

When to Pre-Train Graph Neural Networks? From Data Generation Perspective!

Yuxuan Cao*[†]
Zhejiang University, Fudan University
caoyx@zju.edu.cn

Jiarong Xu*[‡]
Fudan University
jiarongxu@fudan.edu.cn

Carl Yang
Emory University
j.carlyang@emory.edu

Jiaan Wang
Soochow University
jawang.nlp@gmail.com

Yunchao Zhang
Zhejiang University
m.yunchaozhang@gmail.com

Chunping Wang
Finvolution Group
wangchunping02@xinye.com

Lei Chen
Finvolution Group
chenlei04@xinye.com

Yang Yang
Zhejiang University
yangya@zju.edu.cn

ABSTRACT

In recent years, graph pre-training has gained significant attention, focusing on acquiring transferable knowledge from unlabeled graph data to improve downstream performance. Despite these recent endeavors, the problem of negative transfer remains a major concern when utilizing graph pre-trained models to downstream tasks. Previous studies made great efforts on the issue of *what to pre-train* and *how to pre-train* by designing a variety of graph pre-training and fine-tuning strategies. However, there are cases where even the most advanced “pre-train and fine-tune” paradigms fail to yield distinct benefits. This paper introduces a generic framework W2PGNN to answer the crucial question of *when to pre-train* (*i.e.*, in what situations could we take advantage of graph pre-training) before performing effortful pre-training or fine-tuning. We start from a new perspective to explore the complex generative mechanisms from the pre-training data to downstream data. In particular, W2PGNN first fits the pre-training data into graphon bases, each element of graphon basis (*i.e.*, a graphon) identifies a fundamental transferable pattern shared by a collection of pre-training graphs. All convex combinations of graphon bases give rise to a generator space, from which graphs generated form the solution space for those downstream data that can benefit from pre-training. In this manner, the feasibility of pre-training can be quantified as the generation probability of the downstream data from any generator in the generator space. W2PGNN offers three broad applications: providing the application scope of graph pre-trained models, quantifying the feasibility of pre-training, and assistance in selecting pre-training data to enhance downstream performance. We provide

*Both authors contributed equally to this research.

[†]This work was done when the author was a visiting student at Fudan University.

[‡]Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 979-8-4007-0103-0/23/08...\$15.00

<https://doi.org/10.1145/3580305.3599548>

a theoretically sound solution for the first application and extensive empirical justifications for the latter two applications.

CCS CONCEPTS

• **Networks** → **Network algorithms**.

KEYWORDS



graph neural networks, graph pre-training

ACM Reference Format:

Yuxuan Cao, Jiarong Xu, Carl Yang, Jiaan Wang, Yunchao Zhang, Chunping Wang, Lei Chen, and Yang Yang. 2023. When to Pre-Train Graph Neural Networks? From Data Generation Perspective!. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*, August 6–10, 2023, Long Beach, CA, USA. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3580305.3599548>

1 INTRODUCTION

Graph neural networks (GNNs) have undergone rapid development and become increasingly popular for learning graph data [37, 39, 44]. GNNs are usually trained in an end-to-end manner while getting enough labeled data is arduously expensive and sometimes even impractical to access. This motivates some recent advances in pre-training GNNs [14, 15, 21, 24, 31]. The key insight of pre-training GNNs is to learn transferable knowledge from a collection of unlabeled graph data, hoping that the learned knowledge can be easily adapted to downstream tasks. In view of the great success of pre-training in other fields like computer vision and natural language processing [4, 12], graph pre-training is highly expected to be an effective means to improve downstream performance.

However, the intuition that graph pre-trained model would ideally benefit the downstream is far from the truth in the area of graph pre-training. Instead, graph pre-trained models can lead to *negative transfer* on many downstream tasks, especially when the graphs used for pre-training are not necessarily from the same domain as the downstream data [14, 31]. For example, the closed triangles () and open triangles () might yield different interpretations in molecular networks (unstable vs. stable in terms of chemical property) from those in social networks (stable vs. unstable in terms of social relationship); such distinct or reversed semantics does not

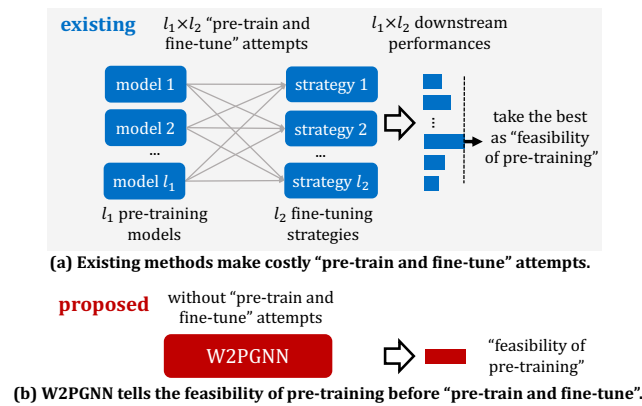


Figure 1: Comparison of existing methods and proposed W2PGNN to answer *when to pre-train* GNNs.

contribute to transferability, and even exacerbates the problem of negative transfer.

To avoid the negative transfer, recent efforts focus on *what to pre-train* and *how to pre-train*, *i.e.*, design/adopt graph pre-training models with a variety of self-supervised tasks to capture different patterns [24, 31, 46] and fine-tuning strategies to enhance downstream performance [9, 16, 42, 49]. However, there do exist some cases that no matter how advanced the pre-training/fine-tuning method is, the transferability from pre-training data to downstream data still cannot be guaranteed. This is because the underlying assumption of deep learning models is that the test data should share a similar distribution as the training data. Therefore, it is a necessity to understand *when to pre-train*, *i.e.*, under what situations the “graph pre-train and fine-tune” paradigm should be adopted.

Towards the answer of when to pre-train GNNs, one straightforward way illustrated in Figure 1(a) is to train and evaluate on all candidates of pre-training models and fine-tuning strategies, and then the resulting best downstream performance would tell us whether pre-training is a sensible choice. If there exist l_1 pre-training models and l_2 fine-tuning strategies, such a process would be very costly as you should make $l_1 \times l_2$ “pre-train and fine-tune” attempts. Another approach is to utilize graph metrics to measure the similarity between pre-training and downstream data, *e.g.*, density, clustering coefficient and etc. However, it is a daunting task to enumerate all hand-engineered graph features or find the dominant features that influenced similarity. Moreover, the graph metrics only measure the pair-wise similarity between two graphs, which cannot be directly and accurately applied to the practical scenario where pre-training data contains multiple graphs.

In this paper, we propose a W2PGNN framework to answer *when to pre-train GNNs from a graph data generation perspective*. The high-level idea is that instead of performing effortful graph pre-training/fine-tuning or making comparisons between the pre-training and downstream data, we study the complex generative mechanism from the pre-training data to the downstream data (Figure 1(b)). We say that downstream data can benefit from pre-training data (*i.e.*, has high feasibility of performing pre-training), if it can be generated with high probability by a graph generator that summarizes the topological characteristic of pre-training data.

The major challenge is how to obtain an appropriate graph generator, hoping that it not only inherits the transferable topological patterns of the pre-training data, but also is endowed with the ability to generate feasible downstream graphs. To tackle the challenge, we propose to design a graph generator based on graphons. We first fit the pre-training graphs into different graphons to construct a *graphon basis*, where each graphon (*i.e.*, element of the graphon basis) identifies a collection of graphs that share common transferable patterns. We then define a *graph generator* as a convex combination of elements in a graphon basis, which serves as a comprehensive and representative summary of pre-training data. All of these possible generators constitute the *generator space*, from which graphs generated form the solution space for the downstream data that can benefit from pre-training.

Accordingly, the feasibility of performing pre-training can be measured as the highest probability of downstream data being generated from any graph generator in the generator space, which can be formulated as an optimization problem. However, this problem is still difficult to solve due to the large search space of graphon basis. We propose to reduce the search space to three candidates of graphon basis, *i.e.*, topological graphon basis, domain graphon basis, and integrated graphon basis, to mimic different generation mechanisms from pre-training to downstream data. Built upon the reduced search space, the feasibility can be approximated efficiently.

Our major contributions are concluded as follows:

- **Problem and method.** To the best of our knowledge, we are the first work to study the problem of when to pre-train GNNs. We propose a W2PGNN framework to answer the question from a data generation perspective, which tells us the feasibility of performing graph pre-training before conducting effortful pre-training and fine-tuning.
- **Broad applications.** W2PGNN provides several practical applications: (1) provide the application scope of a graph pre-trained model, (2) measure the feasibility of performing pre-training for a downstream data and (3) choose the pre-training data so as to maximize downstream performance with limited resources.
- **Theory and Experiment.** We theoretically and empirically justify the effectiveness of W2PGNN. Extensive experiments on real-world graph datasets from multiple domains show that the proposed method can provide an accurate estimation of pre-training feasibility and the selected pre-training data can benefit the downstream performance.

2 PROBLEM FORMULATION

In this section, we first formally define the problem of when to pre-train GNNs. Then, we provide a brief theoretical analysis of the transferable patterns in the problem we study, and finally discuss some non-transferable patterns.

