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  
wanchunping02@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 a theoretically sound solution for the first application and extensive empirical justifications for the latter two applications.

KEYWORDS

graph neural networks, graph pre-training

ACM Reference Format:


1 INTRODUCTION

Graph neural networks (GNNs) have undergone rapid development and become increasingly popular for learning graph data [36, 38, 43]. 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, 23, 30]. 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, 30]. 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 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 [23, 30, 45] and fine-tuning strategies to enhance downstream performance [9, 16, 41, 48]. 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. 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 \( G_{\text{train}} \) and the downstream graph data \( G_{\text{down}} \), 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
When to Pre-Train Graph Neural Networks? From Data Generation Perspective! KDD ’23, August 6–10, 2023, Long Beach, CA, USA

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[21]. Literally, graphon has been studied from two perspectives: as limit of graph sequence, and as graph generator[1, 10, 22]. 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[21]. 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, 28, 33, 35, 47, 51]. 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, 23, 26, 32, 34, 45, 46, 49, 53], or explore cross-domain pre-training strategies [23, 30, 45] as well as fine-tuning strategies [9, 16, 41, 48]. 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 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 $\mathcal{B} = \{ B_1, B_2, \ldots, B_k \}$, where each element $B_i$ represents a graphon fitted from a set of (sub)graphs with similar patterns (i.e., the blue dots $\heartsuit$). To access different combinations of generator basis, each $B_i$ is assigned with a corresponding weight $\alpha_i$ (i.e., the width of blue arrow $\Rightarrow$) and their combination gives rise to a graph generator (i.e., the blue star $\star$). All weighted combinations compose the generator space $\Omega$ (i.e., the gray surface $\bigcirc$), 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 $\Omega$ is a convex combination of generator basis $\mathcal{B} = \{ B_1, B_2, \ldots, B_k \}$. Formally, we define the graph generator as

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

(2)

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

We shall also note that, the graph generator $\alpha(\{ \alpha_i \}, \{ B_i \})$ is indeed a mixed graphon, (i.e., mixture of $k$ graphons $\{ B_1, B_2, \ldots, B_k \}$), 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 $\alpha(\{ \alpha_i \}, \{ B_i \})$, 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 $\Omega$ could benefit from the pre-training, and finally form the possible downstream space.

Formally, for each generator in the generator space $\Omega$ (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:

$$
\nu_1, \cdots, \nu_n \sim \text{Uniform}(0, 1),
$$

$$
\alpha_{ij} \sim \text{Bernoulli}(f(\nu_i, \nu_j)),\quad \forall i, j \in \{1, 2, \ldots, n\},
$$

(3)

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

Therefore, with all generators from the generator space $\Omega$, the possible downstream space is defined as $\mathcal{D} = \{ G \leftarrow f | f \in \Omega \}$. 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, 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

\(^1\)For 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 $f : [0, 1]^2 \rightarrow [0, 1]$. 

---

Figure 2: Illustration of our proposed framework W2PGNN to answer when to pre-train GNNs.
W2PGN 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 $\Omega$, 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 $\Omega$, which can be formulated as an optimization problem as follows.

**Definition 2 (Feasibility of Graph Pre-training).** Given the pre-training data $\mathcal{G}_{\text{train}}$ and downstream data $\mathcal{G}_{\text{down}}$, we have the feasibility of performing pre-training on $\mathcal{G}_{\text{train}}$ to benefit $\mathcal{G}_{\text{down}}$ as

$$\zeta(\mathcal{G}_{\text{train}} \rightarrow \mathcal{G}_{\text{down}}) = \sup_{\{a_i\}, \{B_i\}} \Pr(\mathcal{G}_{\text{down}} \mid f(\{a_i\}, \{B_i\})),$$  \hspace{1cm} (4)

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

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

$$\zeta(\mathcal{G}_{\text{train}} \rightarrow \mathcal{G}_{\text{down}}) = -\inf_{\{a_i\}, \{B_i\}} \text{dist}(f(\{a_i\}, \{B_i\}), B_{\text{down}}).$$  \hspace{1cm} (5)

Following [42], we hire the 2-order Gromov-Wasserstein (GW) distance as our distance function to measure the difference between structured data.

### 4.3 Choose Graphon Basis to Approximate Feasibility

Although the feasibility has been converted to the optimization problem (5), exhausting all possible $\{a_i\}, \{B_i\}$ 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 $\{a_i\}$ 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 $\{B_i\}_{\text{integ}}$. 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 $\{B_i\}_{\text{domain}}$. 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 into their domains, and each split of subgraphs 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 $\{B_i\}_{\text{domain}}$.

