ON SINGLE-ENVIRONMENT EXTRAPOLATIONS IN GRAPH CLASSIFICATION AND REGRESSION TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Extrapolation in graph classification/regression remains an underexplored area of an otherwise rapidly developing field. Our work contributes to a growing literature by providing the first systematic counterfactual modeling framework for extrapolations in graph classification/regression tasks. To show that extrapolation from a single training environment is possible, we develop a connection between certain extrapolation tasks on graph sizes and Lovász's characterization of graph limits. For these extrapolations, standard graph neural networks (GNNs) will fail, while classifiers using induced homomorphism densities succeed, but mostly on unattributed graphs. Generalizing these density features through a GNN subgraph decomposition allows them to also succeed in more complex attributed graph extrapolation tasks. Finally, our experiments validate our theoretical results and showcase some shortcomings of common (interpolation) methods in the literature.

1 INTRODUCTION

In some graph classification and regression applications, the graphs themselves are *representations of a natural process rather than the true state of the process*. Molecular graphs are built from a pairwise atom distance matrix by keeping edges whose distance is below a certain threshold and the choice impacts distinguishability between molecules (Klicpera et al., 2020). Functional brain connectomes are derived from time series but researchers must choose a frequency range for the signals, which affects resulting graph structure (De Domenico et al., 2016). Recent work (e.g. Knyazev et al. (2019); Bouritsas et al. (2020); Xu et al. (2020)) explore extrapolations in real-world tasks, showcasing a growing interest in the underexplored topic of graph extrapolation tasks.

In this work we refer to *graph-processing environment* (or just *environment*) as the collection of heuristics and other data curation processes that gave us the observed graph from the *true state* of the process under consideration. The *true state* alone defines the target variable. Our work is interested in what we refer as the *graph extrapolation task*: predict a target variable from a graph regardless of its environment. In this context, even graph sizes can be determined by the environment. Unsurprisingly, graph extrapolation tasks —a type of out-of-distribution prediction— are only feasible when we make assumptions about these environments.

We define the *graph extrapolation task* as a counterfactual inference task that requires learning environment-invariant (E-invariant) representations. Unfortunately, *graph datasets largely contain a single environment*, while common E-invariant representation methods require training data from multiple environments, including *Independence of causal mechanism (ICM)* methods (Bengio et al., 2019; Besserve et al., 2018; Johansson et al., 2016; Louizos et al., 2017; Raj et al., 2020; Schölkopf, 2019), *Causal Discovery from Change (CDC)* methods (Tian & Pearl, 2001), and *representation disentanglement* methods (Bengio et al., 2019; Goudet et al., 2017; Locatello et al., 2019).

Contributions. Our work contributes to a growing literature by providing, to the best of our knowledge, the first systematic counterfactual modeling framework for extrapolations in graph classification/regression tasks. Existing work, e.g., the parallel work of Xu et al. (2020), define extrapolations geometrically which, while interesting, have a different scope. Our work connects Lovász's graph limit theory with graph-size extrapolation in a family of graph classification and regression tasks. Moreover, existing graph classification/regression methods —including graph neural networks and graph kernels— are generally evaluated on generalization error, which effectively tests only how

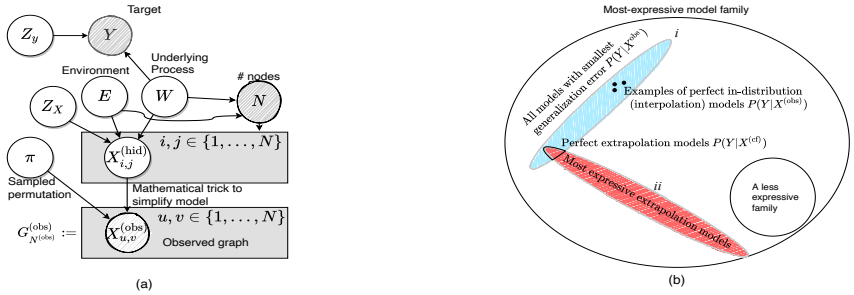

Figure 1: (a) The DAG of the structural causal model (SCM) of our graph extrapolation tasks where hashed (white) vertices represent observed (hidden) variables; (b) Illustrates the relationship between expressive model families and most-expressive extrapolation families.

well they interpolate the training data. We provide a systematic evaluation of these interpolation methods on verifiable extrapolation tasks.

2 A Family of Graph Extrapolation Tasks

Geometrically, extrapolation can be thought as reasoning beyond a convex hull of a set of training points (Hastie et al., 2012; Haffner, 2002; King & Zeng, 2006; Xu et al., 2020). However, for neural networks —and their arbitrary representation mappings— this geometric interpretation is insufficient to describe a truly broad range of tasks. Rather, extrapolations are better described through counterfactual reasoning (Neyman, 1923; Rubin, 1974; Pearl, 2009; Schölkopf, 2019). Specifically we want to ask: *After seeing training data from environment A, how to extrapolate and predict what would have been the model predictions of a test example from an unknown environment B, had the training data also been from B.* For instance, what would have been the model predictions for a large test example graph if our training data had also been large graphs rather than small ones?

A structural causal model for graph classification and regression tasks. In many applications, graphs are simply *representations of a natural process rather than the true state of the process*. In what follows we assume all graphs are simple, meaning all pairs of vertices have at most one edge. Our work defines an n-vertex attributed graph as a sample of a random variable $\mathcal{G}_n := (X^{(\text{obs})}_{1,1}, \ldots, X^{(\text{obs})}_{n,n})$, where $X^{(\text{obs})}_{i,j} \in \Omega^{(\text{e})}$ encodes edges and edge attributes and $X^{(\text{obs})}_{i,i} \in \Omega^{(\text{v})}$ encodes vertex attributes; we will assume $\Omega = \Omega^{(\text{v})} = \Omega^{(\text{e})}$ for simplicity. Consider a supervised task over a graph input $G_n (n \geq 2)$ and its corresponding output Y. We describe the graph and target generation process with the help of a structural causal model (SCM) (Pearl, 2009, Definition 7.1.1).

We first consider a *hidden* random variable $E \in \mathbb{Z}^+$ that describes the graph-processing environment. We also consider an independent hidden random variable W over some arbitrary domain that defines functional topological and attribute characteristics of the graph that are independent of the environment variable E. In the SCM, these two variables are inputs to a deterministic graph-generation function $g : \mathbb{Z}^+ \times \mathbb{D} \times \mathbb{D} \to \Omega^{n \times n}$ that outputs

$$\mathcal{G}^{(\text{hid})}_{N^{(\text{obs})}} := (X^{(\text{hid})}_{1,1}, \ldots, X^{(\text{hid})}_{N^{(\text{obs})}, N^{(\text{obs})}}) = g_E(W, Z_X), \text{ with } N^{(\text{obs})} := \eta(E, W), \tag{1}$$

where Z_X is another independent random variable that defines external noise (likely measurement noise of a device). Equation (1) gives edge and vertex attributes of the graph $\mathcal{G}^{(\text{hid})}_{N^{(\text{obs})}}$ in some canonical order, where η is a function of both E and W that gives the number of vertices in the graph. To understand our definitions, consider the following simple example (divided into two parts).

Erdös-Renyi example (part 1): For a single environment e, let $n = \eta(e)$ be the (fixed) number of vertices of the graphs in our training data, and $p = W$ be the probability that any two vertices of the graph have an edge. Finally, the variable Z_X can be thought as the seed of a random number generator that is drawn $\frac{n(n-1)}{2}$ times to determine if two distinct vertices are connected by an edge. The above defines our training data as a set of Erdös-Renyi random graphs of size n with $p = W$.

The data generation process in Equation (1) could leak information about W through the vertex ids (the order of the vertices). Rather than restricting how W acts on $(X^{(\text{hid})}_{1,1}, \ldots, X^{(\text{hid})}_{N^{(\text{obs})}, N^{(\text{obs})}})$, we

remedy this by adding a random permutation to the vertex indices.

$$\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}} := (X^{(\text{obs})}_{1,1}, \dots, X^{(\text{obs})}_{N^{(\text{obs})},N^{(\text{obs})}}) = (X^{(\text{hid})}_{\pi(1),\pi(1)}, \dots, X^{(\text{hid})}_{\pi(N^{(\text{obs})}),\pi(N^{(\text{obs})})}), \tag{2}$$

where $\pi \sim \text{Uniform}(\mathbb{S}_{N^{(\text{obs})}})$ is an uniform permutation of the indices $\{1, \dots, N^{(\text{obs})}\}$ and $\mathbb{S}_{N^{(\text{obs})}}$ is the permutation group. The observed graph is the outcome of this joint permutation of vertex ids.