DEFINITION 1 (WHEN TO PRE-TRAIN GNNs). *Given the pretraining graph data \mathcal{G}_{train} and the downstream graph data \mathcal{G}_{down} , our main goal is to answer to what extent the “pre-train and fine-tune” paradigm can benefit the downstream data.*

Note that in addition to this main problem, our proposed framework can also serve other scenarios, such as providing the application scope of graph pre-trained models, and helping select

pre-training data to benefit the downstream (please refer to the *application cases* in Section 4.1 for details).

Transferable graph patterns. The success of “pre-train and fine-tune” paradigm is typically attributed to the commonplace between pre-training and downstream data. However, in real-world scenarios, there possibly exists a significant divergence between the pre-training data and the downstream data. To answer the problem of when to pre-train GNNs, the primary task is to define the transferable patterns across graphs.

We here theoretically explore which patterns are transferable between pre-training and downstream data under the performance guarantee of graph pre-training model (with GNN as the backbone).

THEOREM 2.1 (TRANSFERABILITY OF GRAPH PRE-TRAINING MODEL). *Let G_{train} and G_{down} be two (sub)graphs sampled from \mathcal{G}_{train} and \mathcal{G}_{down} , and assume the attribute of each node as a scalar 1 without loss of generality. Given a graph pre-training model e (instantiated as a GNN) with K layers and 1-hop graph filter $\Phi(L)$ (which is a function of the normalized graph Laplacian matrix L), we have*

$$\|e(G_{train}) - e(G_{down})\|_2 \leq \kappa \Delta_{topo}(G_{train}, G_{down}) \quad (1)$$

where $\Delta_{topo}(G_{train}, G_{down}) = \frac{1}{mn} \sum_{i=1}^m \sum_{j'=1}^n \|L_{g_i} - L_{g'_j}\|_2$ measures the topological divergence between G_{train} and G_{down} , where g_i is the K -hop ego-network of node i from G_{train} and L_{g_i} is its corresponding normalized graph Laplacian matrix, m and n are the number of nodes of G_{train} and G_{down} . $e(G_{train})$ and $e(G_{down})$ are the output representations of G_{train} and G_{down} from graph pre-training model, κ is a constant relevant to K , graph filter Φ , learnable parameters of GNN and the activation function used in GNN.

Theorem 2.1 suggests that two (sub)graphs sampled from pre-training and downstream data with similar topology are transferable via graph pre-training model (*i.e.*, sharing similar representations produced by the model). Hence we consider the transferable graph pattern as the topology of a (sub)graph, either node-level or graph-level. Specifically, the node-level transferable pattern could be the topology of the ego-network of a node (or the structural role of a node), irrespective of the node’s exact location in the graph. The graph-level transferable pattern is the topology of the entire graph itself (*e.g.*, molecular network). Such transferable patterns constitute the input space introduced in Section 4.1.

Discussion of non-transferable graph patterns. As a remark, we show that two important pieces of information (*i.e.*, attributes and proximity) commonly used in graph learning are not necessarily transferable across pre-training and downstream data in most real-world scenarios, thus we do not discuss them in this paper.

First, although the attributes carry important semantic meaning in one graph, it can be shown that the attribute space of different graphs typically has little or no overlap at all. For example, if the pre-training and downstream data come from different domains, their nodes would indicate different types of entities and the corresponding attributes may be completely irrelevant. Even for graphs from the similar/same domain, the dimensions/meaning of their node attributes can be totally different and result in misalignment.

The proximity, on the other hand, assumes that closely connected nodes are similar, which also cannot be transferred across graphs.

This assumption depends on the overlaps in neighborhoods and thus only works on graphs with the same or overlapped node set.

3 PRELIMINARY AND RELATED WORKS

Graphons. A Graphon (short for graph function) [1] is a bounded symmetric function $B : [0, 1]^2 \rightarrow [0, 1]$ (different subscripts of B denote different graphons), which can be interpreted as the weighted matrix of an arbitrary undirected graph with uncountable number of nodes[22]. Literately, graphon has been studied from two perspectives: as limit of graph sequence, and as graph generators[1, 10, 23]. *We utilize both perspectives in our framework.*

On one hand, a graphon can be considered as the limit objects of graph sequence, and every convergent graph sequence would converge to a graphon[22]. Thus, a graphon is a comprehensive summary of a collection of arbitrary size graphs. These graphs can be considered topologically similar in the sense that they belong to the same graphon. *In this paper, we utilize a set of graphons as a comprehensive and representative summary of pre-training data.*

Taking graphon as a graph generator, we can associate nodes i and j with points v_i and v_j in $[0, 1]$, and then $B(v_i, v_j)$ serves as the probability to generate the edge between these two nodes. Therefore, a graphon B can generate unweighted graphs of arbitrary sizes, which can be taken as those induced graphs potentially inheriting the topological patterns implied in graphon. *Thus, the generation capability of graphon can help us generate feasible downstream graphs that can benefit from pre-training.*

Graph pre-training and fine-tuning. Graph pre-training models first learn universal knowledge from large-scale graph datasets with self-supervised or unsupervised objectives (*i.e.*, pre-training stage), and then transfer the knowledge to deal with specific downstream tasks (*i.e.*, fine-tuning stage). Among them, some researchers design pre-training tasks based on the neighborhood similarity assumption [5, 7, 18, 29, 34, 36, 48, 52]. The learned graph-specific patterns/knowledge could benefit downstream tasks on the same graphs, but cannot be generalized to unseen graphs. To enhance the transferability of the pre-trained models, some works try to utilize graph data from the same (or similar) domains as the pre-training data [8, 11, 14, 15, 19, 24, 27, 33, 35, 46, 47, 50, 54], or explore cross-domain pre-training strategies [24, 31, 46] as well as fine-tuning strategies [9, 16, 42, 49]. Nevertheless, all these efforts focus on addressing the problem of *what to pre-train and how to pre-train* by developing pre-training or fine-tuning methods. To the first time, we aim to study *when to pre-train* GNNs, *i.e.*, in what situations the graph pre-training should be adopted.

4 METHODOLOGY

In this section, we first present our proposed framework W2PGNN to answer when to pre-train GNNs in Section 4.1. Based on the framework, we further introduce the measure of the feasibility of performing pre-training in Section 4.2. Then in Section 4.3, we discuss our approximation to the feasibility of pre-training. Finally, the complexity analysis of W2PGNN is provided in Section 4.4.

4.1 Framework Overview

W2PGNN framework provides a guide to answer *when to pre-train GNNs from a data generation perspective*. The key insight is that if

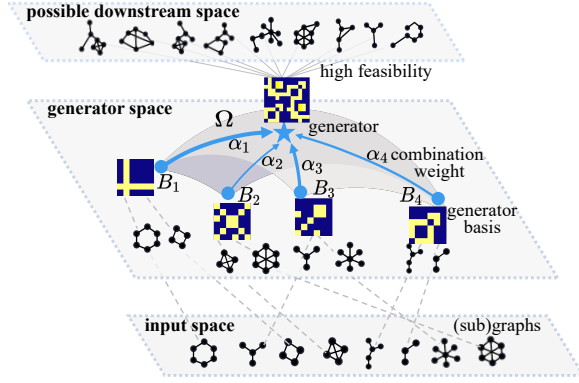


Figure 2: Illustration of our proposed framework W2PGNN to answer when to pre-train GNNs.

downstream data can be generated with high probability by a graph generator that summarizes the pre-training data, the downstream data would present high feasibility of performing pre-training.

The overall framework of W2PGNN can be found in Figure 2. Given the *input space* consisting of pre-training graphs, we fit them into a graph generator in the *generator space*, from which the graphs generated constitute the *possible downstream space*. More specifically, an ideal graph generator should inherit different kinds of topological patterns, based on which new graphs can be induced. Therefore, we first construct a graphon basis $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$, where each element B_i represents a graphon fitted from a set of (sub)graphs with similar patterns (*i.e.*, the blue dots ●). To access different combinations of generator basis, each B_i is assigned with a corresponding weight α_i (*i.e.*, the width of blue arrow →) and their combination gives rise to a graph generator (*i.e.*, the blue star ★). All weighted combinations compose the generator space Ω (*i.e.*, the gray surface ☁), from which graphs generated form the possible solution space of downstream data (shorted as possible downstream space). The generated graphs are those that could benefit from the pre-training data, we say that they exhibit *high feasibility* of performing pre-training.

In the following, we introduce the workflow of W2PGNN in the input space, the generator space and the possible downstream space in detail. Then, the application cases of W2PGNN are given for different practical use.

Input space. The input space of W2PGNN is composed of nodes’ ego-networks or graphs. For node-level pre-training, we take the nodes’ ego-networks to constitute the input space; For graph-level pre-training, we take the graphs (*e.g.*, small molecular graphs) as input space.

Generator space. As illustrated in Figure 2, each point (*i.e.*, graph generator) in the generator space Ω is a convex combination of generator basis $\mathcal{B} = \{B_1, B_2, \dots, B_k\}$. Formally, we define the graph generator as

$$f(\{\alpha_i\}, \{B_i\}) = \sum_{i=1}^k \alpha_i B_i, \quad \text{where } \sum_{i=1}^k \alpha_i = 1, \alpha_i \geq 0. \quad (2)$$

Different choices of $\{\alpha_i\}, \{B_i\}$ comprise different graph generators. All possible generators constitute the *generator space* $\Omega = \{f(\{\alpha_i\}, \{B_i\}) \mid \forall \{\alpha_i\}, \{B_i\}\}$.

