# An Empirical Study of Realized GNN Expressiveness

**Yanbo Wang** [1] **Muhan Zhang** [1]

## Abstract

Research on the theoretical expressiveness of Graph Neural Networks (GNNs) has developed rapidly, and many methods have been proposed to enhance the expressiveness. However, most methods do not have a uniform expressiveness measure except for a few that strictly follow the $k$-dimensional Weisfeiler-Lehman ($k$-WL) test hierarchy, leading to difficulties in quantitatively comparing their expressiveness. Previous research has attempted to use datasets for measurement, but facing problems with difficulty (any model surpassing 1-WL has nearly 100% accuracy), granularity (models tend to be either 100% correct or near random guess), and scale (only several essentially different graphs involved). To address these limitations, we study the realized expressive power that a practical model instance can achieve using a novel expressiveness dataset, BREC, which poses greater difficulty (with up to 4-WL-indistinguishable graphs), finer granularity (enabling comparison of models between 1-WL and 3-WL), a larger scale (consisting of 800 1-WL-indistinguishable graphs that are non-isomorphic to each other). We synthetically test 23 models with higher-than-1-WL expressiveness on BREC. Our experiment gives the first thorough measurement of the realized expressiveness of those state-of-the-art beyond-1-WL GNN models and reveals the gap between theoretical and realized expressiveness. Dataset and evaluation codes are released at: https://github.com/GraphPKU/BREC.

## 1. Introduction

GNNs have been extensively utilized in bioinformatics, recommender systems, social networks, and others, yielding remarkable outcomes (Duvenaud et al., 2015; Barabási et al., 2011; Fan et al., 2019; Wang et al., 2018b; Berg et al., 2017; Zhou et al., 2020). Despite impressive empirical achievements, related investigations have revealed that GNNs exhibit limited abilities to distinguish some similar but non-isomorphic graphs. In practical situations, the inability to recognize specific structures, such as benzene rings (6-member rings), may cause misleading learning results. Xu et al. (2019); Morris et al. (2019) established a connection between the expressiveness of message-passing neural networks (MPNNs) and the WL test for graph isomorphism testing, demonstrating that MPNN's upper bound is 1-WL. Numerous subsequent studies have proposed GNN variants with enhanced expressiveness (Bevilacqua et al., 2022; Cotta et al., 2021; You et al., 2021; Zhang & Li, 2021).

Given the multitude of models employing different approaches, such as feature injection, equivariance maintenance, and subgraph extraction, a unified framework that can theoretically compare the expressive power among various variants is highly desirable. In this regard, several attempts have been made under the $k$-WL architecture. Maron et al. (2019b) propose the concept of $k$-order invariant/equivariant graph networks, which unify linear layers while preserving permutation invariance/equivariance. Additionally, Frasca et al. (2022) unify recent subgraph GNNs and establish that their expressiveness upper bound is 3-WL. Zhang et al. (2023a) further construct a comprehensive expressiveness hierarchy for subgraph GNNs. Nonetheless, the magnitude of the gaps remains unknown. Furthermore, there exist methods that are difficult to categorize within the $k$-WL hierarchy. For instance, Papp & Wattenhofer (2022) propose four extensions of GNNs, each of which cannot strictly compare with the other. Similarly, Feng et al. (2022) propose a GNN that is partially stronger than 3-WL yet fails to distinguish many 3-WL-distinguishable graphs. In a different approach, Huang et al. (2023) propose evaluating expressiveness by enumerating specific significant substructures, such as 6-cycles. Zhang et al. (2023b) introduces biconnectivity as a measurement. Zhang et al. (2024) employs homomorphism to provide a more intricate framework.

Without a unified theoretical characterization of expressiveness, employing expressiveness datasets for empirical testing proves valuable. Notably, three expressiveness datasets, EXP, CSL, and SR25, and some other combinations of difficult graphs have been introduced by Abboud et al. (2021);

[1]Institute of Artificial Intelligence, Peking University, Beijing, China. Correspondence to: Muhan Zhang <muhan@pku.edu.cn>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Murphy et al. (2019); Balcilar et al. (2021); Bouritsas et al. (2022) and have found widespread usage in recent studies. However, the empirical results failed to contribute further to the research due to notable limitations. Firstly, they lack sufficient difficulty. The EXP and CSL datasets solely consist of examples where 1-WL fails, and most recent GNN variants have achieved perfect accuracy on these datasets. Secondly, the granularity of these datasets is too coarse, which means that graphs in these datasets are generated using a single method, resulting in a uniform level of discrimination difficulty. Consequently, the performance of GNN variants often falls either at random guessing (completely indistinguishable) or 100% (completely distinguishable), thereby hindering the provision of a nuanced measure of expressiveness. Lastly, these datasets suffer from small sizes, typically comprising only a few substantially different graphs, raising concerns of incomplete measurement.

To overcome the limitations of previous datasets for a more meaningful empirical evaluation of realized expressiveness, i.e., the expressiveness that a practical model instance can achieve, we first propose BREC, a new expressiveness dataset containing 1-WL-indistinguishable graphs in 4 major categories: Basic, Regular, Extension, and CFI graphs. Compared to previous datasets, BREC has a greater difficulty (up to 4-WL-indistinguishable), finer granularity (can compare models between 1-WL and 3-WL), and larger scale (800 non-isomorphic graphs organized into 400 pairs, which can be easily extended to 319600 pairs or even more).

Due to the increased size and diversity of the dataset, the traditional classification task may not be suitable for training-based evaluation methods relying on generalization. Thus, we propose a novel evaluation procedure based on directly comparing the discrepancies between model outputs to test pure practical expressiveness. Acknowledging the impact of numerical precision owning to tiny differences between graphs, we propose reliable paired comparisons building upon a statistical method (Fisher, 1992; Johnson & Wichern, 2007), which offers a precise error bound. Experiments verify that it aligns well with known theoretical results.

Finally, we comprehensively compared 23 representative beyond-1-WL models on BREC. Our experiments for the first time give a **reliable empirical comparison** of state-of-the-art GNNs' realized expressiveness. Our results facilitate gaining deeper insights into various schemes to enhance GNNs' expressiveness. On BREC, GNN accuracies range from 41.5% to 70.2%, with $I^2$-GNN (Huang et al., 2023) performing the best. The 70.2% highest accuracy also implies that the dataset is **far from saturation**. Our evaluation code is in https://github.com/GraphPKU/BREC.

## 2. Limitations of Existing Datasets

**Preliminary.** We utilize the notation $\{\}$ to represent sets and $\{\{\}\}$ to represent multisets. The cardinality of a (multi)set $\mathbb{S}$ is denoted as $|\mathbb{S}|$. The index set is denoted as $[n] = 1, \ldots, n$. A graph is denoted as $\mathcal{G} = (\mathbb{V}(\mathcal{G}), \mathbb{E}(\mathcal{G}))$, where $\mathbb{V}(\mathcal{G})$ represents the set of *nodes* or *vertices* and $\mathbb{E}(\mathcal{G})$ represents the set of *edges*. Without loss of generality, we assume $|\mathbb{V}(\mathcal{G})| = n$ and $\mathbb{V}(\mathcal{G}) = [n]$.

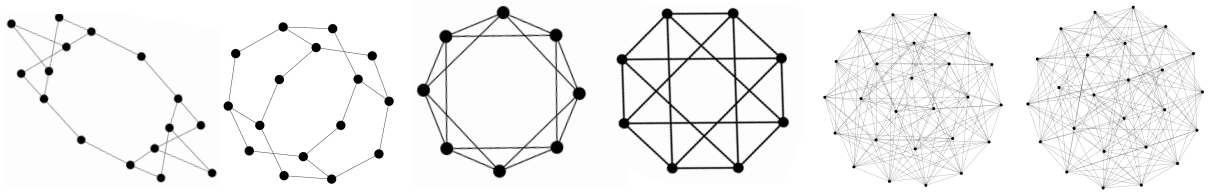
The permutation or reindexing of $\mathcal{G}$ is denoted as $\mathcal{G}^\pi = (\mathbb{V}(\mathcal{G}^\pi), \mathbb{E}(\mathcal{G}^\pi))$ with the permutation function $\pi : [n] \to [n]$, s.t. $(u, v) \in \mathbb{E}(\mathcal{G}) \iff (\pi(u), \pi(v)) \in \mathbb{E}(\mathcal{G}^\pi)$. Here node and edge features are excluded from definitions for briefness. Additional discussions are in Appendix B.

**Graph Isomorphism (GI) Problem.** Two graphs $\mathcal{G}$ and $\mathcal{H}$ are considered isomorphic (denoted as $\mathcal{G} \simeq \mathcal{H}$) if $\exists\ \phi$(a bijection mapping) $:\mathbb{V}(\mathcal{G}) \to \mathbb{V}(\mathcal{H})$ s.t. $(u, v) \in \mathbb{E}(\mathcal{G})$ iff. $(\phi(u), \phi(v)) \in \mathbb{E}(\mathcal{H})$. GI is essential in expressiveness. Only if a GNN successfully distinguishes two non-isomorphic graphs can they be assigned different labels. Some researchers (Chen et al., 2019; Geerts & Reutter, 2022) indicate the equivalence between GI and function approximation, underscoring the importance of GI. However, we currently do not have polynomial-time algorithms for solving the GI problem. A naive solution involves iterating all $n!$ permutations to test whether such a bijection exists.

**Weisfeiler-Lehman algorithm (WL).** WL is a well-known isomorphism test relying on color refinement (Weisfeiler & Leman, 1968). In each iteration, WL assigns a state (or color) to each node by aggregating information from its neighboring nodes' states. This process continues until convergence, resulting in a multiset of node states representing the final graph representation. While WL effectively identifies most non-isomorphic graphs, it may fail in certain simple graphs, leading to the development of extended versions. One such extension is $k$-WL, which treats each $k$-tuple of nodes as a unit for aggregating information. Another slightly different method (Cai et al., 1989) is also referred to as $k$-WL. To avoid confusion, we follow Morris et al. (2019) to call the former $k$-WL and the latter $k$-FWL. Further information can be found in Appendix C.

Given the significance of GI and WL, several expressiveness datasets have been introduced, with the following three being the most frequently utilized. We selected a pair of graphs from each dataset, illustrated in Figure 1. Detailed statistics for these datasets are presented in Table 1.

**EXP Dataset.** This dataset is generated pairwise. Each graph in a pair includes two disconnected components, the core component and planar component, where the former are two 1-WL-indistinguishable counterexamples while the latter are identical and only for adding noise. The task is binary classification based on whether core component sat-



(a) EXP dataset core pair sample (b) CSL graph($m = 10, r = 2/3$) (c) SR25 dataset sample

*Figure 1.* Sample graphs in previous datasets

*Table 1.* Dataset statistics

| Dataset | # Graphs | # Core graphs[a] | # Nodes | Distinguishing difficulty | Evaluation metrics |
|---|---|---|---|---|---|
| EXP | 1200 | 6 | 33-73 | 1-WL-indistinguishable | 2-way classification |
| CSL | 150 | 10 | 41 | 1-WL-indistinguishable | 10-way classification |
| SR25 | 15 | 15 | 25 | 3-WL-indistinguishable | 15-way classification |
| **BREC** | 800 | **800** | **10-198** | **1-WL to 4-WL-indistinguishable** | Reliable Paired Comparisons |

[a] Core graphs represent graphs that actually serve to measure expressiveness.

isfies the SAT condition. Although it is formally consistent with general datasets, the insufficient difficulty and number of different core components (**only three substantially different pairs**) results in most recent GNNs achieving nearly 100% accuracy, making detailed comparisons unavailable.

**CSL Dataset.** This dataset consists of Circulant Skip Links (CSL) graphs, which are 1-WL-indistinguishable 4-degree regular graphs. 10 distinct CSL graphs are generated and reindexed 15 times, resulting in the dataset with 150 graphs. The task is to train a 10-way classification model distinguishing each type. Because of the relatively low difficulty and their fixed structure (only **ten essentially different graphs with the same number of nodes and degree**), many recent expressive GNN models achieve close to 100% accuracy. More details about CSL graphs are in Appendix D.