SCM target variable. We now define our target variable Y. The *true* target of $\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}}$ is

$$Y = h(W, Z_Y), \tag{3}$$

which is given by a deterministic function $h(\cdot)$ that depends only on W and a random noise Z_y independent of W and E. Our final structural causal model is summarized in the DAG of Figure 1(a).

Erdős-Rényi example (part 2): The targets of the Erdős-Rényi graphs in our previous example can be, for instance, the value $Y = W$ in Equation (3), which is also the edge probability p.

Graph extrapolation tasks over new environments. Equation (3) shows that our target variable Y is a function only of W, the functional characteristics of the graph, rather than the graph-processing environment E. Due to the reverse path between Y and E through $\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}}$ in the DAG of Figure 1(a), Y is not independent of E given $\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}}$. These non-causal paths are called backdoor paths since they flow backwards from Y and $\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}}$. Hence, traditional (interpolation) methods can pick-up this correlation, which prevents the learnt model from extrapolating over environments different than the ones provided in the training data (or even over different $P(E)$ distributions).

To address the challenge of predicting Y with backdoor paths, we need a backdoor adjustment (Pearl, 2009, Theorem 3.3.2). Instead of explicitly conditioning on the environment, we seek a graph representation that allows us to fulfill the backdoor adjustment for the SCM in Figure 1(a), as we will show in Proposition 1. Before we proceed, we note that the existing counterfactual notation in the literature (see Definition 7 of Bareinboim et al. (2020)) could be ambiguous in our setting. Hence, we re-propose the powerful concept of random variable coupling from Markov chains (Pitman, 1976; Propp & Wilson, 1996) to describe our counterfactual inference problem:

Definition 1 (Counterfactual coupling (CFC))**.** *A counterfactual coupling of Equations* (1) *to* (3) *is*

$$\begin{aligned} &P(Y = y, \mathcal{G}^{(obs)}_{N^{(obs)}} = G^{(obs)}_{n^{(obs)}}, \mathcal{G}^{(cf)}_{N^{(cf)}} = G^{(cf)}_{n^{(cf)}}) \\ &= \mathbb{E}_{W,Z_X,Z_Y,\pi,E,\tilde{E}}\Big[\mathbb{1}\{y = h(W, Z_Y)\} \cdot \mathbb{1}\{G^{(obs)}_{n^{(obs)}} = \pi(g_E(W, Z_X))\} \\ &\qquad \cdot \mathbb{1}\{G^{(cf)}_{n^{(cf)}} = \pi(g(\tilde{E}, W, Z_X))\} \cdot \mathbb{1}\{n^{(obs)} = \eta(E, W)\} \cdot \mathbb{1}\{n^{(cf)} = \eta(\tilde{E}, W)\}\Big], \end{aligned} \tag{4}$$

where $\mathcal{G}^{(obs)}_{N^{(obs)}} := (X^{(obs)}_{1,1}, \dots, X^{(obs)}_{N^{(obs)},N^{(obs)}})$ *and* $\mathcal{G}^{(cf)}_{N^{(cf)}} := (X^{(cf)}_{1,1}, \dots, X^{(cf)}_{N^{(cf)},N^{(cf)}})$*,* $\pi(\cdot)$ *is defined below, and* E *and* $\tilde{E}$ *are independent random variables that sample environments, potentially with different distributions and supports, and* $\mathbb{1}$ *is the Dirac delta function. The counterfactual coupled variable* $\mathcal{G}^{(cf)}_{N^{(cf)}}$ *asks what would have happened to* $\mathcal{G}^{(obs)}_{N^{(obs)}}$ *if we had used the environment random variable* $\tilde{E}$ *in place of* E *in Equation* (1)*. In an abuse of notation we have defined* $\pi(G^{(\cdot)}_N) := (X^{(\cdot)}_{\pi(1),\pi(1)}, \dots, X^{(\cdot)}_{\pi(N),\pi(N)})$ *above.*

Using Definition 1 we now prove that a graph representation function $\Gamma(\cdot)$ that is E-invariant encodes a backdoor adjustment between $\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}}$, $N^{(\text{obs})}$, E, and Y.

Proposition 1. *Let* $P(Y|\mathcal{G}^{(obs)}_{N^{(obs)}} = G^{(obs)}_{n^{(obs)}})$ *and* $P(Y|\mathcal{G}^{(cf)}_{N^{(cf)}} = G^{(cf)}_{n^{(cf)}})$ *be the conditional target distributions defined by the counterfactually-coupled random variables in Definition 1. For simplicity, assume* $Y \in \mathcal{Y}$ *is discrete. Consider a permutation-invariant graph representation* $\Gamma : \cup_{n=1}^{\infty} \Omega^{n \times n} \to \mathbb{R}^d$*,* $d \geq 1$*, and a link function* $\rho(\cdot, \cdot)$ *such that, for some* $\epsilon, \delta > 0$*, the generalization (interpolation) error is defined as*

$$P(|P(Y = y|\mathcal{G}^{(obs)}_{N^{(obs)}} = G^{(obs)}_{n^{(obs)}}) - \rho(y, \Gamma(G^{(obs)}_{n^{(obs)}}))| \leq \epsilon) \geq 1 - \delta\,, \quad \forall y \in \mathcal{Y},$$

and Γ *is said* **environment-invariant (E-invariant)** *if* $\Gamma(\mathcal{G}^{(obs)}_{N^{(obs)}}) \overset{a.s.}{=} \Gamma(\mathcal{G}^{(cf)}_{N^{(cf)}})$*, where a.s. (almost surely) means* $\Gamma(G^{(obs)}_{n^{(obs)}}) = \Gamma(G^{(cf)}_{n^{(cf)}})$ *for any graphs* $G^{(obs)}_{n^{(obs)}}$ *and* $G^{(cf)}_{n^{(cf)}}$ *that can be sampled. Then, the extrapolation error is*

$$P(|P(Y = y|\mathcal{G}^{(cf)}_{N^{(cf)}} = G^{(cf)}_{n^{(cf)}}) - \rho(y, \Gamma(G^{(cf)}_{n^{(cf)}}))| \leq \epsilon) \geq 1 - \delta\,, \quad \forall y \in \mathcal{Y}. \tag{5}$$

Proposition 1 shows that an E-invariant representation will perform no worse on the counterfactual test data (extrapolation samples from $(Y, \mathcal{G}^{(\text{cf})}_{N^{(\text{cf})}})$) than on a test dataset having the same environment distribution as the training data (samples from $(Y, \mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}})$). Other notions of E-invariant representations are possible (Arjovsky et al., 2019; Schölkopf, 2019), but ours —through coupling— provides a direct relationship with how we learn graph representations from a single training environment. *Our task now becomes finding an E-invariant graph representation* Γ *that generalizes (interpolates) well over the training data distribution.*

In recent years, a crop of interesting research has analyzed the expressiveness of Γ. In what follows we explain why these are related to interpolations rather than extrapolations.

A comment on most-expressive graph representations, interpolations, and extrapolations. The expressiveness of a graph classification/regression method is a measure of model family bias (Morris et al., 2019; Xu et al., 2018a; Gärtner et al., 2003; Maron et al., 2019a; Murphy et al., 2019b). That is, given enough training data, a neural network from a more expressive family can achieve smaller generalization error (interpolation error) than a neural network from a less expressive family, assuming appropriate optimization. However, this power is just a measure of interpolation capability, not extrapolation. Figure 1(b) illustrates a space where each point is a set of neural network parameters from a most-expressive model family. The blue region (ellipsoid i) represents models that can perfectly interpolate over the training distribution (i.e., models with the smallest generalization error). The models in the blue region are mostly fitting spurious training environment E correlations with Y, that will cause poor extrapolations in new environments.

The models illustrated in the red region of Figure 1(b) (ellipsoid ii) are E-invariant and, thus, by Proposition 1, can extrapolate across environments, since they cannot fit these spurious environment correlations. The intersection between the blue and red regions contains models that are optimal both for test data from the same environment distribution as training (**interpolation test**) and test data from a different environment distribution (**extrapolation test**). In our SCM in Equations (1) to (3), the intersection between the blue and red ellipsoids is nonempty. We can denote the models in the red ellipsoid as the most-expressive family of *E-invariant* (Proposition 1). Our work focuses on a family of classifiers and regression models that reside inside the red ellipsoid.