We shall also note that, the graph generator $f(\{\alpha_i\}, \{B_i\})$ is indeed a mixed graphon, (*i.e.*, mixture of k graphons $\{B_1, B_2, \dots, B_k\}$), where each element B_i represents a graphon estimated from a set of similar pre-training (sub)graphs. Furthermore, it can be theoretically justified that the mixed version still preserve the properties of graphons (*c.f.* Theorem 5.1) and the key transferable patterns inherited in B_i (*c.f.* Theorem 5.2). Thus the graph generator $f(\{\alpha_i\}, \{B_i\})$, *i.e.*, mixed graphon, serves as a representative and comprehensive summary of pre-training data, from which unseen graphs with different combinations of transferable patterns can be induced.

Possible downstream space. All the graphs produced by the generators in the generator space Ω could benefit from the pre-training, and finally form the possible downstream space.

Formally, for each generator in the generator space Ω (we denote it as f for simplicity), we can generate a n -node graph as follows. First, we independently sample a random latent variable for each node. Then for each pair of nodes, we assign an edge between them with the probability equal to the value of the graphon at their randomly sampled points. This process can be formulated as:

$$\begin{aligned} v_1, \dots, v_n &\sim \text{Uniform}([0, 1]), \\ A_{ij} &\sim \text{Bernouli}(f(v_i, v_j)), \quad \forall i, j \in \{1, 2, \dots, n\}, \end{aligned} \quad (3)$$

where $f(v_i, v_j) \in [0, 1]$ indicates the corresponding value of the graphon at point (v_i, v_j) ¹, and $A_{ij} \in \{0, 1\}$ indicates the existence of edge between i -th node and j -th node. The adjacency matrix of the sampled graph G is denoted as $A = [A_{ij}] \in \{0, 1\}^{n \times n}, \forall i, j \in [n]$. We summarize this generation process as $G \leftarrow f$.

Therefore, with all generators from the generator space Ω , the possible downstream space is defined as $\mathcal{D} = \{G \leftarrow f \mid f \in \Omega\}$. Note that for each $\{\alpha_i\}, \{B_i\}$, we have a generator f ; and for each generator, we also have different generated graphs. Besides, we theoretically justify that the generated graphs in the possible downstream space can inherit key transferable graph patterns in our generator (*c.f.* Theorem 5.3).

Application cases. The proposed framework is flexible to be adopted in different application scenarios when discussing the problem of when to pre-train GNNs.

- *Use case 1: provide a user guide of a graph pre-trained model.* The possible downstream space \mathcal{D} serves as a user guide of a graph pre-trained model, telling the application scope of graph pre-trained models (*i.e.*, the possible downstream graphs that can benefit from the pre-training data).
- *Use case 2: estimate the feasibility of performing pre-training from pre-training data to downstream data.* Given a collection of pre-training graphs and a downstream graph, one can directly measure the feasibility of performing pre-training on pre-training data, before conducting costly pre-training and fine-tuning attempts. By making such pre-judgement of a kind of transferability,

¹For simplicity, we slightly abuse the notations $f(\cdot, \cdot)$. Note that $f(\{\alpha_i\}, \{B_i\})$ is a function of $\{\alpha_i\}$ and $\{B_i\}$, representing that the generator depends on $\{\alpha_i\}, \{B_i\}$; while for each generator (*i.e.*, mixed graphon) f given $\{\alpha_i\}, \{B_i\}$, it can be represented as a continuous, bounded and symmetric function $f: [0, 1]^2 \rightarrow [0, 1]$.

some unnecessary and expensive parameter optimization steps during model training and evaluation can be avoided.

- *Use case 3: select pre-training data to benefit the downstream.* In some practical scenarios where the downstream data is provided (e.g., a company needs to boost downstream performance of its business data), the feasibility of pre-training inferred by W2PGNN can be used to select data for pre-training to maximize the downstream performance with limited resources.

Use case 1 can be directly given by our produced possible downstream space \mathcal{D} . However, how to measure the feasibility of pre-training in use case 2 and 3 still remains a key challenge. In the following sections, we introduce the formal definition of the feasibility of pre-training and its approximate solution.

4.2 Feasibility of Pre-training

If a downstream graph can be generated with a higher probability from any generator in the generator space Ω , then the graph could benefit more from the pre-training data. We therefore define the feasibility of performing pre-training as the highest probability of the downstream data generated from a generator in Ω , which can be formulated as an optimization problem as follows.

DEFINITION 2 (FEASIBILITY OF GRAPH PRE-TRAINING). *Given the pre-training data \mathcal{G}_{train} and downstream data \mathcal{G}_{down} , we have the feasibility of performing pre-training on \mathcal{G}_{train} to benefit \mathcal{G}_{down} as*

$$\zeta(\mathcal{G}_{train} \rightarrow \mathcal{G}_{down}) = \sup_{\{\alpha_i\}, \{B_i\}} \Pr(\mathcal{G}_{down} | f(\{\alpha_i\}, \{B_i\})), \quad (4)$$

where $\Pr(\mathcal{G}_{down} | f(\{\alpha_i\}, \{B_i\}))$ denotes the probability of the graph sequence sampled from \mathcal{G}_{down} being generated by graph generator $f(\{\alpha_i\}, \{B_i\})$; each (sub)graph represents an ego-network (for node-level task) or a graph (for graph-level task) sampled from the downstream data \mathcal{G}_{down} .

However, the probability $\Pr(\mathcal{G}_{down} | f(\{\alpha_i\}, \{B_i\}))$ of generating the downstream graph from a generator is extremely hard to compute, we therefore turn to converting the optimization problem (4) to a tractable problem. Intuitively, if generator $f(\{\alpha_i\}, \{B_i\})$ can generate the downstream data with higher probability, it potentially means that the underlying generative patterns of pre-training data (characterized by $f(\{\alpha_i\}, \{B_i\})$) and downstream data (characterized by the graphon B_{down} fitted from \mathcal{G}_{down}) are more similar. Accordingly, we turn to figure out the infimum of the distance between $f(\{\alpha_i\}, \{B_i\})$ and B_{down} as the feasibility, i.e.,

$$\zeta(\mathcal{G}_{train} \rightarrow \mathcal{G}_{down}) = - \inf_{\{\alpha_i\}, \{B_i\}} \text{dist}(f(\{\alpha_i\}, \{B_i\}), B_{down}). \quad (5)$$

Following [43], we hire the 2-order Gromov-Wasserstein (GW) distance as our distance function $\text{dist}(\cdot, \cdot)$, as GW distance is commonly used to measure the difference between structured data.

Additionally, we establish a theoretical connection between the above-mentioned distance and the probability of generating the downstream data in extreme case, which further adds to the integrity and rationality of our solution. Detailed proof of the following theorem can be found in Appendix A.

THEOREM 4.1. *Given the graph sequence sampled from downstream data \mathcal{G}_{down} , we estimate its corresponding graphon as B_{down} . If a generator f can generate the downstream graph sequence with probability 1, then $\text{dist}(f, B_{down}) = 0$.*

4.3 Choose Graphon Basis to Approximate Feasibility

Although the feasibility has been converted to the optimization problem (5), exhausting all possible $\{\alpha_i\}, \{B_i\}$ to find the infimum is impractical. An intuitive idea is that we can choose some appropriate graphon basis $\{B_i\}$, which can not only prune the search space but also accelerate the optimization process. Therefore, we aim to first reduce the search space of graphon basis $\{B_i\}$ and then learn the optimal $\{\alpha_i\}$ in the reduced search space.

Considering that the downstream data may be formed via different generation mechanisms (implying various transferable patterns), a single graphon basis might have limited expressivity and completeness to cover all patterns. We therefore argue that a good reduced search space of graphon basis should cover a set of graphon bases. Here, we introduce three candidates of them as follows.

Integrated graphon basis. The first candidate of graphon basis is the integrated graphon basis $\{B_i\}_{integr}$. This graphon basis is introduced based on the assumption that the pre-training and the downstream graphs share very similar patterns. For example, the pre-training and the downstream graphs might come from social networks of different time spans [15]. In the situation, almost all patterns involved in the pre-training data might be useful for the downstream. To achieve this, we directly utilize all (sub)graphs sampled from the pre-training data to estimate one graphon as the graphon basis. This integrated graphon basis serves as a special case of the graphon basis introduced below.

Domain graphon basis. The second candidate is the domain graphon basis $\{B_i\}_{domain}$. The domain information that pre-training data comes from is important prior knowledge to indicate the transferability from the pre-training to downstream data. For example, when the downstream data is molecular network, it is more likely to benefit from the pre-training data from specific domains like biochemistry. This is because the specificity of molecules makes it difficult to learn transferable patterns from other domains, e.g., closed triangle structure represents diametrically opposite meanings (stable vs unstable) in social network and molecular network. Therefore, we propose to split the (sub)graphs sampled from pre-training data according to their domains, and each split of (sub)graphs will be used to estimate a graphon as a basis element. In this way, each basis element reflects transferable patterns from a specific domain, and all basis elements construct the domain graphon basis $\{B_i\}_{domain}$.

Topological graphon basis. The third candidate is the topological graphon basis $\{B_i\}_{topo}$. The topological similarity between the pre-training and the downstream data serves as a crucial indicator of transferability. For example, a downstream social network might benefit from the similar topological patterns in academic or web networks (e.g., closed triangle structure indicates stable relationship in all these networks). Then, the problem of finding topological graphon basis can be converted to partition n (sub)graphs sampled from pre-training data into k -split according to their topology similarity, where each split contains (sub)graphs with similar topology. Each element of graphon basis (i.e., graphon) fitted from each split of (sub)graphs is expected to characterize a specific kind of topological transferable pattern.