**SR25 Dataset.** This dataset consists of 15 3-WL-indistinguishable Strongly Regular Graphs (SR) with parameter srg(25,12,5,6)[1]. In practice, SR25 is transformed into a 15-way classification problem for mapping each graph into a different class where the training and test graphs overlap. Most methods obtain 6.67% (1/15) accuracy due to 3-WL's high expressiveness. However, some methods partially surpassing 3-WL can even achieve 100% accuracy.

These three datasets have limitations regarding difficulty, granularity, and scale. In terms of difficulty, they are all bounded by 3-WL, failing to evaluate models (partly) beyond 3-WL (Feng et al., 2022). In terms of granularity, the graphs are generated in one way with repetive parameters, which easily leads to a 0/1 step function of model performance and cannot measure subtle differences between models. In terms of scale, the number of substantially different graphs in the datasets is small, and the test results may be incomplete to reflect expressiveness measurement.

[1]Strongly regular graphs can be described by four parameters. More details can be found in Appendix A

## 3. BREC: A New Dataset for Expressiveness

We propose a new expressiveness dataset, BREC, to address the limitations regarding difficulty, granularity, and scale. It consists of four major categories of graphs: Basic, Regular, Extension, and CFI. Basic graphs include relatively simple 1-WL-indistinguishable graphs. Regular graphs include four types of subcategorized regular graphs. Extension graphs include special graphs that arise when comparing four kinds of GNN extensions (Papp & Wattenhofer, 2022). CFI graphs include graphs generated by CFI methods[2] (Cai et al., 1989) with high difficulty. Some samples are shown in Fig 2.

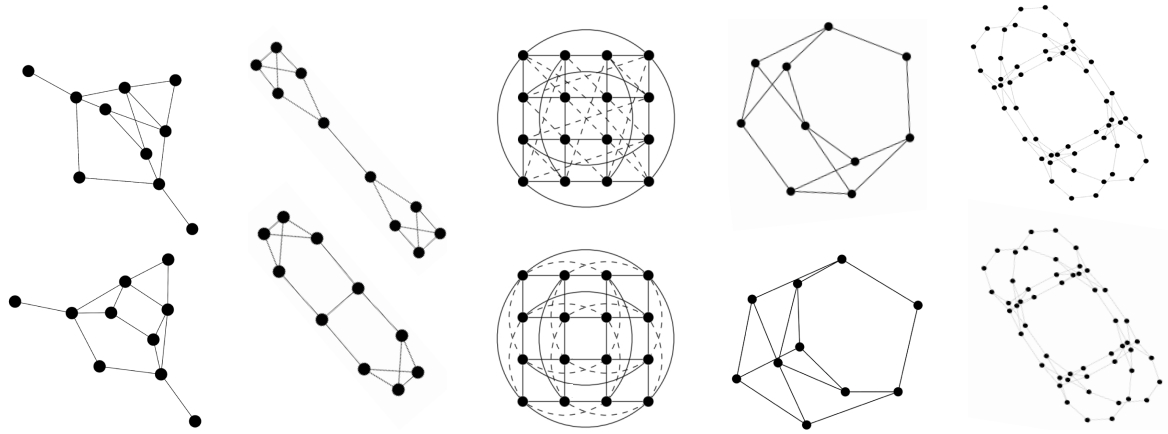
### 3.1. Dataset Composition

BREC includes 800 non-isomorphic graphs arranged in a pairwise manner to construct 400 pairs, with detailed composition as follows: (For detailed generation process, please refer to Appendix P)

**Basic Graphs.** Basic graphs consist of 60 pairs of 1-WL-indistinguishable graphs from an exhaustive search and intentionally designed to be non-regular. This part can be regarded as an augmentation of the EXP dataset with similar difficulty. Nevertheless, it offers a greater abundance of instances and more intricate graph patterns. The relatively small size also facilitates visualization and analysis.

**Regular Graphs.** Regular graphs consist of 140 pairs of reg-

[2]CFI is short for Cai-Furer-Immerman algorithm, which can generate counterexample graphs for any k-WL.



(a) Basic (b) Simple Regular (c) Strongly regular (d) Extension (e) CFI

*Figure 2.* BREC dataset samples

ular graphs, subcategoried to simple regular graphs, strongly regular graphs, 4-vertex condition graphs and distance regular graphs. A regular graph refers to a graph where all nodes possess the same degree. It is 1-WL-indistinguishable, and some studies delve into the analysis of GNN expressiveness from this perspective (Li et al., 2020; Zhang & Li, 2021). By expanding upon the existing subdivisions of regular graphs, this section widens the range of difficulty and complexity. Moreover, unlike the previous datasets, regular graphs are not limited to sharing identical parameters for all graphs within each category, greatly enhancing diversity. More details about regular graphs can be found in Appendix A.

**Extension Graphs.** Extension graphs consist of 100 pairs of graphs inspired by Papp & Wattenhofer (2022). They proposed 4 types of theoretical GNN extensions: $k$-WL hierarchy ($k$-WL), substructure-counting ($S_k$), $k$-hop-subgraph ($N_k$), and node-marking ($M_k$). These extensions do not possess a strict comparative relationship with each other. Leveraging the insights from theoretical analysis and some empirically derived findings, we generated and sampled graphs between 1-WL and 3-WL distinguishing difficulty to improve granularity. For more detailed definition of the GNN extensions, please refer to Appendix E.

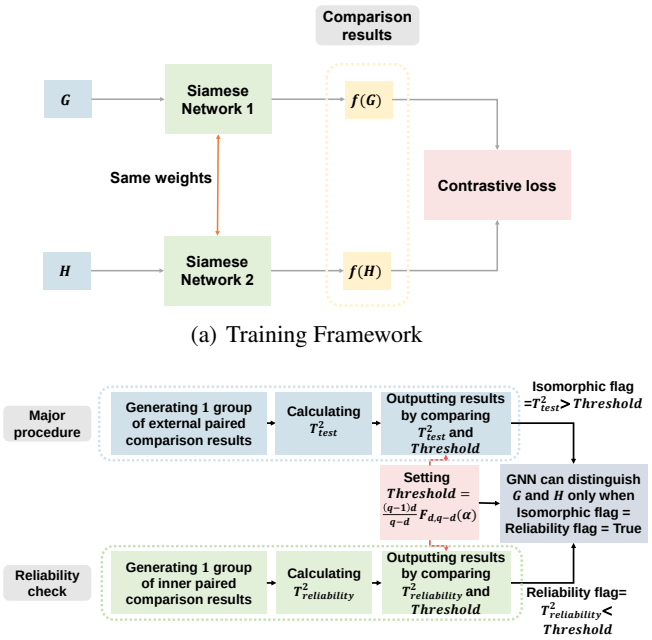
**CFI Graphs.** CFI graphs consist of 100 pairs of graphs inspired by Cai et al. (1989). They developed a method to generate graphs distinguishable by $k$-WL but not by $(k-1)$-WL for any $k$. We firstly implement it and create 100 pairs of graphs spanning up to 4-WL-indistinguishable, even surpassing the current research's upper bounds. As the most challenging part, it pushes the upper limit of difficulty even higher. Furthermore, the graph sizes in this section are larger than other parts (up to 198 nodes). It intensifies the challenge of the dataset, demanding a model's ability to process graphs with heterogeneous sizes effectively.

### 3.2. Advantages

**Difficulty.** The CFI graphs raise difficulty to 4-WL-indistinguishable. The newly involved 4-vertex condition and distance regular graphs also pose greater challenges.

**Granularity.** The different classes of graphs in BREC exhibit varying difficulty levels, each contributing to the dataset in distinct ways. Basic graphs contain fundamental 1-WL-indistinguishable graphs as a starting point. Regular graphs extend the CSL and SR25 datasets. The major components of regular graphs are simple regular graphs and strongly regular graphs, where 1-WL and 3-WL fail, respectively. Including 4-vertex condition graphs and distance regular graphs further elevates the complexity. Extension graphs bridge the gap between 1-WL and 3-WL, offering a finer-grained comparison for evaluating models beyond 1-WL. CFI graphs span the difficulty up to 4-WL-indistinguishable. By comprehensive compositions, BREC explores the boundaries of graph pattern distinguishability.

**Scale.** While previous datasets relied on only a few essentially different graphs, BREC utilizes a collection of 800 different graphs organized as 400 pairs. This significant increase in the number of graphs greatly enhances the diversity. The larger graph set in BREC also contributes to a more varied distribution of graph statistics (Appendix F). In contrast, previous datasets such as CSL and SR25 only have the same number of nodes and degrees across all graphs. BREC can also be easily further scaled up to 319600 pairs by iterating over all possible compare combinations, or even more by adding more graphs as it is a small sample of what we have done in the generation process (Appendix P). However, we deliberately did not scale it to facilitate a lower testing burden and balance the distribution of graphs with different difficulties. The experiments also verify that current size is enough to find subtle differences between models.




(a) Training Framework

(b) RPC Pipeline

*Figure 3.* Evaluation Method. (a) illustrates the training framework, where two non-isomorphic graphs are input into a Siamese network architecture to increase the distance between their respective representations. (b) presents the RPC (Reliability and Performance Characterization) pipeline, which comprises two main components: the Major Procedure and the Reliability Check. The Major Procedure operates on non-isomorphic pairs to quantify the external differences, whereas the Reliability Check is performed on isomorphic pairs to assess internal fluctuations. We calculate the corresponding $T^2$-statistics and compare them with a predefined threshold. A pair is considered successfully distinguished only if it passes both tests.

## 4. RPC: A New Evaluation Diagram

This section introduces a novel training framework and evaluation diagram for BREC. Unlike previous datasets, BREC departs from the conventional classification setting, where each graph is assigned a label, a classification model is trained, and the accuracy on test graphs serves as the measure of expressiveness. The labeling schemes used in previous datasets like semantic labels based on SAT conditions in EXP, or distinct labels for essentially different graphs in CSL and SR25, do not apply to BREC. There are two primary reasons. First, BREC aims to enrich the diversity of graphs, which precludes using a semantic label tied to SAT conditions, as it would significantly limit the range of possible graphs. Second, assigning a distinct label to each graph in BREC would result in an 800-class classification problem, where performance could be influenced by factors other than expressiveness. Our core idea is to measure models' practical "separating power" directly. Thus BREC is organized in pairs, where each pair is individually tested (i.e., we train an individual GNN for each pair) to determine whether a GNN can distinguish them. By adopting a pairwise evaluation method, BREC provides a more focused measure of models' expressiveness, aligning to assess distinguishing ability.

Nevertheless, how can we say a pair of graphs is successfully distinguished? Previous researchers tend to set a small threshold (like 1E-4) manually. If the embedding distance between them is larger than the threshold, the GNN is considered can distinguish them. This setting may satisfy previous datasets' requirements due to the relatively simple construction. However, it lacks **reliability** on numerical precision with more complex graphs where tiny differences may even overlap with numerical fluctuations. In order to yield dependable outcomes, we propose an evaluation method measuring both **external difference** and **internal fluctuations**. Furthermore, we introduce a training framework for pairwise data, employing the siamese network design (Koch et al., 2015) and contrastive loss (Hadsell et al., 2006; Wang et al., 2018a). The pipeline is depicted in Fig 4(a). We also did an ablation study on the effectiveness of our RPC, shown in Appendix H.

### 4.1. Training Framework

We adhere to the siamese network design (Koch et al., 2015) to train a model to distinguish each pair of graphs. The central component consists of two identical models maintaining identical parameters. For a pair of graphs inputted, it outputs a pair of embeddings. Subsequently, the difference between them is assessed using cosine similarity. The loss function is formulated as follows:

$$L(f, \mathcal{G}, \mathcal{H}) = \text{Max}(0, \frac{f(\mathcal{G}) \cdot f(\mathcal{H})}{||f(\mathcal{G})||\ ||f(\mathcal{H})||} - \gamma), \quad (1)$$

where the GNN model $f : \{\mathcal{G}\} \to \mathbb{R}^d$, $\mathcal{G}$ and $\mathcal{H}$ are two graphs, and $\gamma$ is a margin hyperparameter (set to 0 in our experiments). The loss function aims to promote the cosine similarity value lower than $\gamma$, thereby encouraging a greater separation between the two graph embeddings.