Summary. In this section we have defined a family of extrapolation tasks for graph classification and regression using counterfactual modelling, and connected it to the existing literature. Next, we show how these definitions can be applied to a family of random graph models (graphons) first introduced by Diaconis & Freedman (1981).

3 Graph Size Extrapolations and Graphons

Graph datasets are special in that their characteristics can be stable as their size grows (measured in number of vertices). We propose a neural network representation that can be E-invariant given only one environment in training by making use of graphon concentration inequalities. We start with Theorem 1 which gives necessary and sufficient conditions for using these inequalities.

Theorem 1. *Assume our graph-processing heuristic gives the number of vertices as* $N^{(obs)} = \eta(E)$ *and the outputs of* g_{e_1} *and* g_{e_2} *of Equation* (1) *can only differ in their attributes* $\forall e_1, e_2$. *Let* $\overline{\mathcal{G}}^{(obs)}_n|W := \mathbb{E}_E[\overline{\mathcal{G}}^{(obs)}_n|W, N^{(obs)} = n, E]$ *be the* n*-vertex graph output of our graph-processing heuristic over the true underlying data variable* W. *If* $\overline{\mathcal{G}}^{(obs)}_n|W$ *satisfies the following properties:*

1. *Deleting a random vertex* n *from* $\overline{\mathcal{G}}^{(obs)}_n|W$*, and the distribution of the trimmed graph is the same as the distribution of* $\overline{\mathcal{G}}^{(obs)}_{n-1}|W$*, with* $\overline{\mathcal{G}}^{(obs)}_1|W$ *as a trivial graph with a single vertex for all* W.

2. *For every* $1 < k < n$*, the subgraphs of* $\overline{\mathcal{G}}^{(obs)}_n|W$ *induced by* $\{1, \ldots, k\}$ *and* $\{k+1, \ldots, n\}$ *are independent random variables.*

Then, the variable W *can be equivalently defined as* $W = (W', C'_E)$*, where* W' *is a random variable defined over the family of symmetric measurable functions* $W' : [0,1]^2 \to [0,1]$*, i.e.,* W' *is a random graphon function, and, if the graph has attributes,* C'_E *is an environment-dependent random variable that defines vertex and edge attributes, otherwise,* $C'_E = \emptyset$ *is defined as the constant* null.

Under the conditions posed in Theorem 1, it is possible to guarantee that a graph representation is E-invariant even when the training data contains just one environment. Then, by Proposition 1, we

can obtain a model with extrapolation power (assuming the target is independent of E) by passing the E-invariant learnt representation to a downstream classifier such as a neural network or logistic regression. We investigate ways to achieve E-invariance for unattributed and attributed graphs.

3.1 EXTRAPOLATIONS FOR UNATTRIBUTED GRAPHS

We now define an E-invariant graph representation function Γ for all unattributed graph models satisfying the conditions in Theorem 1. Let F_k be an arbitrary k-vertex unattributed graph, and $\text{inj}(F, G)$ be the number of injective homomorphisms of F into a larger unattributed graph G, informally, the number of copies of F in G where we match the edges of F into G but not the nonedges. The injective homomorphism density over the n-vertex graph G_n, $n > k$ is defined as:

$$t_{\text{inj}}(F_k, G_n) = \frac{\text{inj}(F_k, G_n)}{n!/(n-k)!}. \tag{6}$$

The following is a simple but effective representation (feature vector) of G_n. Let $\mathcal{F}_{\leq k}$ denote a totally ordered set (w.l.o.g.) of all possible k'-vertex graphs ($1 \leq k' \leq k$) and $\mathbf{1}_{\text{one-hot}}\{F_{k'}, \mathcal{F}_{\leq k}\}$ be the one-hot vector with a one at the index of $F_{k'}$ in $\mathcal{F}_{\leq k}$ and zeros elsewhere. The representation

$$\Gamma_{\text{1-hot}}(G_n) = \sum_{F_{k'} \in \mathcal{F}_{\leq k}} t_{\text{inj}}(F_{k'}, G_n) \mathbf{1}_{\text{one-hot}}\{F_{k'}, \mathcal{F}_{\leq k}\}, \tag{7}$$

is a vector containing the densities of each type of k'-sized ($k' \leq k$) graph in G_n. The following theorem shows the ability of $\Gamma_{\text{1-hot}}(\overline{\mathcal{G}}_n^{(\text{obs})}|W)$ to be an approximately E-invariant representation in a training dataset with input graphs $\overline{\mathcal{G}}_n^{(\text{obs})}|W$ as given in Theorem 1:

Theorem 2. *Let $\overline{\mathcal{G}}_n^{(obs)}|W$ and $\overline{\mathcal{G}}_{n'}^{(obs)}|W$ be two graphs of sizes n and n', respectively, satisfying Theorem 1. Note that n can be equal to n'. Let $\Gamma_{\text{1-hot}}(\overline{\mathcal{G}}_n^{(obs)}|W)$ be defined as in Equation (7) and $||\cdot||_\infty$ denote the L-infinity norm.. Then, for any integer $k \leq n$, $0 < \epsilon < 1$,*

$$\Pr(||\Gamma_{\text{1-hot}}(\overline{\mathcal{G}}_n^{(obs)}|W) - \Gamma_{\text{1-hot}}(\mathcal{G}_{n'}^{(obs)}|W)||_\infty > \epsilon) \leq 2|\mathcal{F}_{\leq k}|(\exp(-\frac{\epsilon^2}{8k^2}n) + \exp(-\frac{\epsilon^2}{8k^2}n')). \tag{8}$$

Theorem 2 shows that for $k \ll \min(n, n')$, the representations $\Gamma_{\text{1-hot}}(\cdot)$ of two possibly different-sized graphs with the same W are nearly identical. Hence, $\Gamma_{\text{1-hot}}(\overline{\mathcal{G}}_{N^{(\text{obs})}}^{(\text{obs})}|W)$ is an approximately E-invariant representation for $\overline{\mathcal{G}}_{N^{(\text{obs})}}^{(\text{obs})}|W$. Theorem 2 also exposes a trade-off, however. If the observed graphs tend to be relatively small, the required k for nearly E-invariant representations can be small, and, as a result, the expressiveness of $\Gamma_{\text{1-hot}}(\cdot)$ gets compromised. That is, the ability of $\Gamma_{\text{1-hot}}(\cdot)$ to extract information about W from $\overline{\mathcal{G}}_{N^{(\text{obs})}}^{(\text{obs})}|W$ reduces as k decreases. Finally, this guarantees that for appropriate k, passing the representation $\Gamma_{\text{1-hot}}(\overline{\mathcal{G}}_n^{(\text{obs})}|W)$ to a downstream classifier provably approximates the classifier in Equation (5) of Proposition 1. We defer the choice of downstream model and respective bounds to future work and now turn our attention to attributed graphs.

3.2 EXTRAPOLATIONS FOR ATTRIBUTED GRAPHS

We now extend the representation $\Gamma_{\text{1-hot}}(\cdot)$ in Equation (7) to attributed graphs G_n. Attributed graph extrapolation models should also represent the attribute-definer variable C'_E of Theorem 1, but be E-invariant if possible. Hence, we *should not* just extend Equation (7) by making $F_{k'}$ attributed and generalize the injective homomorphism density of Equation (6) to $t_{\text{a-inj}}(F_{k'}, G_n)$ which counts attributed graphs, as the representation would not be E-invariant.

To create attributed graph representations that are less sensitive to environments (but not E-invariant, unfortunately), we start with three observations: First, $\Gamma_{\text{1-hot}}(\cdot)$ in Equation (7) is still E-invariant for attributed graphs, but only carries information about the graph structure (W' of Theorem 1), not its attributes (C'_E of Theorem 1). Second, graph neural networks(GNNs) (Kipf & Welling, 2017; Hamilton et al., 2017; You et al., 2019) learn representations that can capture information from vertex attributes (and edge attributes with some ingenuity). Third, in their Eric-Irma discussions, Arjovsky et al. (2019) observes that very expressive, over-parametrized, neural networks are more

prone to be E-invariant than low capacity representations, since low capacity representations prefer exploiting spurious correlations which tend to be easier to detect.

