88	55.45/53.18	0.0887/0.0393
	GraphECL	46.42±1.20	60.70/60.39	0.0526/0.0466
	GraphMAE	46.46±0.94	50.00/50.00	0.0261/0.0099
	GraphMAE2	45.86±0.09	50.10/50.05	0.0249/0.0109
	S2GAE	44.76±0.43	50.25/50.30	0.0068/0.0049

Table 17: The result of GraphFM in ogbn-arxiv dataset by saving valid model with the best performance in link prediction. " - " means out of memory.

Training Strategy	Models	Node Classification	Link Prediction	Node Clustering
Node Sampling	BGRL	64.97±0.61	98.98/98.95	0.2058/0.0415
	CCA-SSG	67.08±0.43	99.33/99.30	0.2801/0.0429
	GBT	62.70±1.07	98.78/98.69	0.2220/-0.0166
	GCA	-	-	-
	GraphECL	-	-	-
	GraphMAE	65.64±0.93	90.85/87.85	0.3971/0.1851
	GraphMAE2	64.82±0.72	50.09/50.04	0.3760/0.1873
	S2GAE	43.65±2.11	80.50/78.61	0.2064/0.0708
Subgraph Sampling	BGRL	64.63±0.87	99.11/99.09	0.2198/0.0549
	CCA-SSG	61.90±0.73	97.58/97.60	0.2697/0.0476
	GBT	56.44±2.13	87.57/80.41	0.1300/-0.0116
	GCA	59.99±0.82	96.00/95.46	0.2496/0.0070
	GraphECL	55.42±0.95	93.38/93.16	0.3114/0.1375
	GraphMAE	66.43±0.53	89.70/86.17	0.4146/0.1985
	GraphMAE2	64.42±1.01	71.96/64.71	0.3732/0.1763
	S2GAE	38.06±2.43	77.50/76.84	0.1663/0.0426

Table 18: The result of GraphFM in ogbn-arxiv dataset by saving valid model with the best performance in node clustering. " - " means out of memory.

Training Strategy	Models	Node Classification	Link Prediction	Node Clustering
Node Sampling	BGRL	64.64±0.43	98.68/98.57	0.2384/0.0605
	CCA-SSG	43.69±0.73	94.32/92.15	0.2381/0.0305
	GBT	59.40±0.71	84.57/78.01	0.2130/-0.0168
	GCA	-	-	-
	GraphECL	-	-	-
	GraphMAE	68.56±0.94	88.71/85.40	0.4138/0.1976
	GraphMAE2	67.58±0.08	50.00/50.00	0.3950/0.1871
	S2GAE	52.19±1.45	86.83/85.92	0.3126/0.1237
Subgraph Sampling	BGRL	64.17±0.46	98.39/98.21	0.2268/0.0572
	CCA-SSG	49.97±0.62	93.89/93.93	0.3358/0.0787
	GBT	59.85±1.34	50.00/50.00	0.1903/-0.0146
	GCA	61.64±0.95	93.76/89.32	0.2696/-0.0002
	GraphECL	54.04±1.42	92.97/92.70	0.3003/0.1406
	GraphMAE	68.13±0.53	91.31/87.71	0.4055/0.1861
	GraphMAE2	67.57±0.87	50.00/50.00	0.3919/0.1783
	S2GAE	46.50±1.16	83.01/81.88	0.2538/0.0805