The training process **yields several benefits** for the models. Firstly, it helps the GNN to achieve its theoretical expressiveness. The GNN expressiveness analysis focuses primarily on the network's structure without imposing any constraints on its parameters, which means it is exploring the expressiveness of **a group of functions**. If a model with particular parameters can distinguish a pair of graphs, the model's design and structure are considered possessing sufficient expressiveness. However, it is impractical to iterate all possible parameter combinations to test the real upper bound. Hence, training can **realize searching** in the function space, enabling models to achieve better practical expressiveness. Furthermore, training aids components to **possess specific properties**, such as injectivity

and universal approximation, which are vital for attaining theoretical expressiveness. These properties require specific parameter configurations, but randomly initialized parameters may not satisfy. Moreover, through training, model-distinguishable pairs are **more easily discriminated** from model-indistinguishable pairs, which helps reducing the false negative rate caused by numerical precision. The difference between model-distinguishable pairs' embeddings is further magnified in the pairwise contrastive training process. However, the difference for model-indistinguishable pairs caused by numerical precision remains unaffected mainly. The framework is shown in Fig 4(a). We also conducted an ablation study comparing training and random weights, as presented in Appendix G.

### 4.2. Evaluation Method

We evaluate models by comparing the outputs of two non-isomorphic graphs. If we notice a significant difference between the outputs, we conclude that the GNN can distinguish the pair of graphs. However, setting a suitable threshold can be challenging. A large threshold may yield false negatives, which means the model can distinguish the pair, but the observed difference falls short of the threshold. Conversely, a small threshold may yield false positives, which means the model cannot distinguish the pair, but fluctuating errors cause the difference to exceed the threshold.

To address the issue of fluctuating errors, we draw inspiration from Paired Comparisons (Fisher, 1992). It involves comparing two groups of results instead of a single pair. The influence of random errors is mitigated by repeatedly generating results and comparing the two groups of results. Building upon it, we introduce a method called **R**eliable **P**aired **C**omparison (RPC) to verify whether a GNN genuinely produces distinct outputs for a pair of graphs. The pipeline is depicted in Fig 4(b).

RPC consists of two main components: Major Procedure and Reliability Check. The Major Procedure is conducted on a pair of non-isomorphic graphs to measure their dissimilarity. In contrast, the Reliability Check is conducted on graph automorphisms to capture internal fluctuations.

**Major Procedure.** Given two non-isomorphic graphs $\mathcal{G}, \mathcal{H}$, we create $q$ copies of each by random permutation (still isomorphic to original graph) to generate two groups of graphs, denoted as:

$$\mathcal{G}_i,\ \mathcal{H}_i,\ i \in [q]. \tag{2}$$

Supposing the GNN $f : \{\mathcal{G}\} \to \mathbb{R}^d$, we first calculate $q$ differences utilizing Paired Comparisons.

$$\boldsymbol{d}_i = f(\mathcal{G}_i) - f(\mathcal{H}_i),\ i \in [q]. \tag{3}$$

**Assumption 4.1.** $\boldsymbol{d}_i$ are independent $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ random vectors.

The above assumption is based on a more basic assumption that $f(\mathcal{G}_i),\ f(\mathcal{H}_i)$ follow Gaussian distributions, which presumes that random permutation only introduces Gaussian noise to the result.

If the GNN cannot distinguish $\mathcal{G}$ and $\mathcal{H}$, the mean difference should satisfy $\boldsymbol{\mu} = \mathbf{0}$. To check whether the equation holds, we can conduct an $\alpha$-level Hotelling's T-square test, comparing the hypotheses $H_0 : \boldsymbol{\mu} = \mathbf{0}$ against $H_1 : \boldsymbol{\mu} \neq \mathbf{0}$. The $T^2$-statistic for $\boldsymbol{\mu}$ is calculated as follows:

$$T^2 = q(\overline{\boldsymbol{d}} - \boldsymbol{\mu})^T \boldsymbol{S}^{-1} (\overline{\boldsymbol{d}} - \boldsymbol{\mu}), \tag{4}$$

where

$$\overline{\boldsymbol{d}} = \frac{1}{q}\sum_{i=1}^{q} \boldsymbol{d}_i,\ \boldsymbol{S} = \frac{1}{q-1}\sum_{i=1}^{q} (\boldsymbol{d}_i - \overline{\boldsymbol{d}})(\boldsymbol{d}_i - \overline{\boldsymbol{d}})^T. \tag{5}$$

Hotelling's T-square test proves that $T^2$ is distributed as an $\frac{(q-1)d}{q-d} F_{d,q-d}$ random variable, where $F_{d,q-d}$ represents $F$-distribution with degree of freedom $d, q-d$ (Hotelling, 1992). The theorem establishes a connection between the unknown parameter $\boldsymbol{\mu}$ and a definite distribution $F_{d,q-d}$, allowing us to confirm the confidence interval of $\boldsymbol{\mu}$ by testing the distribution fit. To test the hypothesis $H_0 : \boldsymbol{\mu} = \mathbf{0}$, we substitute $\boldsymbol{\mu} = \mathbf{0}$ into Equation (4), obtaining $T^2_{\text{test}} = q\overline{\boldsymbol{d}}^T \boldsymbol{S}^{-1} \overline{\boldsymbol{d}}$. Then an $\alpha$-level test of $H_0 : \boldsymbol{\mu} = \mathbf{0}$ versus $H_1 : \boldsymbol{\mu} \neq \mathbf{0}$ accepts $H_0$ (the GNN cannot distinguish the pair) if:

$$T^2_{\text{test}} = q\overline{\boldsymbol{d}}^T \boldsymbol{S}^{-1} \overline{\boldsymbol{d}} < \frac{(q-1)d}{(q-d)} F_{d,q-d}(\alpha), \tag{6}$$

where $F_{d,q-d}(\alpha)$ is the upper $(100\alpha)$th percentile of the $F$-distribution $F_{d,q-d}$ (Fisher, 1950) with $d$ and $q-d$ degrees of freedom. Similarly, we reject $H_0$ (the GNN can distinguish the pair) if

$$T^2_{\text{test}} = q\overline{\boldsymbol{d}}^T \boldsymbol{S}^{-1} \overline{\boldsymbol{d}} > \frac{(q-1)d}{(q-d)} F_{d,q-d}(\alpha). \tag{7}$$

**Reliability Check.** With an appropriate choice of $\alpha$, the Major Procedure provides a dependable confidence interval for assessing the distinguishability. However, manually selected $\alpha$ based on heuristics may not be optimal. Furthermore, computational precisions can introduce distribution shifts in assumed Gaussian fluctuations. To address this issue, we introduce the Reliability Check. It bridges external differences between two graphs and internal fluctuations within a single graph.

WLOG, we replace $\mathcal{H}$ by permutation of $\mathcal{G}$, i.e., $\mathcal{G}^\pi$. We can then obtain the internal fluctuations within $\mathcal{G}$ by comparing it with $\mathcal{G}^\pi$, and the external difference between $\mathcal{G}$ and $\mathcal{H}$ by comparing $\mathcal{G}$ and $\mathcal{H}$. We utilize the same step as Major

Procedure on $\mathcal{G}$ and $\mathcal{G}^\pi$, calculating the $T^2$-statistics as:

$$T^2_{\text{reliability}} = q\overline{\boldsymbol{d}}^T \boldsymbol{S}^{-1}\overline{\boldsymbol{d}}, \tag{8}$$

$$\text{where } \overline{\boldsymbol{d}} = \frac{1}{q}\sum_{i=1}^{q} \boldsymbol{d}_i, \boldsymbol{d}_i = f(\mathcal{G}_i) - f(\mathcal{G}_i^\pi), i \in [q], \tag{9}$$

$$\boldsymbol{S} = \frac{1}{q-1}\sum_{i=1}^{q}(\boldsymbol{d}_i - \overline{\boldsymbol{d}})(\boldsymbol{d}_i - \overline{\boldsymbol{d}})^T. \tag{10}$$

Recalling that $\mathcal{G}$ and $\mathcal{G}^\pi$ are isomorphic, the GNN should not distinguish between them, implying that $\boldsymbol{\mu} = \mathbf{0}$. Therefore, the test is considered reliable only if $T^2_{\text{reliability}} < \frac{(q-1)d}{(q-d)}F_{d,q-d}(\alpha)$. Combining the reliability and distinguishability results, we get the complete RPC (Fig 4(b)):

For each pair of graphs $\mathcal{G}$ and $\mathcal{H}$, we first calculate the threshold value, denoted as Threshold $= \frac{(q-1)d}{(q-d)}F_{d,q-d}(\alpha)$. Next, we conduct the Major Procedure on $\mathcal{G}$ and $\mathcal{H}$ for distinguishability and perform the Reliability Check on $\mathcal{G}$ and $\mathcal{G}^\pi$ for Reliability. Only when $T^2_{\text{test}}$ from the Major Procedure, and $T^2_{\text{reliability}}$ from the Reliability Check, satisfying $T^2_{\text{reliability}} <$ Threshold $< T^2_{\text{test}}$, do we conclude that the GNN can distinguishing $\mathcal{G}$ and $\mathcal{H}$.

We further propose **R**eliable **A**daptive **P**aired **C**omparisons (RAPC), aiming to adaptively adjust the threshold and provide an upper bound for false positive rates. In practice, we use **RPC** due to its less computational time and satisfactory performance. For more details, please refer to Appendix I.

## 5. Experiment

In this section, we evaluate the expressiveness of 23 representative models using our BREC dataset.

**Model selection.** We evaluate six categories of methods: non-GNN methods, subgraph-based GNNs, $k$-WL-hierarchy-based GNNs, substructure-based GNNs, transformer-based GNNs, and random GNNs. We implement four types of non-GNN baselines based on Papp & Wattenhofer (2022); Ying et al. (2021), including WL test (3-WL and SPD-WL), counting substructures ($S_3$ and $S_4$), k-hop neighbors ($N_1$ and $N_2$), and node marking ($M_1$). We implemented them by adding additional features during the WL test update or using heterogeneous message passing. Note that they are more theoretically significant than practical since they may require exhaustive enumeration or exact isomorphism encoding of various substructures. We additionally included 16 state-of-the-art GNNs, including NGNN (Zhang & Li, 2021), DE+NGNN (Li et al., 2020), DS/DSS-GNN (Bevilacqua et al., 2022), SUN (Frasca et al., 2022), SSWL_P (Zhang et al., 2023a), GNN-AK (Zhao et al., 2022a), KP-GNN (Feng et al., 2022), I$^2$-GNN (Huang et al., 2023), PPGN (Maron et al., 2019a), $\delta$-k-LGNN (Morris et al., 2020), KC-SetGNN (Zhao et al., 2022b), GSN (Bouritsas et al., 2022), DropGNN (Papp et al., 2021), OSAN (Qian et al., 2022), and Graphormer (Ying et al., 2021).

**Observation 1. The realized expressiveness closely conforms to the anticipated theoretical expectations in general. Moreover, it posits an empirical measurement for GNNs lacking precise theoretical expressiveness.**

Table 2 showcases the principal findings. $N_2$ performs best among non-GNNs, and I$^2$-GNN attains the highest accuracy among GNNs. Subgraph-based GNNs generally exhibit outcomes comparable to theoretical comparative results. The results on k-WL hierarchy-based GNNs reveal distinctions between global and local methods. Regarding substructure GNNs and random GNNs, we initially provide empirical evaluation results on expressiveness. Detailed experimental specifications are included in Appendix O. We also firstly implemented the tight bound expressiveness of GNNs to provide a more precise understanding of the realized expressiveness compared to the theoretical expressiveness. The detailed results can be found in Appendix N.