Hence, our proposal replaces the one-hot vector $\mathbf{1}_{\text{one-hot}}\{F_{k'}, \mathcal{F}_{\leq k}\}$ with a GNN applied to $F_{k'}$:

$$\Gamma_{\text{GNN}}(G_n) = \sum_{F_{k'} \in \mathcal{F}_{\leq k}} t_{\text{a-inj}}(F_{k'}, G_n)\text{READOUT}(\text{GNN}(F_{k'})), \tag{9}$$

where READOUT is a permutation-invariant representation such as a sum, Deep Sets (Zaheer et al., 2017), or Janossy Pooling (Murphy et al., 2019a), and $t_{\text{a-inj}}(F_{k'}, G_n)$ is the injective homomorphism density defined over attributed graphs. Unfortunately, GNNs are not most-expressive representations of graphs (Morris et al., 2019; Murphy et al., 2019b; Xu et al., 2018a) and thus $\Gamma_{\text{GNN}}(\cdot)$ is less expressive than $\Gamma_{\text{1-hot}}(\cdot)$ for unattributed graphs. A representation with greater expressive power is

$$\Gamma_{\text{GNN}^+}(G_n) = \sum_{F_{k'} \in \mathcal{F}_{\leq k}} t_{\text{a-inj}}(F_{k'}, G_n)\text{READOUT}(\text{GNN}^+(F_{k'})), \tag{10}$$

where GNN^+ is a most-expressive k-vertex graph representation, which can be achieved by any of the methods of Vignac et al. (2020); Maron et al. (2019a); Murphy et al. (2019b). Since GNN^+ is most expressive, GNN^+ can ignore attributes and map each $F_{k'}$ to a one-hot vector $\mathbf{1}_{\text{one-hot}}\{F_{k'}, \mathcal{F}_{\leq k}\}$; therefore, $\Gamma_{\text{GNN}^+}(\cdot)$ generalizes $\Gamma_{\text{1-hot}}(\cdot)$ of Equation (7) and can choose to be E-invariant by disregarding information about the attributes.

3.3 Practical considerations

While the literature does not offer fast algorithms to count all possible k-vertex injective homomorphism densities in a graph, there is a bijection between *induced* and *injective* homomorphism densities (Borgs et al., 2006). So, we can use *induced* homomorphism densities in our representations without losing expressiveness. While this remains expensive – taking at least $n^{\Omega(k)}$ running time (Chen et al., 2005) if the Exponential Time Hypothesis (Impagliazzo et al., 2001) is true – efficient algorithms exist to *estimate* induced homomorphism densities over all possible *connected* k-vertex subgraphs (Ahmed et al., 2016; Bressan et al., 2017; Chen & Lui, 2018; Chen et al., 2016; Rossi et al., 2019; Wang et al., 2014). Since the densities of *disconnected* k'-vertex subgraphs are likely very correlated with that of *connected* k''-vertex subgraphs, $k'' < k'$, there should be little information lost in restricting $\mathcal{F}_{\leq k}$ in Equations (7), (9) and (10) to contain only connected $F_{k'}$.

For unattributed graphs and $k \leq 5$, we use ESCAPE (Pinar et al., 2017) to obtain exact *induced* homomorphism densities of each connected subgraph of size $\leq k$. For attributed graphs or unattributed graphs with $k > 5$, exact counting becomes intractable so we use R-GPM (Teixeira et al., 2018) to obtain unbiased estimates of *induced* homomorphism counts, from which we compute densities. Finally, Proposition 2 in the Appendix shows that certain biased estimators can be used without losing information in the representations in Equation (10) if READOUT is the sum of vertex embeddings.

4 Related Work

This section presents an overview of the related work. Due to space constraints, a more in-depth discussion with further references are given in the Appendix. In particular, the Appendix gives a detailed description of environment-invariant methods that require multiple environments in training, including *Independence of Causal Mechanism (ICM)*, *Causal Discovery from Change (CDC)* methods, and *representation disentanglement* methods. Also, none of these works focus on graphs.

Counterfactual mechanisms in graph classification/regression and other extrapolation work. There are two key sources of causal relationships on graph classification/regression tasks: *Conterfactuals on graphs*, interested in cause-effects events related to processes running on top of a graph, such as Eckles et al. (2016a;b). *Conterfactuals of graphs*, which is the topic of our work, where we want to ascertain a counterfactual relationship between graphs and their targets in the tasks. We are unaware of prior work in this topic. The parallel work of Xu et al. (2020) (already discussed) is interested in the narrower geometric definition of extrapolation. Previous works also examine empirically the ability of graph networks to extrapolate in physics (Battaglia et al., 2016; Sanchez-Gonzalez et al., 2018), mathematical and abstract reasoning (Santoro et al., 2018; Saxton et al., 2019), and graph algorithms (Bello et al., 2017; Nowak et al., 2017; Battaglia et al., 2018; Velickovic et al., 2018). These works offer little theoretical analysis for why these methods should extrapolate, or a proof

Table 1: Extrapolation performance over **unattributed** graphs **shows clear advantage of environment-invariant representations $\Gamma_\cdot$, with or without GNN, over standard (interpolation) methods in extrapolation test accuracy**. Interpolation and extrapolation distributions contain different-size graphs. (Left) Classifies schizophrenic individuals using brain functional networks where graphs are on average 40% smaller at extrapolation environment. (Right) A supposedly easy classification task with $Y = p \in \{0.2, 0.5, 0.8\}$ as the edge probabilities of the Erdős-Rényi graph, whose sizes are $N^{(\text{obs})} \in \{20, \ldots, 80\}$ in train & test interpolation and $N^{(\text{obs})} \in \{140, \ldots, 200\}$ in test extrapolation. Results show mean (standard deviation) accuracy.

	Accuracy in Schizophrenia Task			Accuracy in Erdős-Rényi Task		
	Interpl. Train	Interpl. Test	**Extrapl. Test (↑)**	Interpl. Train	Interpl. Test	**Extrapl. Test (↑)**
GIN	0.68 (0.02)	0.71 (0.04)	0.41 (0.04)	0.99 (0.01)	0.99 (0.01)	0.36 (0.03)
RPGIN	0.74 (0.02)	0.72 (0.04)	0.44 (0.07)	0.99 (0.01)	1.00 (0.00)	0.36 (0.03)
WL Kernel	1.00 (0.00)	0.63 (0.07)	0.40 (0.00)	1.00 (0.00)	1.00 (0.00)	0.39 (0.00)
GC Kernel	0.61 (0.00)	0.61 (0.06)	0.60 (0.00)	1.00 (0.00)	1.00 (0.00)	**1.00 (0.00)**
$\Gamma_{\text{1-hot}}$ (eq. (7))	0.69 (0.01)	0.70 (0.06)	**0.70 (0.05)**	1.00 (0.00)	1.00 (0.00)	**1.00 (0.00)**
Γ_{GIN} (eq. (9))	0.68 (0.01)	0.71 (0.06)	**0.71 (0.04)**	1.00 (0.00)	1.00 (0.00)	**1.00 (0.00)**
Γ_{RPGIN} (eq. (10))	0.68 (0.01)	0.71 (0.04)	**0.69 (0.04)**	1.00 (0.00)	1.00 (0.00)	**1.00 (0.00)**

Table 2: Extrapolation performance over **attributed** graphs **shows clear advantage of environment-invariant methods that use GNNs**. We count $\#\{\text{5-cliques with no green vertices}\}$. Vertex color distribution changes with environment. Table shows Mean Absolute Error (MAE) over interpolation environment (train & test) and extrapolation test. Results show mean (standard deviation) MAE.

	Interpolation Train MAE	Interpolation Test MAE	**Extrapolation Test MAE (↓)**
Predict train target average	8.46 (0.00)	9.67 (0.00)	8.88 (0.00)
GIN	3.20 (0.80)	3.15 (0.37)	7.34 (0.64)
RPGIN	3.00 (0.73)	2.96 (0.30)	6.90 (0.73)
WL Kernel	6.33 (0.00)	7.11 (0.00)	8.52 (0.00)
GC Kernel (attributed)	4.46 (0.00)	4.66 (0.00)	7.36 (0.00)
GC Kernel (attributed + unattributed)	3.81 (0.00)	5.17 (0.00)	6.43 (0.00)
$\Gamma_{\text{1-hot}}$ (eq. (7))	1.78 (0.60)	3.31 (0.17)	6.17 (0.87)
Γ_{GIN} (eq. (9))	1.12 (0.29)	1.97 (0.80)	**3.92 (0.95)**
Γ_{RPGIN} (eq. (10))	1.57 (0.58)	1.60 (0.35)	**2.66 (0.65)**

that the tasks are really extrapolation tasks over different environments. We hope our work will help guide future extrapolation analysis.