However, the challenge is that for graph structured data that is irregular and complex, we cannot directly measure the topological similarity between graphs. To tackle this problem, we introduce a *graph feature extractor* that maps arbitrary graph into a fixed-length vector representation. To approach a comprehensive and representative set of topological features, we here consider both node-level and graph-level properties.

For node-level topological features, we first apply a set of node-level property functions $[\phi_1(v), \dots, \phi_{m_1}(v)]$ for each node v in graph G to capture the local topological features around it. Considering that the numbers of nodes of two graphs are possibly different, we introduce an aggregation function AGG to summarize the node-level property of all nodes over G to a real number $\text{AGG}(\{\phi_i(v), v \in G\})$. We can thus obtain the node-level topological vector representation as follows.

$$h_{\text{node}}(G) = [\text{AGG}(\{\phi_1(v), v \in G\}), \dots, \text{AGG}(\{\phi_{m_1}(v), v \in G\})].$$

In practice, we calculate degree [2], clustering coefficient [17] and closeness centrality [6] for each node and instantiate the aggregation function AGG as the mean aggregator.

For graph-level topological features, we also employ a set of graph-level property functions for each graph G to serve as the vector representation

$$h_{\text{graph}}(G) = [\psi_1(G), \dots, \psi_{m_2}(G)],$$

where density [38], assortativity [28], transitivity [38] are adopted as graph-level properties here.²

Finally, the final representation of G produced by the graph feature extractor is

$$h = [h_{\text{local}}(G) || h_{\text{global}}(G)] \in \mathbb{R}^{m_1+m_2},$$

where $||$ is the concatenation function that combines both node-level and graph-level features. Given the topological vector representation, we leverage an efficient clustering algorithm K-Means [25] to obtain k-splits of (sub)graphs and finally fit each split into a graphon as one element of topological graphon basis.

Optimization solution. Given the above-mentioned three graphon bases, the choice of graphon basis $\{B_i\}$ can be specified to one of them. In this way, the pre-training feasibility (simplified as ζ) could be approximated in the reduced search space of graphon basis as

$$\zeta \leftarrow -\text{MIN}_{\{\alpha_i\}}(\inf_{\{\alpha_i\}} \text{dist}(f(\{\alpha_i\}, \{B_i\}), B_{\text{down}}), \forall \{B_i\} \in \mathcal{B}), \quad (6)$$

where $\mathcal{B} = \{\{B_i\}_{\text{topo}}, \{B_i\}_{\text{domain}}, \{B_i\}_{\text{integr}}\}$ is the reduced search space of $\{B_i\}$. Thus, the problem can be naturally splitted into three sub-problems with objective of $\text{dist}(f(\{\alpha_i\}, \{B_i\}_{\text{topo}}, B_{\text{down}}))$, $\text{dist}(f(\{\alpha_i\}, \{B_i\}_{\text{domain}}, B_{\text{down}}))$ and $\text{dist}(f(\{\alpha_i\}, \{B_i\}_{\text{integr}}, B_{\text{down}}))$ respectively. Each sub-problem can be solved by updating the corresponding learnable parameters $\{\alpha_i\}$ with multiple gradient descent steps. Taking one step as an example, we have

$$\{\alpha_i\} = \{\alpha_i\} - \eta \nabla_{\{\alpha_i\}} \text{dist}(f(\{\alpha_i\}, \{B_i\}), B_{\text{down}}) \quad (7)$$

where η is the learning rate. Finally, we achieve three infimum distances under different $\{B_i\} \in \mathcal{B}$ respectively, the minimum value among them is the approximation of pre-training feasibility. In practice, we adopt an efficient and differential approximation of

²Other graph-level properties can also be utilized like *diameter* and *Wiener index*, but we do not include them due to their high computational complexity.

GW distance, *i.e.*, entropic regularization GW distance [30], as the distance function. For graphon estimation, we use the “largest gap” method as to estimate graphon B_i .

4.4 Computation Complexity

We now show that the time complexity of W2PNN is much lower than traditional solution. Suppose that we have n_1 and n_2 (sub)graphs sampled from pre-training data and downstream data respectively, and denote $|V|$ and $|E|$ as the average number of nodes and edges per (sub)graph. The overall time complexity of W2PGNN is $O((n_1 + n_2)|V|^2)$. For comparison, traditional solution in Figure 1(a) to estimate the pre-training feasibility should make $l_1 \times l_2$ “pre-train and fine-tune” attempts, if there exist l_1 pre-training models and l_2 fine-tuning strategies. Suppose the batch size of pre-training as b and the representation dimension as d . The overall time complexity of traditional solution is $O(l_1 l_2 ((n_1 + n_2)(|V|^3 + |E|d) + n_1 b d))$. Detailed analysis can be found in Appendix C.

5 THEORETICAL ANALYSIS

In this section, we theoretically analyze the rationality of the generator space and possible downstream space in W2PGNN. Detailed proofs of the following theorems can be found in Appendix A.

5.1 Theoretical Justification of Generator Space

Our generator preserves the properties of graphons. We first theoretically prove that any generator in the generator space still preserve the properties of graphon (*i.e.*, a bounded symmetric function $[0, 1]^2 \rightarrow [0, 1]$, summarized in the following theorem.

THEOREM 5.1. *For a set of graphon basis $\{B_i\}$, the corresponding generator space $\Omega = \{f(\{\alpha_i\}, \{B_i\}) \mid \forall \{\alpha_i\}, \{B_i\}\}$ is the convex hull of $\{B_i\}$.*

Our generator preserves the key transferable patterns in graphon basis. As a preliminary, we first introduce the concept of *graph motifs* as a useful description of transferable graph patterns and leverage *homomorphism density* as a measure to quantify the degree to which the patterns inherited in a graphon.

DEFINITION 3 (GRAPH MOTIFS [26]). *Given a graph $G = (V, E)$ (V and E are node and edge set), graph motifs are substructures $F = (V', E')$ that recur significantly in statistics, where $V' \subset V, E' \subset E$ and $|V'| \ll |V|$.*

Graph motifs can be roughly taken as the key transferable graph patterns across graphs [51]. For example, the motif (A) has the same meaning of “feedforward loop” across networks of control system, gene systems or organisms.

Then, we introduce the measure of homomorphism density $t(F, B)$ to quantify the relative frequency of the key transferable pattern, *i.e.*, graph motifs F , inherited in graphon B .

DEFINITION 4 (HOMOMORPHISM DENSITY [22]). *Consider a graph motif $F = (V', E')$, we define a homomorphisms of F into graph $G = (V, E)$ as an adjacency-preserving map from V' to V , where $(i, j) \in E'$ implies $(i, j) \in E$. There could be multiple maps from V' to V , but only some of them are homomorphisms. Therefore, the definition of homomorphism density $t(F, G)$ is introduced to quantify the relative frequency with which the graph motif F appears in G .*

Analogously, the homomorphism density of graphs can be extended into the graphon B . We denote $t(F, B)$ as the homomorphism density of graph motif F into graphon B , which represents the relative frequency of F occurring in a collection of graphs $\{G_i\}$ that convergent to graphon B , i.e., $t(F, B) = \lim_{i \rightarrow \infty} t(F, \{G_i\})$.

Now, we are ready to quantify how much the transferable patterns in graphon basis can be preserved in our generator by exploring the difference between the homomorphism density of graph motifs into the graphon basis and that into our generator.

THEOREM 5.2. Assume a graphon basis $\{B_1, \dots, B_k\}$ and their convex combination $f(\{\alpha_i\}, \{B_i\}) = \sum_{i=1}^k \alpha_i B_i$. The a -th element of graphon basis B_a corresponds to a motif set. For each motif F_a in the motif set, the difference between the homomorphism density of F_a in $f(\{\alpha_i\}, \{B_i\})$ and that in basis element B_a is upper bounded by

$$|t(F_a, f(\{\alpha_i\}, \{B_i\})) - t(F_a, B_a)| \leq \sum_{b=1, b \neq a}^k |F_a| \alpha_b \|B_b - B_a\|_{\square} \quad (8)$$

where $|F_a|$ is the number of nodes in motif F_a , $\|\cdot\|_{\square}$ is the cut norm.

Theorem 5.2 indicates the graph motifs (i.e., key transferable patterns) inherited in each basis element can be preserved in our generator, which justifies the rationality to take the generator as a representative and comprehensive summary of pre-training data.

5.2 Theoretical Justification of Possible Downstream Space

The possible downstream space includes the graphs generated from generator $f(\{\alpha_i\}, \{B_i\})$. We here provide a theoretical justification that the generated graphs in possible downstream space can inherit key transferable graph patterns (i.e., graph motifs) in the generator.

THEOREM 5.3. Given a graph generator $f(\{\alpha_i\}, \{B_i\})$, we can obtain sufficient number of random graphs $\mathbb{G} = \mathbb{G}(n, f(\{\alpha_i\}, \{B_i\}))$ with n nodes generated from $f(\{\alpha_i\}, \{B_i\})$. The homomorphism density of graph motif F in \mathbb{G} can be considered approximately equal to that in $f(\{\alpha_i\}, \{B_i\})$ with high probability and can be represented as

$$P(|t(F, \mathbb{G}) - t(F, f(\{\alpha_i\}, \{B_i\}))| > \epsilon) \leq 2 \exp\left(-\frac{\epsilon^2 n}{8v(F)^2}\right), \quad (9)$$

where $v(F)$ denotes the number of nodes in F , and $0 \leq \epsilon \leq 1$.