**Non-GNN baselines.** 3-WL successfully distinguishes all Basic graphs, Extension graphs, simple regular graphs and 60 CFI graphs as expected. $S_3$, $S_4$, $N_1$, and $N_2$ demonstrate excellent performance on small-radius graphs such as Basic, Regular, and Extension graphs. However, due to their limited receptive fields, they struggle to distinguish large-radius graphs like CFI graphs. Noting that the expressiveness of $S_3$ and $S_4$ is bounded by $N_1$ and $N_2$, respectively, as analyzed by Papp & Wattenhofer (2022). Conversely, $M_1$ is implemented by heterogeneous message passing, which makes it unaffected by large graph diameters, thus maintaining its performance across different graphs. SPD-WL is another 1-WL extension operated on a complete graph with shortest path distances as edge features. It may degrade to 1-WL on low-radius graphs, causing its relatively poor performance.

**Subgraph-based GNNs.** Regarding subgraph-based models, they can generally distinguish almost all Basic graphs, simple regular graphs and Extension graphs. However, an exception lies with NGNN, which performs poorly in Extension graphs due to its simplicial node selection policy and lack of node labeling. Two other exceptions are KP-GNN and I$^2$-GNN, both exhibiting exceptional performance in Regular graphs. KP-GNN can differentiate a substantial number of strongly regular graphs and 4-vertex condition graphs by peripheral subgraphs, surpassing the 3-WL partially. And I$^2$-GNN surpasses the limitations of 3-WL partially through its enhanced cycle-counting power.

**Observation 2. The realized expressiveness does not consistently increase as the subgraph radius expands. Hence, optimizing the realized expressiveness necessitates identifying an optimal subgraph radius value.**

A salient factor affecting the performance lies in the sub-

*Table 2.* Pair distinguishing accuracies on BREC

| Type | Model | Basic Graphs (60) | | Regular Graphs (140) | | Extension Graphs (100) | | CFI Graphs (100) | | Total (400) | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Number | Accuracy | Number | Accuracy | Number | Accuracy | Number | Accuracy | Number | Accuracy |
| Non-GNNs | 3-WL | 60 | 100% | 50 | 35.7% | 100 | 100% | 60 | 60.0% | 270 | 67.5% |
| | SPD-WL | 16 | 26.7% | 14 | 11.7% | 41 | 41% | 12 | 12% | 83 | 20.8% |
| | $S_3$ | 52 | 86.7% | 48 | 34.3% | 5 | 5% | 0 | 0% | 105 | 26.2% |
| | $S_4$ | 60 | 100% | 99 | 70.7% | 84 | 84% | 0 | 0% | 243 | 60.8% |
| | $N_1$ | 60 | 100% | 99 | 85% | 93 | 93% | 0 | 0% | 252 | 63% |
| | $N_2$ | 60 | 100% | 138 | 98.6% | 100 | 100% | 0 | 0% | 298 | 74.5% |
| | $M_1$ | 60 | 100% | 50 | 35.7% | 100 | 100% | 41 | 41% | 251 | 62.8% |
| Subgraph GNNs | NGNN | 59 | 98.3% | 48 | 34.3% | 59 | 59% | 0 | 0% | 166 | 41.5% |
| | DE+NGNN | 60 | 100% | 50 | 35.7% | 100 | 100% | 21 | 21% | 231 | 57.8% |
| | DS-GNN | 58 | 96.7% | 48 | 34.3% | 100 | 100% | 16 | 16% | 222 | 55.5% |
| | DSS-GNN | 58 | 96.7% | 48 | 34.3% | 100 | 100% | 15 | 15% | 221 | 55.2% |
| | SUN | 60 | 100% | 50 | 35.7% | 100 | 100% | 13 | 13% | 223 | 55.8% |
| | SSWL_P | 60 | 100% | 50 | 35.7% | 100 | 100% | 38 | 38% | 248 | 62% |
| | GNN-AK | 60 | 100% | 50 | 35.7% | 97 | 97% | 15 | 15% | 222 | 55.5% |
| | KP-GNN | 60 | 100% | 106 | 75.7% | 98 | 98% | 11 | 11% | 275 | 68.8% |
| | $I^2$-GNN | 60 | 100% | 100 | 71.4% | 100 | 100% | 21 | 21% | 281 | 70.2% |
| k-WL GNNs | PPGN | 60 | 100% | 50 | 35.7% | 100 | 100% | 23 | 23% | 233 | 58.2% |
| | $\delta$-k-LGNN | 60 | 100% | 50 | 35.7% | 100 | 100% | 6 | 6% | 216 | 54% |
| | KC-SetGNN | 60 | 100% | 50 | 35.7% | 100 | 100% | 1 | 1% | 211 | 52.8% |
| Substructure GNNs | GSN | 60 | 100% | 99 | 70.7% | 95 | 95% | 0 | 0% | 254 | 63.5% |
| Random GNNs | DropGNN | 52 | 86.7% | 41 | 29.3% | 82 | 82% | 2 | 2% | 177 | 44.2% |
| | OSAN | 56 | 93.3% | 8 | 5.7% | 79 | 79% | 5 | 5% | 148 | 37% |
| Transformer GNNs | Graphormer | 16 | 26.7% | 12 | 8.6% | 41 | 41% | 10 | 10% | 79 | 19.8% |

graph radius. GNNs yield superior real expressiveness as the subgraph radius expands and is expected as well for realized expressiveness. Nonetheless, in practical implementation, the expansion of the radius may engender the dilution of information, wherein the receptive field expands to encompass some extraneous or noisy data. Consequently, we consider the subgraph radius as a hyperparameter, fine-tuning it for each model, and present the optimal hyperparameters in Table 10. Further details can be found in Appendix J.

**Observation 3. Profound design and encoding elevate realized expressiveness and practical performance even without theoretical expressiveness improvement.**

The inclusion of distance encoding in DE+NGNN and $I^2$-GNN facilitates enhanced differentiation among distinct hops within a specified subgraph radius, particularly in larger subgraph radii. While certain node labeling techniques have been proven to possess equivalent expressiveness capabilities (Zhang et al., 2023a), it is consistently observed that distance encoding effectively amplifies realized expressiveness and real-world performance. Regarding DS-GNN, DSS-GNN, GNN-AK, SUN, and SSWL_P, they employ analogous aggregation schemes with minor variations in their operations. These models showcase comparable performance, with SSWL_P outperforming the rest, which aligns with the notion that SSWL_P achieves the highest degree of expressiveness through the implementation of meticulously selected components (Zhang et al., 2023a).

**$k$-WL hierarchy-based GNNs.** For the $k$-WL-hierarchy-based models, we adopt two implemented approaches: high-order simulation and local-WL simulation. PPGN serves as the representative work for the former, while $\delta$-k-LGNN and KCSet-GNN as the latter.

**Observation 4. The realized expressiveness does not consistently increase as the number of layer increases. Hence, optimizing the realized expressiveness necessitates identifying optimal number of layers .**

For GNNs emulating k-WL, the number of layers becomes a pivotal factor as each layer represents one WL iteration. Increasing the number of layers results in a progressive enhancement of expressiveness; nevertheless, it may encounter over-smoothing issues. Consequently, a diligent exploration of the optimal hyperparameters is conducted as in K. PPGN perfectly aligns its performance with 3-WL across all graphs except for CFI graphs with large radii. Nonetheless, PPGN still surpasses most GNNs in CFI graphs due to global $k$-WL's global receptive field. For $\delta$-k-LGNN, we set $k = 2$, while for KCSet-GNN, we set $k = 3, c = 2$ to simulate local 3-WL, adhering to the original configuration. By comparing the output results with relatively small diameters, we observed that local WL matches the performance of general $k$-WL. However, local WL exhibits lower performance for CFI graphs due to insufficient receptive fields.

**Substructure-based GNNs** For substructure-based GNNs, we select GSN, which incorporates substructure isomorphism counting as features. The best result obtained for GSN-e is reported when setting $k = 4$. For further explo-

ration of policy and size, please refer to Appendix L.

**Random GNNs** Random GNNs are unsuitable for GI problems since even identical graphs can yield different outcomes due to inherent randomness. However, the RPC can quantify fluctuations in the randomization process, thereby enabling testing for random GNNs. We test DropGNN and OSAN. For more details regarding the crucial factor of random samples, please refer to Appendix M.

**Transformer-based GNNs** For transformer-based GNNs, we select Graphormer, which is anticipated to possess expressiveness with SPD-WL. The results also verify that.

## 6. Conclusion and Future Work

This paper undertakes a thorough empirical re-evaluation of GNN expressiveness, presenting the most comprehensive findings thus far. To address previous limitations on measurement, this study introduces a novel dataset called BREC, along with a test methodology named RPC. The experiment reveals a noticeable disparity between theoretical expectations and realized expressiveness. The algorithms implemented for the first time also provide valuable tools.

In addition to the expressiveness comparison primarily based on graph isomorphism (GI), there exist several alternative metrics for expressiveness evaluation, such as substructure countings. However, they are often conducted on datasets not specifically designed for expressiveness (Huang et al., 2023; Zhao et al., 2022a; Chen et al., 2020), which can lead to biased results caused by spurious correlations. Certain methods may encounter difficulties in identifying a specific substructure, but they might successfully capture another property that is correlated with substructures, thereby yielding false indications of high performance. This problem can be alleviated by BREC with difficult graphs. We reveal the generation process of BREC in Appendix P to be utilized in more tasks. We also hope realized expressiveness will aid researchers exploring in other domains.

## Acknowledgements

This work is partially supported by the National Key R&D Program of China (2022ZD0160300), the National Key R&D Program of China (2021ZD0114702), the National Natural Science Foundation of China (62276003), and Alibaba Innovative Research Program.

## Impact Statement

This paper presents research aimed at enhancing the expressiveness of Graph Neural Networks (GNNs). While GNNs have the potential for wide-ranging applications with significant societal implications, this study specifically focuses on investigating and improving expressiveness, without delving into any specific real-world dataset applications. As a result, we do not emphasize any specific social impacts in this work.

## References

Abboud, R., Ceylan, İ. İ., Grohe, M., and Lukasiewicz, T. The surprising power of graph neural networks with random node initialization. In Zhou, Z. (ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021, pp. 2112–2118. ijcai.org, 2021. doi: 10.24963/ijcai.2021/291. URL https://doi.org/10.24963/ijcai.2021/291.

Babai, L. and Kucera, L. Canonical labelling of graphs in linear average time. In 20th Annual Symposium on Foundations of Computer Science (sfcs 1979), pp. 39–46. IEEE, 1979.

Balcilar, M., Héroux, P., Gauzere, B., Vasseur, P., Adam, S., and Honeine, P. Breaking the limits of message passing graph neural networks. In International Conference on Machine Learning, pp. 599–608. PMLR, 2021.

Barabási, A.-L., Gulbahce, N., and Loscalzo, J. Network medicine: a network-based approach to human disease. Nature reviews genetics, 12(1):56–68, 2011.

Berg, R. v. d., Kipf, T. N., and Welling, M. Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263, 2017.

Bevilacqua, B., Frasca, F., Lim, D., Srinivasan, B., Cai, C., Balamurugan, G., Bronstein, M. M., and Maron, H. Equivariant subgraph aggregation networks. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. URL https://openreview.net/forum?id=dFbKQaRk15w.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1):657–668, 2022.

Brouwer, A. E., Haemers, W. H., Brouwer, A. E., and Haemers, W. H. Strongly regular graphs. Spectra of graphs, pp. 115–149, 2012.

Brouwer, A. E., Ihringer, F., and Kantor, W. M. Strongly regular graphs satisfying the 4-vertex condition. Combinatorica, 43(2):257–276, apr 2023. doi: 10.1007/s00493-023-00005-y. URL https://doi.org/10.1007%2Fs00493-023-00005-y.

Cai, J.-Y., Furer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identification. In 30th Annual Symposium on Foundations of Computer Science, pp. 612–617, 1989. doi: 10.1109/SFCS.1989.63543.

Chen, Z., Villar, S., Chen, L., and Bruna, J. On the equivalence between graph isomorphism testing and function approximation with gnns. Advances in neural information processing systems, 32, 2019.

Chen, Z., Chen, L., Villar, S., and Bruna, J. Can graph neural networks count substructures? Advances in neural information processing systems, 33:10383–10395, 2020.