Graph classification/regression using induced homorphism densities. A related interesting set of works look at induced homorphism densities as graph features for a kernel (Shervashidze et al., 2009; Yanardag & Vishwanathan, 2015; Wale et al., 2008). Kriege et al. (2018) reports that these methods can perform poorly in some tasks. These works focus on generalization (interpolation) error only.

GNN-type representations and subgraph methods. Common GNN methods lack the ability to distinguish nonisomorphic graphs (Morris et al., 2019; Xu et al., 2018a) and cannot count the number of subgraphs such as triangles (3-cliques) (Arvind et al., 2020; Chen et al., 2020). Proposed solutions (e.g. Dasoulas et al. (2019); Chen et al. (2020)) focus on making substructures distinguishable and thus expressivity/universality rather than learning functions that extrapolate. Closer to our representations, other methods based on subgraphs have been proposed. Procedures like mGCMN (Li et al., 2020), HONE (Rossi et al., 2018), and MCN (Lee et al., 2018) learn representations for vertices by extending methods defined over traditional neighborhood (edge) structures to higher-order graphs based on subgraphs; for instance, mGCMN applies a GNN on the derived graph. These methods will not learn subgraph representations in a manner consistent with our extrapolation task. These and other related works (detailed in the Appendix) focus on generalization (interpolation) error only.

5 Empirical Results

This section is dedicated to the empirical evaluation of our theoretical claims, including the ability of the representations in Equations (7), (9) and (10) to extrapolate in the manner predicted by Proposition 1 for tasks that abide by conditions 1 and 2 of Theorem 1. We also test their ability to extrapolate in tasks that do not perfectly fit conditions 1 and 2 of Theorem 1, and in a task with a real dataset. Our results report (i) *interpolation test* performance on held out graphs from the same environment used for training; and (ii) *extrapolation test* performance on held out graphs from different environments. Our code is available[1] and complete details are given in our Appendix.

[1] `https://anonymous.4open.science/r/8af8ed44-8114-4164-9610-94866ad28c3e`

Interpolation representations: We choose a few methods as examples of graph representation interpolations. While not an extensive list, these methods are representative of the literature. Graph Isomorphism Network (**GIN**) (Xu et al., 2018a); Relational Pooling GIN (**RPGIN**) (Murphy et al., 2019b); The Weisfeiler Lehman kernel (**WL Kernel**) (Shervashidze et al., 2011) uses the Weisfeiler-Leman algorithm (Weisfeiler & Lehman, 1968) to provide graph representations.

Extrapolation representations: We experiment with the three representations $\Gamma_{\text{1-hot}}$, Γ_{GNN}, in Equations (7) and (9), and Γ_{RPGNN}, where we use RPGIN as a method of GNN$^+$ in Equation (10). We also test Graphlet counting kernel (**GC Kernel**) (Shervashidze et al., 2009), which is a method that uses a $\Gamma_{\text{1-hot}}$ representation as input to a downstream classifier. We report $\Gamma_{\text{1-hot}}$ separately from GC Kernel since we wanted to add a better downstream classifier than the one used in Shervashidze et al. (2009). Per Section 3.3, we use connected induced subgraph (CIS) densities instead of induced homomorphisms. The CIS size k is a hyperparameter. Our attributed graph experiments rely on estimated CIS densities, an added source of error.

Extrapolation performance over unattributed graphs of varying size. For these unattributed graph experiments, the task is to extrapolate over environments with different graph sizes. These tasks fulfill the conditions imposed by Theorem 1, which allow us to test our theoretical results.

Schizophrenia task. We use the fMRI brain graph data on 71 schizophrenic patients and 74 controls for classifying individuals with schizophrenia (De Domenico et al., 2016). Vertices represent brain regions with edges as functional connectivity. We process the graph differently between interpolation and extrapolation data, where interpolation has exactly 264 vertices (a single environment) and extrapolation has in average 40% fewer vertices. The graphs are dense and processing approximate the conditions imposed by Theorem 1. The value of $k \in \{4, 5\}$ and chosen based on a separate validation error over the interpolation environment. Further details are provided in the Appendix.

Erdős-Rényi task. This is an easy interpolation task. We simulate Erdős-Rényi graphs (Gilbert, 1959; Erdős & Rényi, 1959) which by design perfectly satisfies the conditions in Theorem 1. There are two environments: we train and measure interpolation accuracy graphs of size in $\{20 \dots 80\}$; we extrapolate to graphs from an environment with size in $\{140 \dots 200\}$. The task is to classify the edge probability $p \in \{0.2, 0.5, 0.8\}$ of the generated graph. Further details are in the Appendix.

Unattributed graph results: Table 1 shows that our results perfectly follow Proposition 1 and Theorem 2, where representations $\Gamma_{\text{1-hot}}$ (GC Kernel and new classifier), Γ_{GNN}, Γ_{RPGNN} are the only ones able to extrapolate, while displaying very similar —often identical— interpolation and extrapolation test accuracies in all experiments. All methods perform well in the easier interpolation task.

Extrapolation performance over attributed graphs over varying attributes. Next we try a significantly more challenging scenario, *with conditions that clearly violate Theorem 1*. Here, the attributed graph environments have a shift in observed attributes. We simulate Erdős-Rényi graphs with $N^{(\text{obs})} \sim \text{Uniform}(20, \dots, 25)$ for both interpolation and extrapolation environments. Vertices have red, green, or blue attributes (scheme in Appendix). Target $Y \sim \#\{\text{5-cliques with no green vertices}\}$. In the interpolation environment, 5-cliques are predominantly red, while in extrapolation their colors are more uniform. Representations $\Gamma_{\text{1-hot}}$, Γ_{GNN}, Γ_{RPGNN} use estimates of attributed $k' = 5$ CIS counts, rather than densities due to the task. A representation that learns to merge red and blue clique counts will perform well.

Attributed graph results: Table 2 shows the Mean Absolute Error (MAE) results. We include a *train target average* predictor to provide a reference for a bad MAE. The results show that interpolation representations and $\Gamma_{\text{1-hot}}$ (GC Kernel and new classifier) get distracted by the easy relationship between Y and the density of red cliques, while Γ_{GNN} and Γ_{RPGNN} are significantly more robust, giving similar GNN representations to red and blue cliques. Γ_{GNN} and Γ_{RPGNN} show a gap between interpolation and extrapolation test errors, likely reflecting the deviation in Theorem 1 conditions.

6 CONCLUSIONS

Our work contributes to a growing literature by providing the first systematic counterfactual modeling framework for extrapolations in graph classification/regression tasks. We connected a family of graph extrapolation tasks with Lovász theory of graph limits, and introduced environment-invariant (E-invariant) representations that can provably extrapolate in such scenarios. Our experiments validated our theoretical results and the shortcomings of common (interpolation) methods.

REFERENCES

Ghadeer AbuOda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. Link prediction via higher-order motif features. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 412–429. Springer, 2019.

Nesreen K Ahmed, Theodore L Willke, and Ryan A Rossi. Estimation of local subgraph counts. In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 586–595. IEEE, 2016.

David J Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

Uri Alon. Network motifs: theory and experimental approaches. *Nature Reviews Genetics*, 8(6): 450–461, 2007.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

J Scott Armstrong, Fred Collopy, and J Thomas Yokum. Decomposition by causal forces: a procedure for forecasting complex time series. *International Journal of forecasting*, 21(1):25–36, 2005.

Vikraman Arvind, Frank Fuhlbrück, Johannes Köbler, and Oleg Verbitsky. On weisfeiler-leman invariance: subgraph counts and related graph properties. *Journal of Computer and System Sciences*, 2020.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

Elias Bareinboim, Juan Correa, Duligur Ibeling, and Thomas Icard. On Pearl's hierarchy and the foundations of causal inference. *ACM special volume in honor of Judea Pearl*, 2020.

Jordi Bascompte and Carlos J Melián. Simple trophic modules for complex food webs. *Ecology*, 86 (11):2868–2873, 2005.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591, 2002.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations*, 2017.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv preprint arXiv:1901.10912*, 2019.

Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

Michel Besserve, Naji Shajarisales, Bernhard Schölkopf, and Dominik Janzing. Group invariance principles for causal generative models. In *International Conference on Artificial Intelligence and Statistics*, pp. 557–565, 2018.

Christian Borgs, Jennifer Chayes, László Lovász, Vera T Sós, and Katalin Vesztergombi. Counting graph homomorphisms. In *Topics in discrete mathematics*, pp. 315–371. Springer, 2006.

Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pp. 8–pp. IEEE, 2005.

Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. Counting graphlets: Space vs time. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM'17)*, pp. 557–566. ACM, 2017.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems*, pp. 4868–4879, 2019.

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *arXiv preprint arXiv:2005.03675*, 2020.

Jianer Chen, Benny Chor, Mike Fellows, Xiuzhen Huang, David Juedes, Iyad A Kanj, and Ge Xia. Tight lower bounds for certain parameterized np-hard problems. *Information and Computation*, 201(2):216–231, 2005.

Lina Chen, Xiaoli Qu, Mushui Cao, Yanyan Zhou, Wan Li, Binhua Liang, Weiguo Li, Weiming He, Chenchen Feng, Xu Jia, et al. Identification of breast cancer patients based on human signaling network motifs. *Scientific reports*, 3:3368, 2013.

Xiaowei Chen and John CS Lui. Mining graphlet counts in online social networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 12(4):1–38, 2018.

Xiaowei Chen, Yongkun Li, Pinghui Wang, and John Lui. A general framework for estimating graphlet statistics via random walk. *arXiv preprint arXiv:1603.07504*, 2016.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *arXiv preprint arXiv:2002.04025*, 2020.

George Dasoulas, Ludovic Dos Santos, Kevin Scaman, and Aladin Virmaux. Coloring graph neural networks for node disambiguation. *arXiv preprint arXiv:1912.06058*, 2019.

Manlio De Domenico, Shuntaro Sasai, and Alex Arenas. Mapping multiplex hubs in human functional brain networks. *Frontiers in neuroscience*, 10:326, 2016.

Asim K Dey, Yulia R Gel, and H Vincent Poor. What network motifs tell us about resilience and reliability of complex networks. *Proceedings of the National Academy of Sciences*, 116(39): 19368–19373, 2019.

Persi Diaconis and David Freedman. On the statistics of vision: the julesz conjecture. *Journal of Mathematical Psychology*, 24(2):112–138, 1981.

Dean Eckles, Brian Karrer, and Johan Ugander. Design and analysis of experiments in networks: Reducing bias from interference. *Journal of Causal Inference*, 5(1), 2016a.

Dean Eckles, René F Kizilcec, and Eytan Bakshy. Estimating peer effects in networks with peer encouragement designs. *Proceedings of the National Academy of Sciences*, 113(27):7316–7322, 2016b.

P Erdős and A Rényi. On random graphs i. *Publ. math. debrecen*, 6(290-297):18, 1959.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

V. K. Garg, S. Jegelka, and T. Jaakkola. Generalization and representational limits of graph neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2020.

Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning theory and kernel machines*, pp. 129–143. Springer, 2003.

Edgar N Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. Causal generative neural networks. *arXiv preprint arXiv:1711.08936*, 2017.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of KDD*, pp. 855–864. ACM, 2016.

Patrick Haffner. Escaping the convex hull with extrapolated vector machines. In *Advances in Neural Information Processing Systems*, pp. 753–760, 2002.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.

Patric Hagmann, Maciej Kurant, Xavier Gigandet, Patrick Thiran, Van J Wedeen, Reto Meuli, and Jean-Philippe Thiran. Mapping human whole-brain structural networks with diffusion mri. *PloS one*, 2(7):e597, 2007.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

William L. Hamilton. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*, volume 1. Springer series in statistics, 2012.

Robert L Hemminger. On reconstructing a graph. *Proceedings of the American Mathematical Society*, 20(1):185–187, 1969.

Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.

Douglas N Hoover. Relations on probability spaces and arrays of random variables. *Technical Report, Institute for Advanced Study, Princeton, NJ*, 2, 1979.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Maximilian Ilse, Jakub M Tomczak, and Max Welling. Attention-based deep multiple instance learning. *arXiv preprint arXiv:1802.04712*, 2018.

Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pp. 3020–3029, 2016.

Olav Kallenberg. *Probabilistic symmetries and invariance principles*. Springer Science & Business Media, 2006.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 321–328, 2003.

Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

Paul J Kelly et al. A congruence theorem for trees. *Pacific Journal of Mathematics*, 7(1):961–968, 1957.

Gary King and Langche Zeng. The dangers of extreme counterfactuals. *Political Analysis*, 14(2): 131–159, 2006.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. *arXiv preprint arXiv:2003.03123*, 2020.

Boris Knyazev, Graham W Taylor, and Mohamed Amer. Understanding attention and generalization in graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 4202–4212, 2019.

Nils M Kriege, Christopher Morris, Anja Rey, and Christian Sohler. A property testing framework for the theoretical expressivity of graph kernels. In *IJCAI*, pp. 2348–2354, 2018.

Nils M Kriege, Fredrik D Johansson, and Christopher Morris. A survey on graph kernels. *Applied Network Science*, 5(1):1–42, 2020.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, pp. 1269–1278, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362016.

Guillaume Lample and François Charton. Deep learning for symbolic mathematics. In *International Conference on Learning Representations*, 2020.

John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. Higher-order graph convolutional networks. *arXiv preprint arXiv:1809.07697*, 2018.

Xing Li, Wei Wei, Xiangnan Feng, Xue Liu, and Zhiming Zheng. Representation learning of graphs using graph convolutional multilayer networks based on motifs. *arXiv preprint arXiv:2007.15838*, 2020.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 8230–8241, 2019.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124, 2019.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, pp. 6446–6456, 2017.

László Lovász. *Large networks and graph limits*, volume 60. American Mathematical Soc., 2012.

László Lovász and Balázs Szegedy. Limits of dense graph sequences. *Journal of Combinatorial Theory, Series B*, 96(6):933–957, 2006.

Shmoolik Mangan and Uri Alon. Structure and function of the feed-forward loop network motif. *Proceedings of the National Academy of Sciences*, 100(21):11980–11985, 2003.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*, pp. 2156–2167, 2019a.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019b.

Brendan D McKay. Small graphs are reconstructible. *Australasian Journal of Combinatorics*, 15: 123–126, 1997.

Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of Symbolic Computation*, 60:94–112, 2014.

Changping Meng, S Chandra Mouli, Bruno Ribeiro, and Jennifer Neville. Subgraph pattern neural networks for high-order graph evolution prediction. In *AAAI*, pp. 3778–3787, 2018.

Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 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.

Elizabeth Munch. A user's guide to topological data analysis. *Journal of Learning Analytics*, 4(2): 47–61, 2017.

R. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*, 2019a.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *Proceedings of the 36th International Conference on Machine Learning*, 2019b.

J Neyman. Sur les applications de la theorie des probabilites aux experiences agricoles: essai des principes (masters thesis); justification of applications of the calculus of probabilities to the solutions of certain questions in agricultural experimentation. excerpts english translation (reprinted). *Stat Sci*, 5:463–472, 1923.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023, 2016.

Alex Nowak, Soledad Villar, Afonso S Bandeira, and Joan Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. In *Proceeding of the 34th International Conference on Machine Learning (ICML)*, volume 1050, pp. 22, 2017.

Peter Orbanz and Daniel M Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2014.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114, 2016.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Judea Pearl. *Causality*. Cambridge university press, 2009.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. Escape: Efficiently counting all 5-vertex subgraphs. In *Proceedings of the 26th International Conference on World Wide Web*, pp. 1431–1440, 2017.

JW Pitman. On coupling of markov chains. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 35(4):315–322, 1976.

James Gary Propp and David Bruce Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9(1-2):223–252, 1996.

Nataša Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177–e183, 2007.

Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 459–467, 2018.

Anant Raj, Stefan Bauer, Ashkan Soleymani, Michel Besserve, and Bernhard Schölkopf. Causal feature selection via orthogonal search. *arXiv preprint arXiv:2007.02938*, 2020.

Bastian Rieck, Christian Bock, and Karsten Borgwardt. A persistent weisfeiler-lehman procedure for graph classification. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5448–5458, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Ryan A Rossi, Nesreen K Ahmed, and Eunyee Koh. Higher-order network representation learning. In *Companion Proceedings of the The Web Conference 2018*, pp. 3–4, 2018.