Theorem 5.3 indicates that the homomorphism density of graph motifs into the generated graphs in the possible downstream space can be inherited from our generator to a significant degree.

6 EXPERIMENTS

In this section, we evaluate the effectiveness of W2PGNN with the goal of answering the following questions: (1) Given the pre-training and downstream data, is the feasibility of pre-training estimated by W2PGNN positively correlated with the downstream performance (Use case 2)? (2) When the downstream data is provided, does the pre-training data selected by W2PGNN actually help improve the downstream performance (Use case 3)?

Note that it is impractical to empirically evaluate the application scope of graph pre-trained models (Use case 1), as we cannot enumerate all graphs in the possible downstream space. By answering question (1), it can be indirectly verified that a part of graphs in the

possible downstream space, i.e., the downstream graphs with high feasibility, indeed benefit from the pre-training.

6.1 Experimental Setup

We validate our proposed framework on both node classification and graph classification task.

Datasets. For node classification task, we directly adopt six datasets from [31] as the candidates of pre-training data, which consists of Academia, DBLP(SNAP), DBLP(NetRep), IMDB, Facebook and LiveJournal (from academic, movie and social domains). Regarding the downstream datasets, we adopt US-Airport and H-Index from [31] and additionally add two more datasets Chameleon and Europe-Airport for a more comprehensive results. For graph classification task, we choose the large-scale datasets ZINC15 [32] containing 2 million unlabeled molecules. To enrich the follow-up experimental analysis, we use scaffold split to partition the ZINC15 into five datasets (ZINC15-0, ZINC15-1, ZINC15-2, ZINC15-3 and ZINC15-4) according to their scaffolds [14], such that the scaffolds are different in each dataset. Regarding the downstream datasets, we use 5 classification benchmark datasets BACE, BBBP, MUV, HIV and ClinTox contained in MoleculeNet [40].

Baseline of graph pre-training measures. The baselines can be divided into 3 categories: (1) EGI [53] computes the difference between the graph Laplacian of (sub)graphs from pre-training data and downstream data; (2) Graph Statistics, by which we merge average degree, degree variance, density, degree assortativity coefficient, transitivity and average clustering coefficient to construct a topological vector for each (sub)graph. (3) Clustering Coefficient, Spectrum of Graph Laplacian, and Betweenness Centrality, by which we adopt the distributions of graph properties as topological vectors. For (2) and (3), we calculate the negative value of Maximum Mean Discrepancy distance between the obtained topological vectors of the (sub)graph from pre-training data and that from downstream data. For efficiency, when conducting node classification, we randomly sample 10% nodes for each candidate pre-training dataset and all nodes for each downstream dataset, then extract their 2-hop ego-networks. The final measure is the average of distances/differences between each pair of pre-training and fine-tuning graphs.

Implementation Details. For node classification tasks, we randomly sample 1000 nodes for each pre-training dataset and extract 2-hops ego-networks of sampled nodes to compose our input space, and extract 2-hops ego-networks of all nodes in each downstream dataset to estimate the graphon. For graph classification tasks, we take all graphs in each pre-training dataset to compose our input space and use all graphs in each downstream dataset to estimate graphon. When constructing topological graphon basis, we set the number of clusters $k = 5$. The maximum iterations number of K-Means is set as 300. When constructing domain graphon basis, we take each pre-training dataset as a domain. For graphon estimation, we use the largest gap [3] approach and let the block size of graphon as the average number of nodes in all graphs. When learning α_i , we adopt Adam as the optimizer and set the learning rate η as 0.05. For the GW distance, we adopt its differential and efficient version entropic regularization GW distance with default hyperparameters [30]. We provide an open-source implementation of our model W2PGNN at <https://github.com/caoyxuan/W2PGNN>.

	$N = 2$					$N = 3$				
	US-Airport	Europe-Airport	H-index	Chameleon	Rank	US-Airport	Europe-Airport	H-index	Chameleon	Rank
Graph Statistics	-0.6068	0.3571	-0.6220	-0.2930	10	-0.7096	-0.5052	-0.2930	-0.8173	10
EGI	0.0672	-0.6077	-0.2152	-0.2680	9	-0.2358	-0.5540	-0.2822	-0.6511	9
Clustering Coefficient	-0.0273	0.1519	0.3622	0.3130	5	-0.0039	0.2069	0.4829	0.2279	4
Spectrum of Graph Laplacian	-0.2023	0.1467	0.0794	0.0095	8	-0.7648	-0.4311	0.2611	-0.2300	8
Betweenness Centrality	-0.2739	-0.2554	0.2051	0.2241	7	-0.3421	-0.5903	0.1364	0.0849	7
W2PGNN (intergr)	0.3579	0.1224	0.3313	0.1072	6	0.0841	0.5310	0.4213	-0.0916	6
W2PGNN (domain)	0.4774	0.4666	0.6775	0.3460	3	0.7132	0.5523	0.7381	0.1857	3
W2PGNN (topo)	0.2059	0.3908	0.3745	0.4464	4	0.4900	0.5061	0.4072	0.1497	5
W2PGNN ($\alpha = 1$)	0.4172	0.5206	0.6829	0.4391	2	0.5282	0.6663	0.7240	0.3246	1
W2PGNN	0.3941	0.5336	0.7162	0.4838	1	0.5089	0.6706	0.6754	0.3166	2

Table 1: Pearson correlation coefficient between the estimated pre-training feasibility and the best downstream performance on node classification. N denotes the number of candidate pre-training datasets that form the pre-training data. Bold indicates the highest coefficient. “Rank” represents the overall ranking on all downstream datasets.

	$N = 2$						$N = 3$					
	BACE	BBBP	MUV	HIV	ClinTox	Rank	BACE	BBBP	MUV	HIV	ClinTox	Rank
Graph Statistics	-0.4118	-0.1328	0.3858	0.0174	-0.3577	9	-0.3093	-0.1430	0.1946	0.3545	-0.1372	7
EGI	0.2912	-0.6862	0.4488	0.0587	0.0452	7	0.4570	0.3230	0.3024	0.4144	-0.0085	3
Clustering Coefficient	-0.5098	-0.5097	0.3754	0.4738	0.5154	8	-0.4080	0.3217	-0.1190	-0.2483	-0.4248	9
Spectrum of Graph Laplacian	-0.0633	-0.4878	-0.3413	-0.1125	-0.2562	10	-0.3563	-0.1611	-0.2294	-0.2448	0.3001	8
Betweenness Centrality	-0.0021	0.7755	0.4040	0.0339	0.3411	6	-0.3695	-0.4568	-0.2752	-0.3035	-0.2129	10
W2PGNN (intergr)	0.7547	0.7790	0.2907	0.7033	0.5639	3	0.4081	0.4687	-0.0567	0.3802	0.4354	5
W2PGNN (domain)	0.7334	0.7689	0.5395	0.6831	0.5431	5	0.0864	0.3680	0.0187	0.4784	0.3765	6
W2PGNN (topo)	0.6656	0.7164	0.8131	0.7391	0.5406	2	0.1109	0.5357	0.0514	0.3265	0.4724	4
W2PGNN ($\alpha = 1$)	0.6549	0.7690	0.6730	0.7033	0.5639	4	0.5287	0.7102	0.1925	0.5893	0.5430	2
W2PGNN	0.7549	0.7776	0.8131	0.7044	0.5784	1	0.6207	0.6696	0.5227	0.6529	0.5994	1

Table 2: Pearson correlation coefficient between the feasibility and the best downstream performance on graph classification.

6.2 Results of Pre-training Feasibility

Setup. As a pre-judgement to assess the necessity of pre-training before conducting any pre-training/fine-tuning attempts, the graph pre-training feasibility should reveal the optimal case that downstream data can benefit from pre-training data. However, it is impractical to obtain the optimal case, because we cannot enumerate all factors affecting model performance, *e.g.*, pre-training strategies, fine-tuning strategies, backbone models. Hence we use the best downstream performance achieved among existing commonly-used pre-training models as an approximation.

For node classification tasks, we use the following 4 graph pre-training models: GraphCL [46] and GCC models [31] with three different hyper-parameter (*i.e.*, 128, 256 and 512 rw-hops). For graph classification tasks, we adopt 7 SOTA pre-training models: AttrMasking [14], ContextPred [14], EdgePred [14], Infomax [14], GraphCL [46], GraphMAE [13] and JOAO [45]. When pre-training, we directly use the default hyper-parameters of pre-training models except the rw-hops in GCC. During fine-tuning, we freeze the parameters of pre-trained models and utilize the logistic regression as classifier for node classification and SVM as classifier for graph classification, following [31] and its fine-tuning hyper-parameters. The downstream results are reported as the average of Micro F1 and ROC-AUC under 10 runs on node classification and graph classification respectively. For each downstream task, we take the best performance among all methods.