**Topological graphon basis.** The third candidate is the topological graphon basis $\{B_i\}_{\text{topo}}$. 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 \( \{ \phi_1(v), \ldots, \phi_m(v) \} \) 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 \( \text{AGG} \) to summarize the node-level property of all nodes over \( G \) to a real number 

\[
\text{AGG}(\{\phi(v), v \in G\}) = \{\text{AGG}(\phi_1(v), v \in G), \ldots, \text{AGG}(\phi_m(v), v \in G)\}.
\]

In practice, we calculate degree [2], clustering coefficient [17] and closeness centrality [6] for each node and instantiate the aggregation function \( \text{AGG} \) 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

\[
h_{\text{graph}}(G) = \{\psi_1(G), \ldots, \psi_m(G)\},
\]

where density [37], assortativity [27], transitivity [37] are adopted as graph-level properties here.

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

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

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 [24] 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 \( \mathcal{B}_1 \) can be specified to one of them. In this way, the pre-training feasibility (simplified as \( \zeta \)) could be approximated in the reduced search space of graphon basis as

\[
\zeta \leftarrow -\text{MIN}(\inf_{\{\alpha_t\}} \text{dist}(f(\{\alpha_t\}, \mathcal{B}_1), \mathcal{B}_{\text{down}}), \forall \mathcal{B}_1 \in \mathcal{B}),
\]

where \( \mathcal{B} = \{\mathcal{B}_1_{\text{topo}}, \mathcal{B}_1_{\text{domain}}, \mathcal{B}_1_{\text{integ}}\} \) is the reduced search space of \( \mathcal{B}_1 \). Thus, the problem can be naturally split into three sub-problems with objective of dist \((f(\{\alpha_t\}, \mathcal{B}_1), \mathcal{B}_{\text{down}}))\), dist \((f(\{\alpha_t\}, \mathcal{B}_1), \mathcal{B}_{\text{down}}))\) and dist \((f(\{\alpha_t\}, \mathcal{B}_1), \mathcal{B}_{\text{down}}))\) respectively. Each sub-problem can be solved by updating the corresponding learnable parameters \( \{\alpha_t\} \) with multiple gradient descent steps. Taking one step as an example, we have

\[
\{\alpha_t\} = \{\alpha_t\} - \eta \nabla_{\{\alpha_t\}} \text{dist}(f(\{\alpha_t\}, \mathcal{B}_1), \mathcal{B}_{\text{down}})
\]

where \( \eta \) is the learning rate. Finally, we achieve three infimum distances under different \( \mathcal{B}_1 \) 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 GW distance, i.e., entropic regularization GW distance [29], as the distance function. For graphon estimation, we use the “largest gap” method as to estimate graphon \( \mathcal{B}_1 \).

---

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 W2PNN 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 \( O(l_1 l_2 ((n_1 + n_2)(|V|^3 + |E|d) + n_1 bd)) \).

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 W2PNN. 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 \( \mathcal{B}_1 \), the corresponding generator space \( \Omega = \{f(\{\alpha_t\}, \mathcal{B}_1) \mid \forall \{\alpha_t\}, \mathcal{B}_1 \} \) is the convex hull of \( \mathcal{B}_1 \).

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 [25]).** 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 [50]. For example, the motif \( (\alpha_{\omega}) \) has the same meaning of “feedforward loop” across networks of control system, gene networks 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 [21]).** 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., $t(F, B) = \lim_{i \to \infty} t(F, \{G_i\})$.

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 generated graph and that into our generator.

**Theorem 5.2.** Assume a graphon basis $\{B_1, \ldots, B_k\}$ 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

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

where $|F_a|$ represents the number of nodes in motif $F_a$, $\| \cdot \|_2$ 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 $\mathcal{G} = \mathcal{G}(n, f(\{\alpha_i\}, \{B_i\}))$ with $n$ nodes generated from $f(\{\alpha_i\}, \{B_i\})$. The homomorphism density of graph motif $F$ in $\mathcal{G}$ can be considered approximately equal to that in $f(\{\alpha_i\}, \{B_i\})$ with high probability and can be represented as

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

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 [30] 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 [30] 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 [31] 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 [39].

**Baseline of graph pre-training measures.** The baselines can be divided into 3 categories: (1) EGI [52] 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 $\alpha_i$, we adopt Adam as the optimizer and set the learning rate $\eta = 0.05$. For the GW distance, we adopt its differential and efficient version entropic regularization GW distance with default hyperparameters [29]. We provide an open-source implementation of our model W2PGNN at https://github.com/caoyxuan/W2PGNN.
We directly use the default hyper-parameters of pre-training models except the rw-hops in GCC. During fine-tuning, we freeze the trained parameters of pre-trained models and utilize the logistic regression model as our evaluation samples. When constructing the pre-training data, we need to construct multiple \( \langle G_{\text{train}}, G_{\text{down}} \rangle \) sample pairs as our evaluation samples. When constructing the \( \langle G_{\text{train}}, G_{\text{down}} \rangle \) 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.