Ryan A Rossi, Nesreen K Ahmed, Aldo Carranza, David Arbour, Anup Rao, Sungchul Kim, and Eunyee Koh. Heterogeneous network motifs. *arXiv preprint arXiv:1901.10026*, 2019.

Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of educational Psychology*, 66(5):688, 1974.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin A Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, 2018.

Adam Santoro, Felix Hill, David Barrett, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning*, pp. 4477–4486, 2018.

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

Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Random features strengthen graph neural networks. *arXiv preprint arXiv:2002.03155*, 2020.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*, 2019.

Bernhard Schölkopf. Causality for machine learning. *arXiv preprint arXiv:1911.10500*, 2019.

J Scott Armstrong and Fred Collopy. Causal forces: Structuring knowledge for time-series extrapolation. *Journal of Forecasting*, 12(2):103–115, 1993.

Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of escherichia coli. *Nature genetics*, 31(1):64–68, 2002.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Prateek Mittal, and Mung Chiang. Rogue signs: Deceiving traffic sign recognition with malicious ads and logos. *CoRR*, abs/1801.02780, 2018.

Olaf Sporns and Rolf Kötter. Motifs in brain networks. *PLoS biology*, 2(11):e369, 2004.

Lewi Stone and Alan Roberts. Competitive exclusion, or species aggregation? *Oecologia*, 91(3): 419–424, 1992.

Lewi Stone, Daniel Simberloff, and Yael Artzy-Randrup. Network motifs and their origins. *PLoS computational biology*, 15(4):e1006749, 2019.

Mahito Sugiyama, M. Elisabetta Ghisu, Felipe Llinares-López, and Karsten Borgwardt. graphkernels: R and python packages for graph comparison. *Bioinformatics*, 34(3):530–532, 2017.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4231–4242, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Carlos HC Teixeira, Leornado Cotta, Bruno Ribeiro, and Wagner Meira. Graph pattern mining and learning through user-defined relations. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 1266–1271. IEEE, 2018.

Komal K. Teru, Etienne Denis, and William L. Hamilton. Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2020.

Jin Tian and Judea Pearl. Causal discovery from changes. *UAI*, 2001.

Stanislaw M Ulam. A collection of mathematical problems. *Wiley, New York*, 29, 1960.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *ICLR*, 2018.

Clement Vignac, Andreas Loukas, and Pascal Frossard. Building powerful and equivariant graph neural networks with structural message-passing. *arXiv e-prints*, pp. arXiv–2006, 2020.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

Li Wang, Hongying Zhao, Jing Li, Yingqi Xu, Yujia Lan, Wenkang Yin, Xiaoqin Liu, Lei Yu, Shihua Lin, Michael Yifei Du, et al. Identifying functions and prognostic biomarkers of network motifs marked by diverse chromatin states in human cell lines. *Oncogene*, 39(3):677–689, 2020a.

Pinghui Wang, John CS Lui, Bruno Ribeiro, Don Towsley, Junzhou Zhao, and Xiaohong Guan. Efficiently estimating motif statistics of large networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(2):1–27, 2014.

Yiwei Wang, Wei Wang, Yuxuan Liang, Yujun Cai, and Bryan Hooi. Graphcrop: Subgraph cropping for graph classification. *arXiv preprint arXiv:2009.10564*, 2020b.

Van J Wedeen, Patric Hagmann, Wen-Yih Isaac Tseng, Timothy G Reese, and Robert M Weisskoff. Mapping complex tissue architecture with diffusion spectrum magnetic resonance imaging. *Magnetic resonance in medicine*, 54(6):1377–1386, 2005.

Boris Weisfeiler and AA Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. volume 80 of *Proceedings of Machine Learning Research*, pp. 5453–5462, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018b. PMLR.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. *arXiv preprint arXiv:2009.11848*, 2020.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.

Wei Ye, Omid Askarisichani, Alex Jones, and Ambuj Singh. Deepmap: Learning deep representations for graph classification. *arXiv preprint arXiv:2004.02131*, 2020.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. volume 97 of *Proceedings of Machine Learning Research*, pp. 7134–7143, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Wenchao Yu, Cheng Zheng, Wei Cheng, Charu C Aggarwal, Dongjin Song, Bo Zong, Haifeng Chen, and Wei Wang. Learning deep network representations with adversarially regularized autoencoders. In *Proc. of AAAI*, pp. 2663–2671. ACM, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 5165–5175, 2018.

A PROOF OF PROPOSITION 1

Proposition 1. *Let $P(Y|\mathcal{G}^{(obs)}_{N^{(obs)}} = G^{(obs)}_{n^{(obs)}})$ and $P(Y|\mathcal{G}^{(cf)}_{N^{(cf)}} = G^{(cf)}_{n^{(cf)}})$ be the conditional target distributions defined by the counterfactually-coupled random variables in Definition 1. For simplicity, assume $Y \in \mathcal{Y}$ is discrete. Consider a permutation-invariant graph representation $\Gamma : \cup_{n=1}^{\infty}\Omega^{n\times n} \to \mathbb{R}^d$, $d \geq 1$, and a link function $\rho(\cdot,\cdot)$ such that, for some $\epsilon, \delta > 0$, the generalization (interpolation) error is defined as*

$$P(|P(Y = y|\mathcal{G}^{(obs)}_{N^{(obs)}} = G^{(obs)}_{n^{(obs)}}) - \rho(y, \Gamma(G^{(obs)}_{n^{(obs)}}))| \leq \epsilon) \geq 1 - \delta\,, \quad \forall y \in \mathcal{Y},$$

and Γ is said **environment-invariant (E-invariant)** *if $\Gamma(\mathcal{G}^{(obs)}_{N^{(obs)}}) \overset{a.s.}{=} \Gamma(\mathcal{G}^{(cf)}_{N^{(cf)}})$, where a.s. (almost surely) means $\Gamma(G^{(obs)}_{n^{(obs)}}) = \Gamma(G^{(cf)}_{n^{(cf)}})$ for any graphs $G^{(obs)}_{n^{(obs)}}$ and $G^{(cf)}_{n^{(cf)}}$ that can be sampled.*
Then, the extrapolation error is

$$P(|P(Y = y|\mathcal{G}^{(cf)}_{N^{(cf)}} = G^{(cf)}_{n^{(cf)}}) - \rho(y, \Gamma(G^{(cf)}_{n^{(cf)}}))| \leq \epsilon) \geq 1 - \delta\,, \quad \forall y \in \mathcal{Y}. \tag{5}$$

Proof. By Equation (3), Y is only a function of W and some independent random noise, not E. Then, replacing E by $\tilde{E}$ in Definition 1 will not affect the distribution of Y, which yields $P(Y = y|\mathcal{G}^{(\text{obs})}_{N^{(\text{obs})}} = G^{(\text{obs})}_{n^{(\text{obs})}}) = P(Y = y|\mathcal{G}^{(\text{cf})}_{N^{(\text{obs})}} = G^{(\text{cf})}_{n^{(\text{obs})}})$. Since, by definition $\Gamma(G^{(\text{obs})}_{n^{(\text{obs})}}) = \Gamma(G^{(\text{cf})}_{n^{(\text{obs})}})$ for any two graphs $G^{(\text{obs})}_{n^{(\text{obs})}}$ and $G^{(\text{cf})}_{n^{(\text{cf})}}$ that can be sampled by our data generation process, we have that $\rho(y, \Gamma(G^{(\text{obs})}_{n^{(\text{obs})}})) = \rho(y, \Gamma(G^{(\text{cf})}_{n^{(\text{cf})}}))$, concluding our proof.