For a comprehensive evaluation on the correlation between the estimated pre-training feasibility and the best downstream performance, we need to construct multiple $\langle \mathcal{G}_{\text{train}}, \mathcal{G}_{\text{down}} \rangle$ sample pairs as our evaluation samples. When constructing the $\langle \mathcal{G}_{\text{train}}, \mathcal{G}_{\text{down}} \rangle$

sample pairs for each downstream data, multiple pre-training data are required to be paired with it. Hence we adopt the following two settings to augment the choice of pre-training data for more possibilities. We use N as the number of dataset candidates contained in pre-training data. For $N = 2$ and $N = 3$, we randomly select 2 and 3 pre-training dataset candidates, respectively as pre-training data. We enumerate all possible combination cases for graph classification tasks and randomly select 40% of all cases for node classification tasks for efficiency.

Results. Table 1 (for node classification) and Table 2 (for graph classification) show the Pearson correlation coefficient between the best downstream performance and the estimated pre-training feasibility by W2PGNN and baselines for each downstream dataset. A higher coefficient indicates a better estimation of pre-training feasibility. We also include 4 variants of W2PGNN: W2PGNN (intergr), W2PGNN (domain) and W2PGNN (topo) only utilize the integrated graphon basis, domain graphon basis and topological graphon basis to approximate feasibility respectively, and W2PGNN ($\alpha = 1$) set the learnable combination weights $\{\alpha_i\}$ as constant 1. We have the following observations. (1) The results show that our model achieve the highest overall ranking in most cases, indicating the superiority of our proposed framework. (2) We find that the measures provided by other baselines sometimes show no correlation or negative correlation with the best downstream performance. (3) Comparing W2PGNN and its variants, we find that although the variants sometimes achieve superior performance on some downstream datasets, they cannot consistently perform well on all datasets. In contrast, the top-ranked W2PGNN can provide a more comprehensive picture with various graph bases and learnable combination weights.

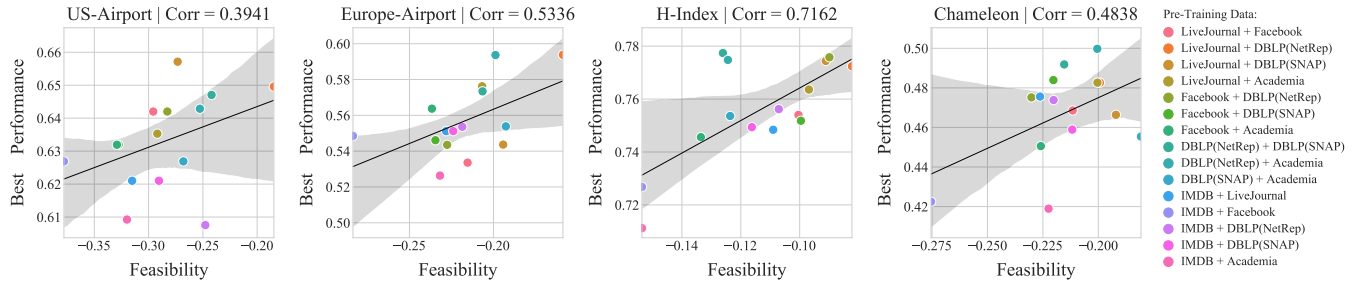


Figure 3: Pre-training feasibility vs. the best downstream performance on node classification when the selection budget is 2.

	N = 2					N = 3				
	US-Airport	Europe-Airport	H-index	Chameleon	Rank	US-Airport	Europe-Airport	H-index	Chameleon	Rank
All Datasets	65.62	55.65	75.22	46.81	-	65.62	55.65	75.22	46.81	-
Graph Statistics	64.20	53.36	74.30	44.31	4	62.27	54.58	72.88	43.87	5
EGI	64.96	57.37	74.30	43.21	2	62.27	57.36	72.88	45.93	3
Clustering Coefficient	62.61	52.87	77.74	43.21	3	62.94	54.58	75.18	44.66	4
Spectrum of Graph Laplacian	61.76	57.88	73.14	42.20	5	63.95	54.87	73.90	44.66	2
Betweenness Centrality	64.96	52.87	73.50	41.63	6	62.27	54.87	75.18	43.87	6
W2PGNN	64.96	57.88	77.24	45.54	1	63.95	57.59	75.68	46.07	1

Table 3: Node classification results when performing pre-training on different selected pre-training data. We also provide the results of using all pre-training data without selection for your reference (see “All Datasets” in the table).

To provide a deeper understanding of the feasibility estimated by W2PGNN, Figure 3 shows our estimated pre-training feasibility (in x-axis) versus the best downstream performance on node classification (in y-axis) of all <pre-training data, downstream data> pairs (one point represents the result of one pair) when the selection budget is 2. The plots when the selection budget is 3 and the plots under graph classification can be found in Appendix B.1. We find that there is a strong positive correlation between estimated pre-training feasibility and the best downstream performance on all downstream datasets, which also suggests the significance of our feasibility.

6.3 Results of Pre-Training Data Selection

Given the downstream data, a collection of pre-training dataset candidates and a selection budget (*i.e.*, the number of datasets selected for pre-training) due to limited resources, we aim to select the pre-training data with the highest feasibility, so as to benefit the downstream performance.

Setup. We here adopt two settings, *i.e.*, selection budget is set as 2 and 3 respectively. The datasets that are augmented for more pre-training data choices in Section 6.2 can be directly used as the candidates of pre-training datasets here. Then, the selected pre-training data serves as the input of graph pre-training model. For node classification tasks, we adopt GCC as the pre-training model as an example, because it is the pre-training model that can be generalized across domains and most of the datasets used for node classification are taken from it [31]. For graph classification tasks, we take GraphCL as the pre-training model as it provides multiple graph augmentation approaches and is more general [46].

Results. Table 3 shows the results of pre-training data selection on node classification task. (The results on graph classification is included in Appendix B.2). We have the following observations. (1) We can see that the pre-training data selected by W2PGNN ranks

first, which is the most suitable one for downstream. (2) We find that sometimes simple graph property like clustering coefficient serves as a good choice on a specific dataset (*i.e.*, H-index), when the budget of pre-training data is 2. It is because that H-index exhibits the largest clustering coefficient compared to other downstream datasets, which facilitates the data selection via clustering coefficient. However, such simple graph property is only applicable when the downstream dataset shows a strong indicator of the property, and is not helpful when you need to select more datasets for pre-training (see results under $N=3$). (3) Moreover, it is also interesting to see that using all pre-training data for pre-training is not always a reliable choice. We find that carefully selecting pre-training data can not only benefit downstream performance but also reduce computation resources.

7 CONCLUSION

This paper proposes a W2PGNN framework to answer the question of *when to pre-train* GNNs based on the generative mechanisms from pre-training to downstream data. W2PGNN designs a graphon-based graph generator to summarize the knowledge in pre-training data, and the generator can in turn produce the solution space of downstream data that can benefit from the pre-training. W2PGNN is theoretically and empirically shown to have great potential to provide the application scope of graph pre-training models, estimate the feasibility of pre-training and help select pre-training data.

ACKNOWLEDGMENTS

This work was partially supported by NSFC (62206056), Zhejiang NSF (LR22F020005), the National Key Research and Development Project of China (2018AAA0101900), and the Fundamental Research Funds for the Central Universities.