Cotta, L., Morris, C., and Ribeiro, B. Reconstruction for powerful graph representations. Advances in Neural Information Processing Systems, 34:1713–1726, 2021.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. Advances in neural information processing systems, 28, 2015.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In The world wide web conference, pp. 417–426, 2019.

Feng, J., Chen, Y., Li, F., Sarkar, A., and Zhang, M. How powerful are k-hop message passing graph neural networks. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), Advances in Neural Information Processing Systems, 2022. URL https://openreview.net/forum?id=nN3aVRQsxGd.

Fisher, R. A. Contributions to mathematical statistics. 1950.

Fisher, R. A. Statistical methods for research workers. Springer, 1992.

Frasca, F., Bevilacqua, B., Bronstein, M., and Maron, H. Understanding and extending subgraph gnns by rethinking their symmetries. Advances in Neural Information Processing Systems, 35:31376–31390, 2022.

Geerts, F. and Reutter, J. L. Expressiveness and approximation properties of graph neural networks. In The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, 2022. URL https://openreview.net/forum?id=wIzUeM3TAU.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), volume 2, pp. 1735–1742. IEEE, 2006.

Hotelling, H. The generalization of student's ratio. In Breakthroughs in statistics: Foundations and basic theory, pp. 54–65. Springer, 1992.

Huang, N. T. and Villar, S. A short tutorial on the weisfeiler-lehman test and its variants. In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 8533–8537. IEEE, 2021.

Huang, Y., Peng, X., Ma, J., and Zhang, M. Boosting the cycle counting power of graph neural networks with i$^2$-gnns. In The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, 2023. URL https://openreview.net/pdf?id=kDSmxOspsXQ.

Johnson, R. A. and Wichern, D. W. Applied multivariate statistical analysis. Pearson Prentice Hall, Upper Saddle River, N.J, 6th ed edition, 2007. ISBN 978-0-13-187715-3. OCLC: ocm70867129.

Koch, G., Zemel, R., Salakhutdinov, R., et al. Siamese neural networks for one-shot image recognition. In ICML deep learning workshop, volume 2. Lille, 2015.

Li, P., Wang, Y., Wang, H., and Leskovec, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. Advances in Neural Information Processing Systems, 33:4465–4478, 2020.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. Advances in neural information processing systems, 32, 2019a.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019b. URL https://openreview.net/forum?id=Syx72jC9tm.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 33, pp. 4602–4609, 2019.

Morris, C., Rattan, G., and Mutzel, P. Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. Advances in Neural Information Processing Systems, 33:21824–21840, 2020.

Murphy, R., Srinivasan, B., Rao, V., and Ribeiro, B. Relational pooling for graph representations. In International Conference on Machine Learning, pp. 4663–4673. PMLR, 2019.

Papp, P. A. and Wattenhofer, R. A theoretical comparison of graph neural network extensions. In International Conference on Machine Learning, pp. 17323–17345. PMLR, 2022.

Papp, P. A., Martinkus, K., Faber, L., and Wattenhofer, R. Dropgnn: Random dropouts increase the expressiveness of graph neural networks. Advances in Neural Information Processing Systems, 34:21997–22009, 2021.

Qian, C., Rattan, G., Geerts, F., Niepert, M., and Morris, C. Ordered subgraph aggregation networks. Advances in Neural Information Processing Systems, 35:21030–21045, 2022.

Sato, R. A survey on the expressive power of graph neural networks. arXiv preprint arXiv:2003.04078, 2020.

Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., Li, Z., and Liu, W. Cosface: Large margin cosine loss for deep face recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5265–5274, 2018a.

Wang, H., Zhang, F., Wang, J., Zhao, M., Li, W., Xie, X., and Guo, M. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In Proceedings of the 27th ACM international conference on information and knowledge management, pp. 417–426, 2018b.

Weisfeiler, B. and Leman, A. The reduction of a graph to canonical form and the algebra which appears therein. nti, Series, 2(9):12–16, 1968.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, Conference Track Proceedings. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

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

You, J., Gomes-Selman, J. M., Ying, R., and Leskovec, J. Identity-aware graph neural networks. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pp. 10737–10745, 2021.

Zhang, B., Feng, G., Du, Y., He, D., and Wang, L. A complete expressiveness hierarchy for subgraph GNNs via subgraph weisfeiler-lehman tests. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of Proceedings of Machine Learning Research, pp. 41019–41077. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/zhang23k.html.

Zhang, B., Luo, S., Wang, L., and He, D. Rethinking the expressive power of gnns via graph biconnectivity. In The Eleventh International Conference on Learning Representations, 2023b.

Zhang, B., Gai, J., Du, Y., Ye, Q., He, D., and Wang, L. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness, 2024.

Zhang, M. and Li, P. Nested graph neural networks. Advances in Neural Information Processing Systems, 34:15734–15747, 2021.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any GNN with local structure awareness. In International Conference on Learning Representations, 2022a. URL https://openreview.net/forum?id=Mspk_WYKoEH.

Zhao, L., Shah, N., and Akoglu, L. A practical, progressively-expressive gnn. Advances in Neural Information Processing Systems, 35:34106–34120, 2022b.

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. Graph neural networks: A review of methods and applications. AI open, 1:57–81, 2020.

## A. Details on Regular Graphs

In this section, we introduce the relationship between four types of regular graphs. The inclusion relations of them are shown in Figure 4, but their difficulty relations and inclusion relations are not consistent.

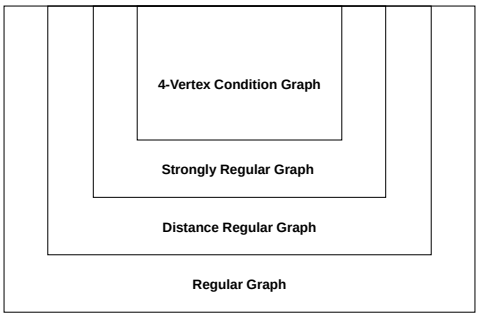



*Figure 4.* Regular graphs relationship

A graph is deemed a regular graph when all of its vertices possess an identical degree. If a regular graph, with $v$ vertices and degree $k$, satisfies the additional conditions wherein any two adjacent vertices share $\lambda$ common neighbors, and any two non-adjacent vertices share $\mu$ common neighbors, it is categorized as a strongly regular graph. Hence, it can be represented as $\text{srg}(v, k, \lambda, \mu)$, denoting its four associated parameters. It is worth mentioning that the diameter of a connected strongly regular graph is always 2 (Brouwer et al., 2012).

Regular graphs and strongly regular graphs find wide application in expressiveness analysis. The difficulty of strongly regular graphs surpasses that of general regular graphs due to the imposition of additional requirements. Notably, the simplest strongly regular graphs with identical parameters ($\text{srg}(16, 6, 2, 2)$) are exemplified by the Shrikhande graph and the $4 \times 4$-Rook's graph, as depicted in Figure 2(c).

Both 4-vertex condition graphs and distance regular graphs introduce heightened complexities, albeit in opposing directions. A 4-vertex condition graph is a strongly regular graph with an additional property that mandates the determination of the number of edges between the common neighbors of two vertices based on their connectivity (Brouwer et al., 2023). Conversely, distance regular graphs expand upon the definition of strongly regular graphs by specifying that for any two vertices $v$ and $w$, the count of vertices at a distance $j$ from $v$ and at a distance $k$ from $w$ relies solely on $j$, $k$, and the distance between $v$ and $w$. Notably, a distance regular graph with a radius of 2 is equivalent to a strongly regular graph.

The 4-vertex condition graph has yet to be explored in previous research endeavors. Similarly, instances of distance regular graphs are relatively scarce and analyzing them through examples proves to be challenging. To encourage further research in these domains, we have incorporated them into BREC.

## B. Node Features

In this section, we present the concept of node features and edge features in graphs.

We commence by providing the definition of graphs using an adjacency matrix representation. Consider a graph where the node features are represented by a $d_n$-dimensional vector, and the edge features are represented by a $d_e$-dimensional vector. This graph can be denoted as $\mathcal{G} = (\mathbf{V}(\mathcal{G}), \mathbf{E}(\mathcal{G}))$, where $\mathbf{V}(\mathcal{G}) \in \mathbb{R}^{n \times d_n}$ represents the node features, and $\mathbf{E}(\mathcal{G}) \in \mathbb{R}^{n \times n \times (d_e+1)}$ represents the edge features, with $n$ being the number of nodes in the graph. The adjacency matrix of the graph is denoted as $\mathbf{A}(\mathcal{G}) \in \mathbb{R}^{n \times n} = \boldsymbol{E}(\mathcal{G})_{:,:,(d_e+1)}$, where $\mathbf{A}(\mathcal{G})_{i,j} = 1$ if $(i, j) \in \mathbb{E}(\mathcal{G})$ (i.e., if nodes $i$ and $j$ are connected by an edge), otherwise $\boldsymbol{A}(\mathcal{G})_{i,j} = 0$. The feature of node $i$ is represented by $\boldsymbol{V}(\mathcal{G})_{i,:}$, and the feature of edge $(i, j)$ is represented by $\boldsymbol{E}(\mathcal{G})_{i,j,1:d_e}$. The permutation (or reindexing) of $\mathcal{G}$ is denoted as $\mathcal{G}^\pi = (\mathbf{V}(\mathcal{G}), \mathbf{E}(\mathcal{G}))$ with permutation $\pi : [n] \to [n]$, such that $\boldsymbol{V}(\mathcal{G})_{i,:} = \boldsymbol{V}(\mathcal{G})_{\pi(i),:}$ and $\boldsymbol{E}(\mathcal{G})_{i,j,:} = \boldsymbol{E}(\mathcal{G})_{\pi(i),\pi(j),:}$.

Next, we explore the utilization of features. It is evident that incorporating node features during initialization and edge features during message passing can enhance the performance of GNNs, given appropriate hyperparameters and training. However, we should consider whether features can truly represent graph structures or provide additional expressiveness. Let us categorize features into two types.

The first type involves fully utilizing the original features, such as distances to other nodes or spectral embeddings. While using these features can aid GNNs in solving Graph Isomorphism (GI) problems, this type of feature requires a dedicated design to effectively utilize them. For instance, if we aim to recognize a 6-cycle in a graph, we can manually identify the cycle and assign distinct features to each node within the cycle. In this way, the GNN can recognize the cycle by aggregating the six distinctive features. However, the injecting strategy influences expressiveness and requires further analysis. Utilizing distance can also enhance expressiveness but also need a suitable design (like subgraph distance encoding and SPD-WL).

The second type entails incorporating additional features, such as manually selected node identifiers. it is important to note that this improvement stems from reduced difficulty rather than increased expressiveness. For instance, given a pair of non-isomorphic graphs with high similarity, we can manually find the components causing the distinguishing difficulty and assign identifiers to help models overcome them. However, this process is generally unavailable in practice.

In summary, we can conclude that features have the potential to introduce expressiveness, but this should be accomplished through model design rather than relying solely on the dataset. In the case of BREC, a dataset created specifically for testing expressiveness, we do not include additional meaningful features. Instead, we employ the same vector for all node features and edge features and adhere to specific model settings to incorporate graph-specific features, such as the distance between nodes in distance encoding based models.

## C. WL Algorithm

This section briefly introduces the WL algorithm and two high-order variants.

The 1-WL algorithm, short for "1-Weisfeiler-Lehman," is an initial version of the WL algorithm. It serves as a graph isomorphism algorithm and can be employed to generate a distinctive label for each graph.

In the 1-WL algorithm, every node in the graph maintains a state or color, which undergoes refinement during each iteration by incorporating information from the states of its neighboring nodes. As the algorithm progresses, the graph representation evolves into a multiset of node states, ultimately converging to a final representation.

To circumvent these examples, researchers have devised a technique to augment each node in the 1-WL test, resulting in the development of the $k$-WL test (Babai & Kucera, 1979). The $k$-dimensional Weisfeiler-Lehman test expands the scope of the test to consider colorings of k-tuples of nodes instead of individual nodes. This extension allows for a more comprehensive analysis of graph structures and assists in overcoming the limitations posed by certain examples.