### 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 [45] and GCC models [30] 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 [45], GraphMAE [13] and JOAO [44]. 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 [30] 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 \( \langle G_{\text{train}}, G_{\text{down}} \rangle \) sample pairs as our evaluation samples. When constructing the \( \langle G_{\text{train}}, G_{\text{down}} \rangle \) sample pairs, 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.

<table>
<thead>
<tr>
<th>N = 2</th>
<th>N = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>W2PGNN (intergr)</td>
<td>-0.6086</td>
</tr>
<tr>
<td>W2PGNN (domain)</td>
<td>0.6774</td>
</tr>
<tr>
<td>W2PGNN (topo)</td>
<td>-0.2033</td>
</tr>
<tr>
<td>Betweenness Centrality</td>
<td>-0.2739</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>N = 2</th>
<th>N = 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACE</td>
<td>BBBP</td>
</tr>
<tr>
<td>W2PGNN (intergr)</td>
<td>0.7547</td>
</tr>
<tr>
<td>W2PGNN (domain)</td>
<td>0.7334</td>
</tr>
<tr>
<td>W2PGNN (topo)</td>
<td>0.6656</td>
</tr>
<tr>
<td>W2PGNN (( \alpha = 1 ))</td>
<td>0.6549</td>
</tr>
<tr>
<td>W2PGNN</td>
<td>0.7549</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>N = 2</th>
<th>US-Airport</th>
<th>Europe-Airport</th>
<th>H-index</th>
<th>Chameleon</th>
<th>Rank</th>
<th>US-Airport</th>
<th>Europe-Airport</th>
<th>H-index</th>
<th>Chameleon</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Datasets</td>
<td>65.62</td>
<td>55.65</td>
<td>75.22</td>
<td>46.81</td>
<td>-</td>
<td>65.62</td>
<td>55.65</td>
<td>75.22</td>
<td>46.81</td>
<td>-</td>
</tr>
<tr>
<td>Graph Statistics</td>
<td>64.20</td>
<td>53.36</td>
<td>74.30</td>
<td>44.31</td>
<td>4</td>
<td>62.27</td>
<td>54.58</td>
<td>72.88</td>
<td>43.87</td>
<td>5</td>
</tr>
<tr>
<td>Clustering Coefficient</td>
<td>64.96</td>
<td>57.37</td>
<td>74.30</td>
<td>43.21</td>
<td>2</td>
<td>62.27</td>
<td>53.86</td>
<td>72.88</td>
<td>45.93</td>
<td>3</td>
</tr>
<tr>
<td>Spectrum of Graph Laplacian</td>
<td>64.61</td>
<td>52.87</td>
<td>77.74</td>
<td>49.21</td>
<td>3</td>
<td>62.94</td>
<td>54.58</td>
<td>75.18</td>
<td>44.66</td>
<td>4</td>
</tr>
<tr>
<td>Betweenness Centrality</td>
<td>64.96</td>
<td>52.87</td>
<td>73.50</td>
<td>41.63</td>
<td>6</td>
<td>62.27</td>
<td>54.87</td>
<td>73.90</td>
<td>44.66</td>
<td>2</td>
</tr>
<tr>
<td>W2PGNN</td>
<td>64.96</td>
<td>57.88</td>
<td>77.24</td>
<td>45.54</td>
<td>1</td>
<td>63.95</td>
<td>57.59</td>
<td>75.68</td>
<td>46.07</td>
<td>1</td>
</tr>
</tbody>
</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 [30]. For graph classification tasks, we take GraphCL as the pre-training model as it provides multiple graph augmentation approaches and is more general [45].

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 graph-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


[9] Xiueting Han, Zhenhuan Huang, Bang An, and Jing Bai. 2021. Adaptive Transfer Learning on Graph Neural Networks. In SIGKDD.


[22] Mengying Sun, Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In SIGKDD.


[26] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, and Xia Hu. 2022. G-Mixup: Graph Contrastive Coding for Graph Neural Network Pre-Training. In SIGKDD.


[33] Xinyang 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.

[34] Mengying Sun, Jiezhong Qiu, Qibin Chen, and Jie Tang. 2021. Does Self-Supervision Help Graph Convolutional Networks?. In NeurIPS. Vol. 34. 15870–15882.