REFERENCES

- [1] Edo M Airolidi, Thiago B Costa, and Stanley H Chan. 2013. Stochastic blockmodel approximation of a graphon: Theory and consistent estimation. In *NeurIPS*. 692–700.
- [2] Katy Börner, Soma Sanyal, Alessandro Vespignani, et al. 2007. Network science. *Annu. rev. inf. sci. technol.* 41, 1 (2007), 537–607.
- [3] Antoine Channaron, Jean-Jacques Daudin, and Stéphane Robin. 2012. Classification and estimation in the Stochastic Blockmodel based on the empirical degrees. *Electronic Journal of Statistics* 6 (2012), 2574–2601.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*. 4171–4186.
- [5] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings via Diffusion Wavelets. In *SIGKDD*.
- [6] Linton C Freeman et al. 2002. Centrality in social networks: Conceptual clarification. *Social network: critical concepts in sociology. Londres: Routledge* 1 (2002), 238–263.
- [7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD*.
- [8] Hakim Hafidi, Mounir Ghogho, Philippe Ciblat, and Ananthram Swami. 2020. GraphCL: Contrastive Self-Supervised Learning of Graph Representations. *ArXiv abs/2007.08025* (2020).
- [9] Xueting Han, Zhenhuan Huang, Bang An, and Jing Bai. 2021. Adaptive Transfer Learning on Graph Neural Networks. In *SIGKDD*.
- [10] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. 2022. G-Mixup: Graph Data Augmentation for Graph Classification. In *ICML*.
- [11] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive multi-view representation learning on graphs. In *ICML*. PMLR, 4116–4126.
- [12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*. 9729–9738.
- [13] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. 2022. GraphMAE: Self-Supervised Masked Graph Autoencoders. In *SIGKDD*. 594–604.
- [14] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. 2020. Strategies for pre-training graph neural networks. In *ICLR*.
- [15] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In *SIGKDD*.
- [16] Ziniu Hu, Changjun Fan, Ting Chen, Kai-Wei Chang, and Yizhou Sun. 2019. Pre-Training Graph Neural Networks for Generic Structural Feature Extraction. *ArXiv abs/1905.13728* (2019).
- [17] Marcus Kaiser. 2008. Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks. *New Journal of Physics* 10 (2008), 083042.
- [18] Thomas Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *ArXiv abs/1611.07308* (2016).
- [19] Pengyong Li, Jun Wang, Ziliang Li, Yixuan Qiao, Xianggen Liu, Fei Ma, Peng Gao, Sen Song, and Guowang Xie. 2021. Pairwise Half-graph Discrimination: A Simple Graph-level Self-supervised Strategy for Pre-training Graph Neural Networks. In *IJCAI*.
- [20] Sihang Li, Xiang Wang, An Zhang, Yingxin Wu, Xiangnan He, and Tat-Seng Chua. 2022. Let Invariant Rationale Discovery Inspire Graph Contrastive Learning. In *ICML*. 13052–13065.
- [21] Can Liu, Yuncong Gao, Li Sun, Jinghua Feng, Hao Yang, and Xiang Ao. 2022. User Behavior Pre-training for Online Fraud Detection. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3357–3365.
- [22] László Lovász. 2012. *Large networks and graph limits*. Vol. 60. American Mathematical Soc.
- [23] László Lovász and Balázs Szegedy. 2006. Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B* 96, 6 (2006), 933–957.
- [24] Yuanfu Lu, Xunqiang Jiang, Yuan Fang, and Chuan Shi. 2021. Learning to Pre-train Graph Neural Networks. In *AAAI*.
- [25] J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, 281–297.
- [26] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.
- [27] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning Distributed Representations of Graphs. *ArXiv abs/1707.05005*.
- [28] Mark EJ Newman. 2003. Mixing patterns in networks. *Physical review E* 67, 2 (2003), 026126.
- [29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *SIGKDD*.
- [30] Gabriel Peyré, Marco Cuturi, and Justin Solomon. 2016. Gromov-wasserstein averaging of kernel and distance matrices. In *ICML*. PMLR, 2664–2672.
- [31] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *SIGKDD*.
- [32] T. Sterling and John J. Irwin. 2015. ZINC 15 – Ligand Discovery for Everyone. *Journal of Chemical Information and Modeling* 55 (2015), 2324 – 2337.
- [33] Fan-Yun Sun, Jordan Hoffmann, and Jian Tang. 2020. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. In *ICLR*.
- [34] Ke Sun, Zhanxing Zhu, and Zhouchen Lin. 2020. Multi-Stage Self-Supervised Learning for Graph Convolutional Networks. In *AAAI*.
- [35] Mengying Sun, Jing Xing, Huijun Wang, Bin Chen, and Jiayu Zhou. 2021. MoCL: Data-driven Molecular Fingerprint via Knowledge-aware Contrastive Learning from Molecular Graph. In *SIGKDD*.
- [36] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*.
- [37] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
- [38] Stanley Wasserman and Katherine Faust. 1994. *Social network analysis: Methods and applications*. (1994).
- [39] Max Welling and Thomas N Kipf. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.
- [40] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. 2018. MoleculeNet: a benchmark for molecular machine learning. *Chemical science* 9, 2 (2018), 513–530.
- [41] Feng Xia, Jiaying Liu, Hansong Nie, Yonghao Fu, Liangtian Wan, and Xiangjie Kong. 2019. Random walks: A review of algorithms and applications. *IEEE Transactions on Emerging Topics in Computational Intelligence* 4, 2 (2019), 95–107.
- [42] Jun Xia, Jiangbin Zheng, Cheng Tan, Ge Wang, and Stan Z. Li. 2022. Towards Effective and Generalizable Fine-tuning for Pre-trained Molecular Graph Models. *bioRxiv* (2022).
- [43] Hongteng Xu, Peilin Zhao, Junzhou Huang, and Dixin Luo. 2021. Learning Graphon Autoencoders for Generative Graph Modeling. *ArXiv abs/2105.14244* (2021).
- [44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *ICLR*.
- [45] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. 2021. Graph contrastive learning automated. In *ICML*. PMLR, 12121–12132.
- [46] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. In *NeurIPS*, Vol. 33. 5812–5823.
- [47] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2020. When Does Self-Supervision Help Graph Convolutional Networks?. In *PMLR*, Vol. 119. 10871–10880.
- [48] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *IJCAI*.
- [49] Jiyang Zhang, Xi Xiao, Long-Kai Huang, Yu Rong, and Yatao Bian. 2022. Fine-Tuning Graph Neural Networks via Graph Topology induced Optimal Transport. In *IJCAI*.
- [50] Zaixin Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. 2021. Motif-based Graph Self-Supervised Learning for Molecular Property Prediction. In *NeurIPS*.
- [51] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Chee-Kong Lee. 2021. Motif-based graph self-supervised learning for molecular property prediction. In *NeurIPS*, Vol. 34. 15870–15882.
- [52] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver J. Woodford, Meng Jiang, and Neil Shah. 2021. Data Augmentation for Graph Neural Networks. In *AAAI*.
- [53] Qi Zhu, Yidan Xu, Haonan Wang, Chao Zhang, Jiawei Han, and Carl Yang. 2021. Transfer Learning of Graph Neural Networks with Ego-graph Information Maximization. In *NeurIPS*.
- [54] Yanqiao Zhu, Yichen Xu, Feng Yu, Q. Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. *ArXiv abs/2006.04131* (2020).

A PROOFS

A.1 Proof of Theorem 4.1

We first show the following lemma, which would be used in the proof of Theorem 4.1,

LEMMA 1. (Proposition 11.32 in [22]) For every graphon W , generating a W -random graph $\mathbb{G}(n, W)$ for $n = 1, 2, \dots$ we get a graph sequence such that $\mathbb{G}(n, W) \rightarrow W$ with probability 1.

Proof of Theorem 4.1: Since we assume $\mathcal{G}_{\text{down}}$ can be generated from $f(\{\alpha_i\}, \{B_i\})$ with probability 1, according to Lemma 1, we can have that $\mathcal{G}_{\text{down}} \rightarrow f(\{\alpha_i\}, \{B_i\})$ with probability 1. On the other hand, since B_{down} is the graphon fitted by $\mathcal{G}_{\text{down}}$, which means $\mathcal{G}_{\text{down}}$ is convergent to graphon B_{down} and we have $\mathcal{G}_{\text{down}} \rightarrow B_{\text{down}}$. Hence B_{down} is equivalent to f , then $\text{dist}(f, B_{\text{down}}) = 0$.

A.2 Proof of Theorem 5.1

Proof: Given an arbitrary set of graphons $X = \{B_i\}$ and $i = 1 \dots k$ the general form of the convex combination of graphons in X can be represented as:

$$f(\{\alpha_i\}, \{B_i\}) = \sum_{i=1}^k \alpha_i B_i, \quad \left(\sum_{i=1}^k \alpha_i = 1, \alpha_i \geq 0 \right). \quad (10)$$

Next we prove that our generator preserves the properties of graphons. the convex combination of graphons is still a bounded symmetric function $[0, 1]^2 \rightarrow [0, 1]$. First, we prove that the convex combination $f(\{\alpha_i\}, \{B_i\})$ is still symmetric. Since $f(\{\alpha_i\}, \{B_i\})$ is symmetric if and only if it satisfies that $f(\{\alpha_i\}, \{B_i\}) = f(\{\alpha_i\}, \{B_i\})^T$, we have the following derivations:

$$\begin{aligned} f(\{\alpha_i\}, \{B_i\})^T &= (\alpha_1 B_1 + \dots + \alpha_k B_k)^T \\ &= \alpha_1 B_1^T + \dots + \alpha_k B_k^T \\ &= \alpha_1 B_1 + \dots + \alpha_k B_k = f(\{\alpha_i\}, \{B_i\}). \end{aligned} \quad (11)$$

Then we prove that the convex combination $f(\{\alpha_i\}, \{B_i\})$ is still in $[0, 1]$ Let B_{max} indicates the maximum B_i , meanwhile B_{min} indicates the minimum B_i

$$f(\{\alpha_i\}, \{B_i\}) = (\alpha_1 B_1 + \dots + \alpha_k B_k) \leq \sum_{i=1}^k \alpha_i B_{\text{max}} = B_{\text{max}} \leq 1. \quad (12)$$

$$f(\{\alpha_i\}, \{B_i\}) = (\alpha_1 B_1 + \dots + \alpha_k B_k) \geq \sum_{i=1}^k \alpha_i B_{\text{min}} \geq B_{\text{min}} \geq 0. \quad (13)$$

Thus, for a set of graphon basis X , the generator space Ω is the set of all convex combinations of basis elements in X , Ω is the convex hull of X .

A.3 Proof of Theorem 5.2

We first show the following two lemmas, which would be used in the proof of Theorem 5.2.

The first lemma is known as counting lemma for graphons, provided in Lemma 10.23 in [22].