In addition to the $k$-WL test, Cai et al. (1989) proposed an alternative WL test algorithm that also extends to $k$-tuples. This variant is commonly referred to as the $k$-FWL ($k$-folklore-WL) test. The $k$-FWL test differs from the $k$-WL test in terms of how neighbors are defined and the order in which aggregation is performed on tuples and multisets.

There are three notable results associated with these tests:

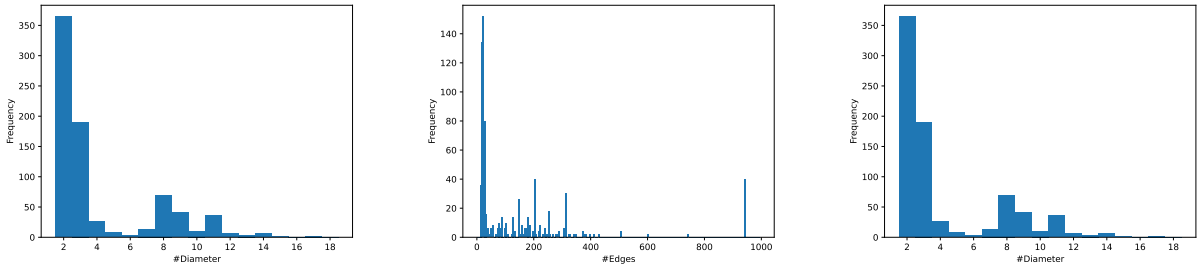
1. 1-WL = 2-WL
2. $k$-WL $>$ $(k-1)$-WL, $(k > 2)$
3. $(k-1)$-FWL = $k$-WL

More details can be found in Sato (2020); Huang & Villar (2021).

## D. Circulant Skip Links (CSL) Graphs

A CSL graph is defined as follows: Let $r$ and $m$ be co-prime natural numbers with $r < m - 1$. $\mathcal{G}(m, r) = (\mathbb{V}, \mathbb{E})$ is an undirected 4-regular graph with $\mathbb{V} = [m]$, where the edges form a cycle and include skip links. Specifically, for the cycle, $(j, j+1) \in \mathbb{E}$ for $j \in [m-1]$, and $(m, 1) \in \mathbb{E}$. For the skip links, the sequence is recursively defined as $s_1 = 1$, $s_{i+1} = (s_i + r) \bmod m + 1$, and $(s_i, s_{i+1}) \in \mathbb{E}$ for any $i \in \mathbb{N}$. Two Sample graphs with $m = 10, r = 2\ or\ 3$ are shown in Fig 1(b).

In the CSL dataset, 10 CSL graphs with $m = 41$ and $r = 2, 3, 4, 5, 6, 9, 11, 12, 13, 16$ are generated. Thus resulting in 10 non-isomorphic but 1-WL-indistinguishable graphs.




(a) #Nodes Distribution (b) #Edges Distribution (c) Diameter Distribution

*Figure 5.* BREC Statistics

## E. GNN Extensions

In this section, we revisit definitions of the four GNN extensions as proposed by Papp & Wattenhofer (2022).

**$k$-WL hierarchy ($k$-WL)**. The same as $k$-dimensional Weisfeiler-Leman algorithm.

**Substructure-counting ($S_k$)**. Let us define an $S_k$ GNN as follows: we assume that there is a preprocessing phase where for each $k' \in \{1, 2, ..., k\}$, we consider every different connected graph $\mathcal{G}$ on $k'$ nodes (up to isomorphism), and we count the number of times this graph $\mathcal{G}'$ appears as an induced subgraph such that $u$ is one of the nodes of $\mathcal{G}'$. We add these numbers as new features to each node $u$ in the graph, and then we run a standard GNN on the graph with these extended features.

**$k$-hop-subgraph ($N_k$)**. We assume that there is a mapping from all possible induced $k$-hop neighborhoods (that is, all graphs of radius at most $k$ up to isomorphism) to the real numbers, and each node $u$ is equipped with this number as an extra feature in a preprocessing step. This is then followed by regular message passing (i.e. a standard GNN) for enough rounds.

**node-marking ($M_k$)**. We define $M_k$ as the GNN which combines a standard GNN run over all distinct $k'$-markings of $u$, for every $k' \in \{0, 1, ..., k\}$. That is, $M_k$ considers every version of the $d$-hop neighborhood around $u$ obtained by marking at most $k$ distinct nodes, computes an embedding for $u$ with the same GNN in each case, and then combines these into a final embedding for $u$.

Extension graphs are generated by comparing various GNN extensions with specific examples. For a detailed description of the generation process, please refer to Appendix P.

## F. BREC Statistics

Here we give some statistics of the BREC dataset, shown in Figure 5.

## G. Ablation Study on Training and Random Weight

In this section, we conducted an ablation study to investigate the necessity of training. By examining the performance degradation resulting from the removal of the training phase, we confirmed the essential nature of the training process. The results are presented in Table 3.

## H. Abliation Study on RPC

To evaluate the necessity of RPC, we conducted an ablation study to compare it with a simple threshold strategy. The datasets, training parameters, and sample sizes remained constant; the only variable was the evaluation method. We calculated the mean embeddings for two graph sets requiring differentiation and used the L2 norm of their differences as the distance metric. We then compared the outcomes using various threshold levels, with the results detailed in Table 4.

The configuration 'Different_1E-4' indicates that two distinct sets of graphs are differentiated with a threshold of 1E-4; the number associated with this configuration reflects the count of successfully distinguished graph pairs. Conversely, 'Same_1E-4' applies to graph sets from identical graphs with the same threshold. Here, a result of 0 would imply complete

*Table 3.* Ablation study on training

| Model | Random Performance | Training Performance | Performance Degradation Compared to Original Results |
|---|---|---|---|
| NGNN | 166 | 166 | 0 % |
| DE+NGNN | 226 | 231 | 2.2 % |
| DS-GNN | 80 | 222 | 64.0 % |
| DSS-GNN | 142 | 221 | 35.7 % |
| SUN | 216 | 223 | 3.1 % |
| SSWL_P | 224 | 248 | 9.7 % |
| GNN-AK | 219 | 222 | 1.4 % |
| KP-GNN | 253 | 275 | 8.0 % |
| $I^2$-GNN | 267 | 281 | 5.0 % |
| PPGN | 212 | 233 | 9.0 % |
| $\delta$-k-LGNN | 58 | 216 | 73.1 % |
| KC-SetGNN | 207 | 211 | 1.9 % |
| GSN | 254 | 254 | 0 % |
| DropGNN | 8 | 177 | 95.5 % |
| OSAN | 124 | 148 | 16.2 % |
| Graphormer | 79 | 79 | 0 % |

*Table 4.* Ablation study on RPC

| Model | Different_1E-4 | Same_1E-4 | Different_1E-1 | Same_1E-1 | Each_Pair_Threshold | Largest_Threshold | RPC |
|---|---|---|---|---|---|---|---|
| NGNN | 331 | 160 | 191 | 25 | 290 | 161 | 166 |
| DE+NGNN | 400 | 214 | 297 | 75 | 322 | 221 | 231 |
| DS-GNN | 241 | 88 | 223 | 19 | 260 | 201 | 222 |
| DSS-GNN | 247 | 168 | 186 | 41 | 261 | 15 | 221 |
| SUN | 263 | 221 | 199 | 48 | 255 | 44 | 223 |
| SSWL_P | 255 | 109 | 223 | 0 | 264 | 229 | 248 |
| GNN-AK | 234 | 18 | 128 | 0 | 244 | 217 | 222 |
| KP-GNN | 304 | 136 | 260 | 9 | 295 | 206 | 275 |
| $I^2$-GNN | 336 | 123 | 261 | 0 | 304 | 281 | 281 |
| PPGN | 392 | 189 | 339 | 107 | 315 | 0 | 233 |
| $\delta$-k-LGNN | 209 | 0 | 131 | 0 | 245 | 211 | 216 |
| KC-SetGNN | 270 | 237 | 211 | 0 | 234 | 211 | 211 |
| GSN | 260 | 4 | 254 | 0 | 259 | 254 | 254 |
| DropGNN | 400 | 400 | 400 | 400 | 327 | 191 | 177 |
| OSAN | 0 | 0 | 0 | 0 | 236 | 1 | 148 |
| Graphormer | 0 | 0 | 0 | 0 | 167 | 68 | 79 |

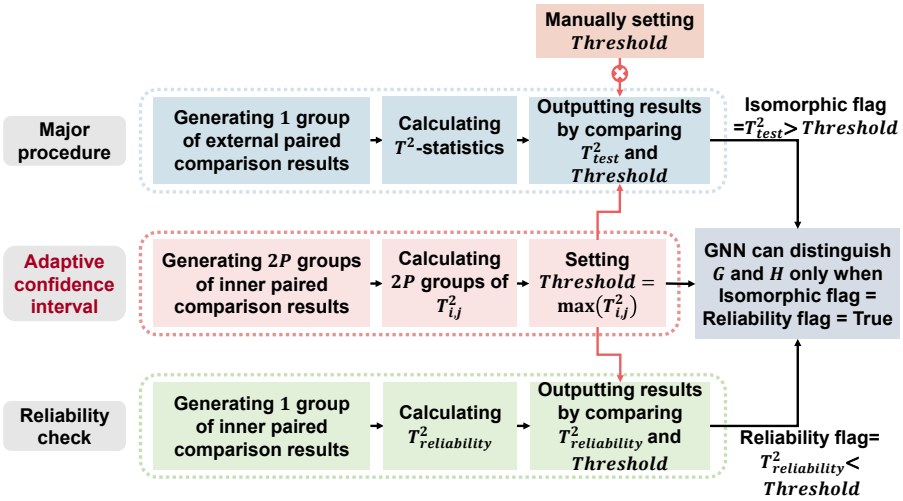



*Figure 6.* RAPC pipeline.

accuracy, as no false positives—incorrectly distinguished pairs—are expected. Under the 'Largest_Threshold' setting, the threshold is determined by the greatest distance among identical graphs and then applied to different graphs to secure optimal and trustworthy outcomes. Conversely, in the 'Each_Pair_Threshold' configuration, a unique threshold for each graph pair is set based on the distance among identical graphs, but this method is still considered undependable. For indistinguishable graphs, the distances between identical sets and different sets are thought to be from the same statistical distribution. Hence, there is a 50% chance that the distance between identical sets could be greater than that between different sets, potentially resulting in erroneous identification. The 'RPC' describes the outcomes when using RPC for evaluations with the same results as in main Table 2.

The study revealed that a small fixed threshold often leads to unreliable results, as shown by the misclassified samples in 'Same_1E-4'. The issue persists with larger thresholds and is further compounded by performance decline. Using a specific threshold for each graph pair ('Each_Pair_Threshold') is also unreliable. Certain methods, such as DE+NGNN and PPGN, have at times failed to outperform 3-WL. Nevertheless, these methods have demonstrated superior performance to 3-WL, which reliably discriminates between 270 graph pairs. While the 'Largest_Threshold' method appears reliable, we noted a decline in performance, especially for some models.

In conclusion, our RPC illustrates a clear advantage in providing robust and dependable results with a user-friendly and widely applicable approach, underscoring its importance.

## I. RAPC: a Reliable and Adaptive Evaluation Method

In this section, we propose RAPC with an additional stage called adaptive confidence interval based on RPC. Though RPC performs excellently in experiments with a general theoretical guarantee in reliability, with manually setting $\alpha$. We still want to make the procedure more automated. In addition, we found that the inner fluctuations of each pair, i.e. $T^2_{\text{reliability}}$, vary from pairs. This means some graph outputs are more stable than others, and their threshold can be larger than others. However, it is impossible to manually set the confidence interval ($\alpha$) for all pairs, thus, we propose an adaptive confidence interval method to solve this problem. The key idea is to set the threshold according to minimum internal fluctuations.