[36] Xueting Han, Zhenhuan Huang, Bang An, and Jing Bai. 2021. Adaptive Transfer Learning on Graph Neural Networks. In SIGKDD.


[38] Yuanming Sun, Jiezhong Qiu, Qibin Chen, and Jie Tang. 2021. Does Self-Supervision Help Graph Convolutional Networks?. In NeurIPS. Vol. 34. 5812–5823.

[39] Xinlei Chen, Yuxiao Dong, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In SIGKDD.


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 [21]) For every graphon $W$, generating a $W$-random graph $G(n, W)$ for $n = 1, 2, \ldots$ we get a graph sequence such that $G(n, W) \to W$ with probability 1.

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

A.2 Proof of Theorem 5.1

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

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

Next we prove that our generator preserves the properties of graphons.

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

$$f(\{a_i\}, \{B_i\})^T = (a_1 B_1 + \ldots + a_k B_k)^T$$

$$= a_1 B_1^T + \ldots + a_k B_k^T \quad (11)$$

$$= a_1 B_1 + \ldots + a_k B_k = f(\{a_i\}, \{B_i\}).$$

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

$$f(\{a_i\}, \{B_i\}) = (a_1 B_1 + \ldots + a_k B_k) \leq \sum_{i=1}^{k} a_i B_{\text{max}} = B_{\text{max}} \leq 1. \quad (12)$$

$$f(\{a_i\}, \{B_i\}) = (a_1 B_1 + \ldots + a_k B_k) \geq \sum_{i=1}^{k} a_i B_{\text{min}} \geq B_{\text{min}} \geq 0. \quad (13)$$

Thus, for a set of graphon basis $X$, the generator space $\Omega$ is the set of all convex combinations of basis elements in $X$, $\Omega$ 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.

**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 $q_i$ for the $i$-th term), altogether implying that

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

**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 $f(\{a_i\}, \{B_i\})$.

Applying Lemma 2, we have the following derivations:

$$|t(F_a, f(\{a_i\}, \{B_i\})) - t(F_a, B_a)| \leq |F_a| \|f(\{a_i\}, \{B_i\}) - B_a\|_\square$$

$$|t(F_a, f(\{a_i\}, \{B_i\})) - t(F_a, B_a)| \leq |F_a| \sum_{b=1}^{k} a_b B_b - k \sum_{b=1}^{k} a_b B_a]|\square$$

Combining with the triangle inequality, we have:

$$|t(F_a, f(\{a_i\}, \{B_i\})) - t(F_a, B_a)| \leq |F_a| \sum_{b=1}^{k} a_b (B_b - B_a)|\square. \quad (18)$$

A.4 Proof of Theorem 5.3

**Lemma 4.** (Corollary 10.4 in [21]) Let $W$ be a graphon, $n \geq 1$, $0 < \epsilon < 1$, and let $F$ be a simple graph, then the $W$-random graph $G = G(n, W)$ satisfies

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

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

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

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.
Best Performance

We show the time complexity of W2PNN and the traditional solution. Suppose that we have $n_1$ and $n_2$ (sub)graphs sampled from the 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 W2PNN consists of three components: computation of three graphon base ($\{B_i\}_{\text{topo}}$, $\{B_i\}_{\text{domain}}$, $\{B_i\}_{\text{inter}}$), graphon estimation of downstream data (i.e., $B_{\text{down}}$), computation of GW distance (i.e., dist($\cdot$, $\cdot$)).

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], (3) $O(n_1)$ [17] and (3) $O(n_1|V|^2)$ [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].

The computation of downstream data costs $O(n_2|V|^2)$ [3].

C. COMPUTATION COMPLEXITY ANALYSIS

We show the time complexity of W2PNN and the traditional solution. Suppose that we have $n_1$ and $n_2$ (sub)graphs sampled from the 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 W2PNN consists of three components: computation of three graphon base ($\{B_i\}_{\text{topo}}$, $\{B_i\}_{\text{domain}}$, $\{B_i\}_{\text{inter}}$), graphon estimation of downstream data (i.e., $B_{\text{down}}$), computation of GW distance (i.e., dist($\cdot$, $\cdot$)).

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], (3) $O(n_1)$ [17] and (3) $O(n_1|V|^2)$ [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].

The computation of downstream data costs $O(n_2|V|^2)$ [3].

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 [30] 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) [30], $O(n_1|E|d)$ (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 $O(n_2(|V|^2 + |E|d))$ and downstream predictor $O(n_2d)$ (which can be ignored), under the simple freezing mode. Thus the overall time complexity of traditional solution is $O(l_1l_2((n_1 + n_2)(|V|^2 + |E|d) + n_1bd))$. 

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 GraphCLL [45]. 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.