LEMMA 2. Let F be a graph motif and let B, B' be two graphon. Then we have

$$|t(F, B) - t(F, B')| \leq |F| \|B - B'\|_{\square}. \quad (14)$$

Each absolute value term in the sum is bounded by the cut norm $\|B - B'\|_{\square}$. When we fix all other irrelevant variables (everything except u_i and v_i for the i -th term), altogether implying that

$$|t(F, B) - t(F, B')| \leq |F| \|B - B'\|_{\square}. \quad (15)$$

LEMMA 3. The cut norm of a graphon $\|B\|_{\square}$ is defined as

$$\|B\|_{\square} = \sup_{S, T \subseteq [0, 1]} \left| \int_{S \times T} B \right|, \quad (16)$$

where the supremum is taken over all measurable subsets S and T . Obviously, suppose $\alpha \in \mathbb{R}$, we have

$$\|\alpha B\|_{\square} = \sup_{S, T \subseteq [0, 1]} \left| \int_{S \times T} \alpha B \right| = \sup_{S, T \subseteq [0, 1]} \left| \alpha \int_{S \times T} B \right| = \alpha \|B\|_{\square}. \quad (17)$$

Proof of Theorem 5.2: Based on the Lemma 2 and Lemma 3, we have the following derivations. The a -th element B_a in graphon basis has its corresponding motif set. Each motif F_a is expected to be preserved and to exhibit similar frequency (i.e., homomorphism density) in $f(\{\alpha_i\}, \{B_i\})$.

Applying Lemma 2, we have the following derivations:

$$\begin{aligned} |t(F_a, f(\{\alpha_i\}, \{B_i\})) - t(F_a, B_a)| &\leq |F_a| \|f(\{\alpha_i\}, \{B_i\}) - B_a\|_{\square} \\ |t(F_a, f(\{\alpha_i\}, \{B_i\})) - t(F_a, B_a)| &\leq |F_a| \left\| \sum_{b=1}^k \alpha_b B_b - \sum_{b=1}^k \alpha_b B_a \right\|_{\square} \\ |t(F_a, f(\{\alpha_i\}, \{B_i\})) - t(F_a, B_a)| &\leq |F_a| \left\| \sum_{b=1}^k \alpha_b (B_b - B_a) \right\|_{\square}. \end{aligned} \quad (18)$$

Combining with the triangle inequality, we have:

$$|t(F_a, f(\{\alpha_i\}, \{B_i\})) - t(F_a, B_a)| \leq \sum_{b=1, b \neq a}^k |F_a| \alpha_b \|B_b - B_a\|_{\square}. \quad (19)$$

where $a=1 \dots k$, $|F|$ represents the number of nodes in motif F , and $\|\cdot\|_{\square}$ is the cut norm.

A.4 Proof of Theorem 5.3

LEMMA 4. (Corollary 10.4 in [22]) Let W be a graphon, $n \geq 1$, $0 < \varepsilon < 1$, and let F be a simple graph, then the W -random graph $\mathbb{G} = \mathbb{G}(n, W)$ satisfies

$$P(|t(F, \mathbb{G}) - t(F, W)| > \varepsilon) \leq 2 \exp\left(-\frac{\varepsilon^2 n}{8v(F)^2}\right). \quad (20)$$

Proof of Theorem 5.3: Apply Lemma 4 in our setting, let graphon B as W Lemma 4, we have

$$P(|t(F, \mathbb{G}) - t(F, B)| > \varepsilon) \leq 2 \exp\left(-\frac{\varepsilon^2 n}{8v(F)^2}\right). \quad (21)$$

B ADDITIONAL RESULTS

B.1 Results of Pre-training Feasibility

Figure 4 shows the results on node classification when the selection budget is 3. Figure 5 shows the results on graph classification when the selection budget is 2 and 3 respectively. We find that in all cases, there is a strong positive correlation between pre-training feasibility and the best downstream performance.

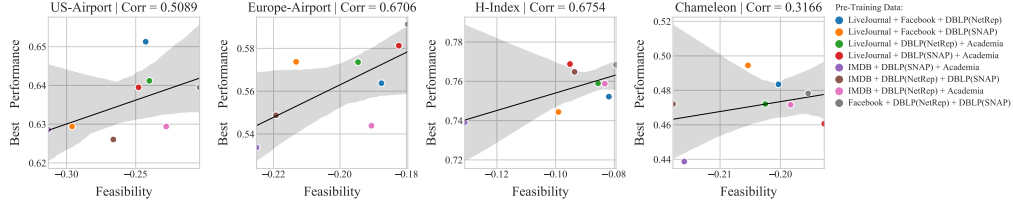


Figure 4: Pre-training feasibility vs. the best downstream performance on node classification when the selection budget is 3.

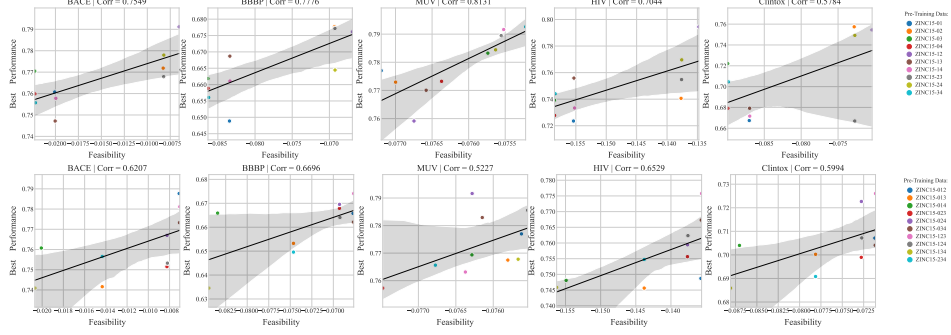


Figure 5: Pre-training feasibility vs. best downstream performance on graph classification when the selection budget is 2 and 3.

	N = 2					Rank	N = 3					Rank
	BACE	BBBP	MUV	HIV	ClinTox		BACE	BBBP	MUV	HIV	ClinTox	
All Datasets	74.86	62.67	73.80	68.31	60.58	-	74.86	62.67	73.80	68.31	60.58	-
Graph Statistics	71.52	58.47	69.42	70.31	62.99	2	65.95	62.36	68.98	68.83	59.86	5
EGI	68.46	65.65	69.42	68.29	65.21	3	68.48	61.00	68.98	68.83	60.35	3
Clustering Coefficient	71.52	60.06	69.42	70.31	58.91	5	65.95	61.00	68.98	68.83	60.35	6
Spectrum of Graph Laplacian	69.43	65.65	68.57	68.88	59.82	6	70.21	61.00	69.14	69.18	63.02	2
Betweenness Centrality	71.52	63.20	69.42	71.72	58.91	4	67.08	67.41	68.98	68.83	60.35	4
W2PGNN	73.33	65.46	74.17	70.69	64.21	1	73.54	65.02	72.49	71.18	62.26	1

Table 4: Graph classification results when performing pre-training on different selected pre-training data. We also provide the results of using all pre-training data without selection for your reference (see “All Datasets” in the table).

B.2 Results of Pre-Training Data Selection

Table 4 shows the results of pre-training data selection on graph classification tasks. The backbone pre-training model used here is GraphCL [46]. We can see that the pre-training data selected by W2PGNN ranks the first, which suggests that the effectiveness of our strategy on the graph classification task is still significant.

C COMPUTATION COMPLEXITY ANALYSIS

We show the time complexity of W2PGNN and the traditional solution. Suppose that we have n_1 and n_2 (sub)graphs sampled from pre-training data and downstream data respectively. Denote $|V|$ and $|E|$ as the average number of nodes and edges per (sub)graph.

The time complexity of W2PGNN consists of three components: computation of three graphon base ($\{B_i\}_{\text{topo}}$, $\{B_i\}_{\text{domain}}$, $\{B_i\}_{\text{integr}}$), graphon estimation of downstream data (i.e., B_{down}), computation of GW distance (i.e., $\text{dist}(\cdot, \cdot)$): (1) Among the estimation of three graphon base, the topological feature extraction is the most time-consuming one. It mainly involves the topological feature extractor, K-means clustering and graphon estimation of pre-training data, which costs $O(n_1|E|)$ (which is taken as the complexity of the most costly property closeness) [6], $O(n_1)$ [17] and $O(n_1|V|^2)$ [3], respectively. The domain and integrated graphon basis only include the graphon estimation of pre-training data and cost $O(n_1|V|^2)$ [3]. (2)

The graphon estimation of downstream data costs $O(n_2|V|^2)$ [3]. (3) The computation of GW distance costs $O(|V|^3)$ [30], which can be ignored because we have $|V| \ll n_1 + n_2$ (For node-level transferable patterns, extracting the ego-network of sampled nodes via breadth first search costs $O((n_1 + n_2)(|V| + |E|))$, which can be ignored). So the overall time complexity of W2PGNN is $O((n_1 + n_2)|V|^2)$.

For comparison, traditional solution make $l_1 \times l_2$ “pre-train and fine-tune” attempts, if there are l_1 pre-training models and l_2 fine-tuning strategies. Suppose the batch size of pre-training as b and the representation dimension as d . The time complexity of each pre-training model (taking the most general graph pre-training model GCC [31] as example) is typically from data augmentation, GNN encoder and contrastive loss, which costs $O(n_1|V|^3)$ (subgraphs sampled from random walk with restarts as augmentation) [41], $O(n_1|E|d)$ (GIN as graph encoder) and $O(n_1bd)$ [20], respectively. (Note that other data augmentations like dropping nodes cost $O(|V|^2)$, but they cannot achieve good performance on node classification in our pre-training experiments.) The time complexity of each fine-tuning strategy involves the inference of pre-trained model $O(n_2(|V|^3 + |E|d))$ and downstream predictor $O(n_2d)$ (which can be ignored), under the simple freezing mode. Thus the overall time complexity of traditional solution is $O(l_1l_2((n_1 + n_2)(|V|^3 + |E|d) + n_1bd))$.