Given a pair of non-isomorphic graphs $\mathcal{G}$ and $\mathcal{H}$ to be tested. For simplicity, we rename $\mathcal{G}$ as $\mathcal{G}_1$, $\mathcal{H}$ as $\mathcal{G}_2$. For each graph ($\mathcal{G}_1$ and $\mathcal{G}_2$), we generate $p$ groups of graphs, with each group containing $2q$ graphs, represented by:

$$\mathcal{G}_{i,j,k},\ i \in [2],\ j \in [p],\ k \in [2q]. \tag{11}$$

Similarly, we can calculate $T^2$-statistics for each group ($2p$ groups in total):

$$T^2_{i,j} = q\overline{\boldsymbol{d}}^T_{i,j}\boldsymbol{S}_{i,j}\overline{\boldsymbol{d}}_{i,j},\ i \in [2],\ j \in [p]. \tag{12}$$

where

$$\overline{\boldsymbol{d}}_{i,j} = \frac{1}{q}\sum_{k=1}^{q} \boldsymbol{d}_{i,j,k},\ \boldsymbol{d}_{i,j,k} = f(\mathcal{G}_{i,j,k}) - f(\mathcal{G}_{i,j,k+q}),\ i \in [2],\ j \in [p],\ k \in [q],$$
$$\boldsymbol{S}_{i,j} = \frac{1}{q-1}\sum_{j=1}^{q}(\boldsymbol{d}_{i,j,k} - \overline{\boldsymbol{d}}_{i,j})(\boldsymbol{d}_{i,j,k} - \overline{\boldsymbol{d}}_{i,j})^T. \tag{13}$$

Similar to major procedure, we can conduct an $\alpha$-level test of $H_0 : \delta = \mathbf{0}$ versus $H_1 : \delta \neq \mathbf{0}$, it should always accept $H_0$(the GNN cannot distinguish them) since the $2q$ graphs in each group are essentially the same. And $T^2$-statistics should satisfy the:

$$T^2_{i,j} = q\overline{\boldsymbol{d}}^T_{i,j}\boldsymbol{S}_{i,j}\overline{\boldsymbol{d}}_{i,j} < \frac{(q-1)n}{(q-n)}F_{n,q-n}(\alpha). \tag{14}$$

If the GNN can distinguish the pair, $T^2_{\text{test}}$ in major procedure and $T^2_{i,j}$ in adaptive confidence interval should satisfy the:

$$T^2_{\text{test}} > \frac{(q-1)n}{(q-n)}F_{n,q-n}(\alpha) > T^2_{i,j}, \forall i \in [2],\ j \in [p]. \tag{15}$$

Thus we set the adaptive confidence interval as Threshold $=$ Max$_{i\in\{1,2\},\ p\in\{1,\ldots,P\}}\{T^2_{i,p}\}$. Then we conduct Major Procedure and Reliability Check based on Threshold similar to RPC. The pipeline is shown in Fig 6.

In our analysis of the current evaluation method, we take into account the probabilities of false positives and false negatives. Typically, achieving extremely low levels of both probabilities simultaneously is challenging, and there is often a trade-off between them. However, since false positives can undermine the reliability of the methods, we prioritize establishing stringent bounds for this type of error. On the other hand, false negatives are explained in a more intuitive manner, acknowledging their presence but placing greater emphasis on minimizing false positives.

Regarding false positives, we give the following theorem.

**Theorem I.1.** *The false positive rate with adaptive confidence interval is* $\frac{1}{2^{2P}}$.

*Proof.* We first define false positives more formally. False positives mean the GNN $f$ cannot distinguish $\mathcal{G}$ and $\mathcal{H}$, but we reject $H_0$ and accept $H_1$. $f$ cannot distinguish $\mathcal{G}$ and $\mathcal{H}$ means $f(\mathcal{G}) = f(\mathcal{H}) = f(\mathcal{G}^\pi) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathcal{G}}, \boldsymbol{\Sigma}_{\mathcal{G}})$. Since $\boldsymbol{d}_i$ in major procedure and $\boldsymbol{d}_{i,j,k}$ in adaptive confidence interval are derived from paired comparison by same function outputs, i.e., from $f(\mathcal{G})$ and $f(\mathcal{H})$, and from $f(\mathcal{G})$ and $f(\mathcal{G}^\pi)$, respectively. $\boldsymbol{d}_i$ and $\boldsymbol{d}_{i,j,k}$ should follow the same distribution, leading that $T^2_{\text{test}}$ and $T^2_{i,j}$ are independently random variables following the same distribution. Thus $P(T^2_{\text{test}} > T^2_{i,j}) = \frac{1}{2}$. Then we can calculate the probability of false positives as

$$P(\textit{Rejecting } H_0) = P(T^2_{\text{test}} > \textit{Threshold} = \textit{Max}_{i\in[2],\ j\in[p]}\{T^2_{i,j}\}) = \frac{1}{2^{2p}}. \tag{16}$$

Thus we proof theorem I.1. □

Regarding false negatives, we propose the following explanation. A small threshold can decrease the false negative rate. Thus without compromising the rest of the theoretical analysis, we give the minimum value of the threshold. Equation 14 introduces a minimum threshold restriction. We obtain the threshold strictly based on it by taking the maximum value, which is the theoretical minimum threshold that minimizes the false negative rate.

## J. Subgraph GNNs

In this section, we discuss settings for subgraph GNN models. The most important setting is the subgraph radius. As discussed before, a larger radius can capture more structural information, increasing the model's expressiveness. However, it will include more invalid information, making reaching the theoretical upper bound harder. Thus we need to find a balance between the two.

To achieve this, we first explore the maximum structural information that can be obtained under a given radius. Following Papp & Wattenhofer (2022), we implement $N_k$ method, which embeds the isomorphic type of $k$-hop subgraph when

*Table 5.* A general theoretical expressiveness upper bound of subgraph with radius $k$

| Radius | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| #Accurate on BREC | 252 | 298 | 300 | 327 | 326 | 385 | 398 | 398 | 399 | 400 |

*Table 6.* The performance of 3-WL with different iteration times

| Iterations | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| #Accurate on BREC | 193 | 209 | 217 | 264 | 270 |

initializing. This method is only available in the theoretical analysis as one can not solve the GI problem by manually giving graph isomorphic type. We mainly use it as a general expressiveness upper bound of subgraph GNNs. The performance of $N_k$ on BREC is shown in Table 5. Actually, $N_3$ already successfully distinguishes all graphs except for CFI graphs. $k = 6$ is an important threshold as $N_k$ outperforms 3-WL (expressiveness upper bound for most subgraph GNNs (Frasca et al., 2022; Zhang et al., 2023a)) in all types of graphs. An interesting discovery is that increasing the radius does not always lead to expressiveness increasing as expected. This is caused by the fact that we only encode the exact $k$-hop subgraph instead of 1 to $k$-hop subgraphs. This phenomenon is similar to subgraph GNNs, revealing the advantages of using distance encoding.

We then test the subgraph GNNs' radii by increasing them until reaching the best performance, which is expected to be a perfect balance. For some methods, radius$= 6$ is the best selection, which is consistent with the theory. The exceptions are NGNN, NGNN+DE, KPGNN, $I^2$-GNN and SSWL_P. NGNN directly uses an inner GNN to calculate subgraph representation, whose expressiveness is restricted by the inner GNN. As the subgraph radius increases, though the subgraph contains information, the simple inner GNN can hardly give a correct representation. That's why radius$= 1$ is the best setting for NGNN. NGNN+DE and $I^2$-GNN add distance encodings, making the subgraph with a large radius can always clearly extract a subgraph with a small radius. Therefore, a large radius$= 8$ is available. KPGNN utilizes a similar setting by incorporating distance to subgraph representation, and radius$= 8$ is also the best setting. KPGNN can also use graph diffusion to replace the shortest path distance. Though graph diffusion outperforms some graphs, the shortest path distance is generally a better solution. Previous findings reveal the advantages of using distance, which we hope can be more widely used in further research. SSWL_P achieves better expressiveness with theoretical minimum components, making more information available.

## K. $k$-WL Hierarchy GNNs

In this section, we discuss settings for $k$-WL hierarchy GNN models. $k$-WL algorithm requires a converged tuple embedding distribution for GI. However, $k$-WL hierarchy GNNs do not have the definition of converging. It will output the final embeddings after a specific number of layers, i.e., the iteration times of $k$-WL. Thus we need to give a suitable number of layers where the $k$-WL converged after the number of iteration times. In theory, increasing the number of layers always leads to a non-decreasing expressiveness, since the converged distribution will not change furthermore. However, more layers may cause over-smoothing, leading to worse performance in practice.

To keep a balance, we utilize similar methods for subgraph GNNs. We first analyze the iteration times of 3-WL, shown in Table 6. One can see 6 iteration times are enough for all types of graphs. Then we increase the layers of $k$-WL GNNs until reaching the best performance. We finally set 5 layers for PPGN, 4 layers for KCSet-GNN and 6 layers for $\delta$-k-LGNN.

## L. Substructure-based GNNs

In this section, we discuss the performance of substructure-based GNN models. Specifically, we focus on the GSN (Graph Substructure Network) model proposed by Bouritsas et al. (2022), which offers a straightforward neural network implementation, denoted as GSN-v, of the $S_k$ substructure. Additionally, we introduce GSN-e, a slightly stronger version of GSN-v that incorporates features on edges instead of just nodes.

Experimental results presented in Table 7 demonstrate that GSN-v achieves a perfect match with the performance of $S_k$. Furthermore, GSN-e outperforms GSN-v, indicating superior performance when edge features are included.

*Table 7.* Substructure-based model performance on BREC

| | Basic Graphs (60) | | Regular Graphs (140) | | Extension Graphs (100) | | CFI Graphs (100) | | Total (400) | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | Number | Accuracy | Number | Accuracy | Number | Accuracy | Number | Accuracy | Number | Accuracy |
| $S_3$ | 52 | 86.7% | 48 | 34.3% | 5 | 5% | 0 | 0% | 105 | 26.2% |
| $S_4$ | 60 | 100% | 99 | 70.7% | 84 | 84% | 0 | 0% | 243 | 60.8% |
| GSN-v(k=3) | 52 | 86.7% | 48 | 34.3% | 5 | 5% | 0 | 0% | 105 | 26.2% |
| GSN-v(k=4) | 60 | 100% | 99 | 70.7% | 84 | 84% | 0 | 0% | 243 | 60.8% |
| GSN-e(k=3) | 59 | 98.3% | 48 | 34.3% | 52 | 52% | 0 | 0% | 159 | 39.8% |
| GSN | 60 | 100% | 99 | 70.7% | 95 | 95% | 0 | 0% | 254 | 63.5% |

*Table 8.* The performance of DropGNN with different sample numbers

| #Samples | 100 | 200 | 400 | 800 | 1200 | 1600 |
|---|---|---|---|---|---|---|
| #Accurate on BREC | 177 | 222 | 242 | 253 | 260 | OOM |

## M. Random GNNs

In this section, we delve into the settings for random GNNs. Random GNNs leverage samples from graphs using specific strategies, and both the number of samples and the sampling strategies have an impact on performance.

For DropGNN, the sampling strategy revolves around a relatively straightforward approach of deleting nodes. As for the number of samples, it is recommended to set it to the average number of nodes in the dataset. In our reported results, we set the number of samples to 100, which aligns with the average number of nodes. The ablation study results on the number of samples can be found in Table 8.

Another approach, OSAN, proposes a data-driven method that achieves similar performance with fewer samples. This is achieved by training the model to select diverse samples. However, it requires an additional training framework and may not necessarily lead to improved performance. In our case, we select the edge-deleting strategy and set the number of samples to 20.

## N. Tight Bound Expressiveness Measurement

In this section, we aim to clarify two concepts: "expressiveness upperbound" and "expressiveness tight bound." It is important to differentiate between these two concepts, with the former typically referring to a model whose expressiveness upperbound is aligned with a specific hierarchy (e.g., 3-WL), and the latter indicating the precise expressiveness achievable through an implementation. While previous research has predominantly focused on the "expressiveness upperbound" due to its connection to other frameworks and ease of analysis and comparison, the concept of "expressiveness tight bound" has received less attention as it is often considered in isolation from existing work.

Nevertheless, a tight bound can offer significant benefits within our benchmark, serving as a universal framework for evaluating and comparing all results. To address this gap, we have collected the exact theoretical approximations of various models and initially implemented them to improve the understanding of expressiveness, as shown in Table 9.

For SWL, GSWL, and SSWL, we recommend referencing the paper by Zhang et al. (2023a). When referring to "unlimited," we are discussing OSAN utilizing a random edge deletion strategy, which theoretically offers unlimited expressiveness but is challenging to achieve in practical applications. In the context of $S_4$ on the edge, we are describing the process of incorporating substructure information into edges and utilizing the 1-WL algorithm. As for NGNN-WL, KPGNN-WL, $I^2$-GNN-WL, LWL-Plus, and KC-SetWL, we are referring to enhancing the original models by replacing model layers with appropriate multiset and hash functions

*Table 9.* Tight bound expressiveness

| Model | Expressiveness Tight Bound | Theoretical Performance | Practical Performance |
|---|---|---|---|
| NGNN | NGNN-WL with $k = 1$ | 167 | 166 |
| DE+NGNN | DENGNN-WL with $k = 8$ | 247 | 231 |
| DS-GNN | SWL(VS) with $k = 6$ | 243 | 222 |
| DSS-GNN | GSWL(VS) with $k = 6$ | 232 | 221 |
| SUN | GSWL(VS) with $k = 6$ | 232 | 223 |
| SSWL_P | SSWL(VS) with $k = 8$ | 262 | 258 |
| GNN-AK | GSWL(VS) with $k = 6$ | 232 | 222 |
| KP-GNN | KPGNN-WL with $k$ =8 | 291 | 275 |
| $I^2$-GNN | $I^2$-GNN-WL | 301 | 281 |
| PPGN | 3-WL | 270 | 233 |
| $\delta$-k-LGNN | LWL-Plus | 267 | 216 |
| KC-SetGNN | KC-SetWL | 213 | 211 |
| GSN | $S_4$ on edge | 254 | 254 |
| DropGNN | $M_1$ | 251 | 177 |
| OSAN | Unlimited | 400 | 148 |
| Graphormer | SPD-WL | 83 | 79 |

*Table 10.* Model hyperparameters

| Model | Radius | Layers | Inner dim | Learning rate | Weight decay | Batch size | Epoch | Early stop threshold |
|---|---|---|---|---|---|---|---|---|
| NGNN | 1 | 6 | 16 | $1e-4$ | $1e-5$ | 32 | 20 | 0.01 |
| DE+NGNN | 8 | 6 | 128 | $1e-4$ | $1e-5$ | 32 | 30 | 0.01 |
| DS-GNN | 6 | 10 | 32 | $1e-4$ | $1e-5$ | 32 | 30 | 0 |
| DSS-GNN | 6 | 9 | 32 | $1e-4$ | $1e-4$ | 32 | 20 | 0.01 |
| SUN | 6 | 9 | 32 | $1e-4$ | $1e-4$ | 32 | 20 | 0.01 |
| SSWL_P | 8 | 8 | 64 | $1e-5$ | $1e-5$ | 8 | 20 | 0.1 |
| GNN-AK | 6 | 4 | 32 | $1e-4$ | $1e-4$ | 32 | 10 | 0.1 |
| KP-GNN | 8 | 8 | 32 | $1e-4$ | $1e-4$ | 32 | 20 | 0.3 |
| $I^2$GNN | 8 | 5 | 32 | $1e-5$ | $1e-4$ | 16 | 20 | 0.2 |
| PPGN | / | 5 | 32 | $1e-4$ | $1e-4$ | 32 | 20 | 0.2 |
| $\delta$-k-LGNN | / | 6 | 16 | $1e-4$ | $1e-4$ | 16 | 20 | 0.2 |
| KC-SetGNN | / | 4 | 64 | $1e-4$ | $1e-4$ | 16 | 15 | 0.3 |
| GSN | / | 4 | 64 | $1e-4$ | $1e-5$ | 16 | 20 | 0.1 |
| DropGNN | / | 10 | 16 | $1e-3$ | $1e-5$ | 16 | 100 | 0 |
| OSAN | / | 8 | 64 | $1e-3$ | $1e-5$ | 16 | 40 | 0 |
| Graphormer | / | 12 | 80 | $2e-5$ | 0 | 16 | 100 | 0 |

## O. Experiment Settings

All experiments were performed on a machine equipped with an Intel Core i9-10980XE CPU, an NVIDIA RTX4090 graphics card, and 256GB of RAM.

**RPC settings.** For non-GNN methods, the output results are uniquely determined, and as such, this part of the experiment does not require RPC. It is worth noting that most non-GNN baselines involve running graph isomorphism testing software on subgraphs, and they mainly serve as theoretical references in our evaluation.

Regarding GNNs, we employ RPC with $q = 32$ and $d = 16$ to evaluate their performance. Considering a confidence level of $\alpha = 0.95$, which is a typical setting in statistics, the threshold should be set to $\frac{(q-1)d}{(q-d)} F_{d,q-d}(\alpha) = 31 F_{16,16}(0.95) = 72.34$.

To ensure robustness, we repeat all evaluation methods ten times using different seeds selected from the set $\{100, 200, \ldots, 1000\}$. We consider the final results reliable only if the model passes the Reliability Check for all graphs with any seed, meaning that the quantification of the output embedding distance between isomorphic pairs is always smaller than the threshold. The reported results are selected as the best results rather than the average, as we aim to explore the upper bound of expressiveness.

**Training settings.** We employ a Siamese network design and utilize the cosine similarity loss function. Another commonly used loss function is contrastive loss (Hadsell et al., 2006), which directly calculates the difference between two outputs. However, we opt for cosine similarity loss due to its advantage of measuring output difference under the same scale through normalization. This approach prevents model outputs from being excessively amplified, which could otherwise magnify minor precision errors and treat them as differentiated results of the model.

We use the Adam optimizer with a learning rate searched from $\{1e-3, 1e-4, 1e-5\}$, weight decay selected from $\{1e-3, 1e-4, 1e-5\}$, and batch size chosen from $\{8, 16, 32\}$. Graphormer, on the other hand, follows the original training settings on ZINC.

We incorporate an early stopping strategy, which halts training when the loss reaches a small value. While for random GNNs, we do not utilize early stopping. The maximum number of epochs is typically set to around 20 since the model can often distinguish a pair relatively quickly.

**Model hyperparameters.** The most crucial hyperparameters related to expressiveness, such as the subgraph radius for subgraph GNNs and the number of layers for $k$-WL hierarchy GNNs, are determined through theoretical analysis, as outlined in Appendix J and K. These hyperparameters have a direct impact on the expressiveness of the models.

Other hyperparameters also implicitly influence expressiveness. We generally adopt the same settings as previous expressiveness datasets, with two exceptions: inner embedding dimension and batch normalization.

The inner embedding dimension reflects the model's capacity. For smaller and simpler expressiveness datasets used in the past, a small embedding dimension has been sufficient. However, the appropriate embedding dimension for BREC is unknown, so we generally conduct a search within the range of $16, 32, 64, 128$.

Additionally, we utilize batch normalization for all models, even though it may not have been used in all previous models. Batch normalization helps control the outputs within a suitable range, which can be beneficial for distinguishing graph pairs.

The detailed hyperparameter settings for each method are provided in Table 10.

**Model Costs.** We present the time consumption and model parameter size in Table 11 to offer additional information that helps users to better utilize or optimize the models. Time consumption is divided into two parts: preprocessing time and evaluation time. Preprocessing time refers to the time taken to prepare the dataset, including tasks like computing substructures or preprocessing subgraphs. Usually, preprocessing occurs once during hyperparameter tuning, except in cases such as when adjustments to the subgraph radius are made. Evaluation time denotes the time required to assess each particular combination of hyperparameters.

To address potential memory constraints or prolonged processing times encountered by some methods when handling complex graphs, we have excluded 4-vertex condition graphs and 40 pairs of 3-WL-indistinguishable CFI graphs from their evaluation. The time consumption for these methods is indicated as '$> K$', showing that processing time exceeds a certain threshold K. The symbol '$>> K$' signifies that only those graphs distinguishable by the 3-WL test have been retained. Consequently, strongly regular graphs, 4-vertex condition graphs, distance-regular graphs, and 40 pairs of 3-WL-indistinguishable CFI graphs have been removed. By implementing this strategy, we guarantee that the performance of the

*Table 11.* Model costs

| Model | Preprocess Time(s) | Evaluation Time(s) | Total Time(s) | Parameter Size (KB) |
|---|---|---|---|---|
| NGNN | 516 | 388 | 904 | 14 |
| DE+NGNN | 3123 | 1087 | 4200 | 787 |
| DS-GNN | 2343 | 3300 | 5643 | 173 |
| DSS-GNN | 2343 | 1008 | 3351 | 162 |
| SUN | 2343 | 1925 | 4268 | 748 |
| SSWL_P | >>644 | >>4085 | >>4729 | 924 |
| GNN-AK | 768 | 1048 | 1816 | 4742 |
| KP-GNN | 73113 | 873 | 73986 | 366 |
| $I^2$GNN | >10460 | >7013 | >17473 | 236 |
| PPGN | 47 | 156 | 477 | 90 |
| $\delta$-k-LGNN | 5 | 2532 | 2537 | 62 |
| KC-SetGNN | >>118005 | >>5670 | >>123675 | 1416 |
| GSN | >5231 | >150 | >5381 | 431 |
| DropGNN | 393 | 533 | 926 | 33 |
| OSAN | <1 | 187023 | 187023 | 749 |
| Graphormer | >>66 | >>13681 | >>13747 | 3667 |

models under evaluation is not overestimated, as they fall behind the discriminative power of the 3-WL test.

## P. Graph Generation

In this section, we provide an overview of how the graphs in the BREC dataset were generated.

**Basic graphs.** This category consists of 60 pairs of graphs, each containing 10 nodes. To generate these graphs, the 1-WL algorithm was applied to all 11.7 million graphs with 10 nodes, resulting in a hash value for each graph. Among these graphs, 83,074 happened to have identical hash values as others. From this set, 60 pairs of graphs were randomly selected.

**Regular graphs.** This category includes 140 pairs of regular graphs. For the 50 simple regular graphs, the search was conducted for regular graphs with 6 to 10 nodes, and 50 pairs of regular graphs with the same parameters were randomly selected. For the 50 strongly regular graphs, the number of nodes ranged from 16 to 35. The graphs were obtained from sources such as http://www.maths.gla.ac.uk/ es/srgraphs.php and http://users.cecs.anu.edu.au/ bdm/data/graphs.html. For the 20 4-vertex condition graphs, a search was conducted on http://math.ihringer.org/srgs.php, and the simplest 20 pairs of 4-vertex condition graphs with the same parameters were selected. For the 20 distance regular graphs, a search was performed on https://www.distanceregular.org/, and the simplest 20 pairs of distance regular graphs with the same parameters were chosen.

**Extension graphs.** This category consists of 100 pairs of graphs based on comparing results between GNN extensions. The $S_3$, $S_4$, and $N_1$ algorithms were applied to all 1-WL-indistinguishable graphs with 10 nodes. This yielded 4,612 $S_3$-indistinguishable graphs, 1,132 $N_1$-indistinguishable graphs, and 136 $S_4$-indistinguishable graphs. From these sets, 60 pairs of $S_3$-indistinguishable graphs, 20 pairs of $N_1$-indistinguishable graphs, and 10 pairs of $S_4$-indistinguishable graphs were randomly selected. Care was taken to ensure that no graphs were repeated. Additionally, 10 pairs of graphs were added using a virtual node strategy, including 5 pairs obtained by adding a virtual node to a 10-node regular graph and 5 pairs based on $C_{2l}$ and $C_{l,l}$ as described in Papp & Wattenhofer (2022).

**CFI graphs.** This category consists of 100 pairs of graphs generated based on the CFI methods proposed by Cai et al. (1989). All CFI graphs with backbones ranging from 3 to 7-node graphs were generated. From this set, 60 pairs of 1-WL-indistinguishable graphs, 20 pairs of 3-WL-indistinguishable graphs, and 20 pairs of 4-WL-indistinguishable graphs were randomly selected.

These different categories of graphs provide a diverse range of graph structures and properties for evaluating the expressiveness of GNN models.