□

B PROOF OF THEOREM 1

Theorem 1. *Assume our graph-processing heuristic gives the number of vertices as $N^{(obs)} = \eta(E)$ and the outputs of g_{e_1} and g_{e_2} of Equation (1) can only differ in their attributes $\forall e_1, e_2$. Let $\overline{\mathcal{G}}^{(obs)}_n|W := \mathbb{E}_E[\overline{\mathcal{G}}^{(obs)}_n|W, N^{(obs)} = n, E]$ be the n-vertex graph output of our graph-processing heuristic over the true underlying data variable W. If $\overline{\mathcal{G}}^{(obs)}_n|W$ satisfies the following properties:*

1. *Deleting a random vertex n from $\overline{\mathcal{G}}^{(obs)}_n|W$, and the distribution of the trimmed graph is the same as the distribution of $\overline{\mathcal{G}}^{(obs)}_{n-1}|W$, with $\overline{\mathcal{G}}^{(obs)}_1|W$ as a trivial graph with a single vertex for all W.*

2. *For every $1 < k < n$, the subgraphs of $\overline{\mathcal{G}}^{(obs)}_n|W$ induced by $\{1, \ldots, k\}$ and $\{k+1, \ldots, n\}$ are independent random variables.*

Then, the variable W can be equivalently defined as $W = (W', C'_E)$, where W' is a random variable defined over the family of symmetric measurable functions $W' : [0,1]^2 \to [0,1]$, i.e., W' is a random graphon function, and, if the graph has attributes, C'_E is an environment-dependent random variable that defines vertex and edge attributes, otherwise, $C'_E = \emptyset$ is defined as the constant null.

Proof. First, a direct consequence of Equation (2) is that the distribution of $\overline{\mathcal{G}}^{(\text{obs})}_n|W$ is invariant under relabeling of the vertices (permutation invariance). We add this latter condition to conditions 1 and 2 of Theorem 1. Given these three conditions, Theorem 2.7 of Lovász & Szegedy (2006) states that $\mathcal{G}^{(\text{obs})}_n|W$'s graph topology is equivalent to that of the graphon model[2] $G(n, W')$ with $W' : [0,1]^2 \to [0,1]$ as a symmetric function. That is, we can redefine g_E of Equation (1) as g'_E such that the composition $\pi \circ \mathbb{E}_E[g'_E](W, Z_X)$ of Equations (1) and (2) is a graphon model. Since, the topology generated by g_e does not change with the environment e, the original (g_E) and the new graph generation processes (g'_E) would be indistinguishable for whatever distribution $P(E)$. For encoding the graph attributes into a single random variable, we simply need to define a random variable C' with an appropriate distribution $C' \sim P(C'|W')$ whose domain has the same cardinality as the graph attribute domain. If the graph has no attributes, we can define $P(C' = \emptyset|W) = 1$. □

[2]The graphon model was described as a W-random graph in Lovász & Szegedy (2006), with the notation later changing in the literature to match that of Diaconis & Freedman (1981), the first paper to describe the model.

C PROOF OF THEOREM 2

Theorem 2. *Let $\overline{\mathcal{G}}_n^{(obs)}|W$ and $\overline{\mathcal{G}}_{n'}^{(obs)}|W$ be two graphs of sizes n and n', respectively, satisfying Theorem 1. Note that n can be equal to n'. Let $\Gamma_{1\text{-}hot}(\overline{\mathcal{G}}_n^{(obs)}|W)$ be defined as in Equation (7) and $||\cdot||_\infty$ denote the L-infinity norm.. Then, for any integer $k \leq n$, $0 < \epsilon < 1$,*

$$\Pr(||\Gamma_{1\text{-}hot}(\overline{\mathcal{G}}_n^{(obs)}|W) - \Gamma_{1\text{-}hot}(\mathcal{G}_{n'}^{(obs)}|W)||_\infty > \epsilon) \leq 2|\mathcal{F}_{\leq k}|(\exp(-\frac{\epsilon^2}{8k^2}n) + \exp(-\frac{\epsilon^2}{8k^2}n')). \quad (8)$$

Proof. From Lovász & Szegedy (2006, Theorem 2.5), we know

$$\Pr(|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, W)| > \epsilon) \leq 2\exp(-\frac{\epsilon^2}{2k^2}n) \quad (11)$$

Since $|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, W)| \leq \frac{\epsilon}{2}$ and $|t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})}) - t(F_{k'}, W)| \leq \frac{\epsilon}{2}$ implies $|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})})| \leq \epsilon$.

$$\begin{aligned}
&\Pr(|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})})| > \epsilon) = 1 - \Pr(|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})})| \leq \epsilon) \\
&\leq 1 - \Pr(|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, W)| \leq \frac{\epsilon}{2}) \cdot \Pr(|t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})}) - t(F_{k'}, W)| \leq \frac{\epsilon}{2}) \\
&\leq 1 - (1 - 2\exp(-\frac{\epsilon^2}{8k^2}n))(1 - 2\exp(-\frac{\epsilon^2}{8k^2}n')) \\
&= 2(\exp(-\frac{\epsilon^2}{8k^2}n) + \exp(-\frac{\epsilon^2}{8k^2}n')) - 4\exp(-\frac{\epsilon^2}{8k^2}(n+n')) \\
&\leq 2(\exp(-\frac{\epsilon^2}{8k^2}n) + \exp(-\frac{\epsilon^2}{8k^2}n'))
\end{aligned} \quad (12)$$

Then we know

$$\begin{aligned}
\Pr(||\Gamma(\mathcal{G}_n^{(\text{obs})}) - \Gamma(\mathcal{G}_{n'}^{(\text{obs})})||_\infty \leq \epsilon) &= \Pr(|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})})| \leq \epsilon, \text{ for all } F_{k'} \in \mathcal{F}_{\leq k}) \\
&\geq 1 - \sum_{F_{k'} \in \mathcal{F}_{\leq k}} \Pr(|t(F_{k'}, \mathcal{G}_n^{(\text{obs})}) - t(F_{k'}, \mathcal{G}_{n'}^{(\text{obs})})| > \epsilon) \\
&\geq 1 - 2|\mathcal{F}_{\leq k}|(\exp(-\frac{\epsilon^2}{8k^2}n) + \exp(-\frac{\epsilon^2}{8k^2}n'))
\end{aligned} \quad (13)$$

It follows the Bonferroni inequality that, $\Pr(\cap_{i=1}^N A_i) \geq 1 - \sum_{i=1}^N \Pr(\tilde{\text{A}}_\text{i})$, where A_i and its complement $\tilde{A}_i$ are any events. Therefore, $\Pr(||\Gamma(\mathcal{G}_n^{(\text{obs})}) - \Gamma(\mathcal{G}_{n'}^{(\text{obs})})||_\infty > \epsilon) \leq 2|\mathcal{F}_{\leq k}|(\exp(-\frac{\epsilon^2}{8k^2}n) + \exp(-\frac{\epsilon^2}{8k^2}n'))$, concluding the proof.

□

D BIASES IN INDUCED HOMOMORPHISM DENSITIES

Let $\mathcal{C}_{\leq k}$ and $\mathcal{C}_k$ denote all possible connected k'-vetex graphs ($1 \leq k' \leq k$) and all possible connected k-vetex graphs respectively, C_k is an arbitrary k-vertex connected graph. Induced homomorphism densities over all possible k-vertex connected graph for an n-vertex graph G_n is defined as:

$$\omega(C_k, G_n) = \frac{\text{ind}(C_k, G_n)}{\sum_{C_k \in \mathcal{C}_k} \text{ind}(C_k, G_n)}$$

The $t(\cdot,\cdot)$ and $\mathcal{F}_{\leq k}$ are replaced by $\omega(\cdot,\cdot)$ and $\mathcal{C}_{\leq k}$ in Equations (7), (9) and (10) for graph representations in our experiments.

Achieving unbiased estimates for induced homomorphism densities usually requires sophisticated methods and enormous amount of time. We show that a biased estimator can also work for the GNN^+ in Equation (10) if the bias is multiplicative and the READOUT function is simply the sum of the node embeddings. We formalize it as followed.

Proposition 2. *Assume $\hat{\omega}(C_k, G_n)$ is a biased estimator for $\omega(C_k, G_n)$ for any k and k-sized connected graphs C_k in an n-vertex G_n, such that $\mathbb{E}(\hat{\omega}(C_k, G_n)) = \beta(C_k)\omega(C_k, G_n)$, where $\beta(C_k)$ ($\beta(\cdot) > 0$) is the bias related to the graph C_k, and the expectation is over the sampling procedure. The expected learned representation $\mathbb{E}(\sum_{C_{k'} \in \mathcal{C}_{\leq k}} \hat{\omega}(C_{k'}, G_n)\mathbf{1}^{\mathrm{T}}(\textit{GNN}^+(C_{k'})))$ can be the same as using the true induced homomorphism densities $\omega(\cdot, \cdot)$.*

Proof. If we can learn the representation $\text{GNN}^+(C_k) = \text{GNN}^+{}_0(C_k)/\beta(C_k)$ for all $C_{k'} \in \mathcal{C}_{\leq k}$, and $\text{GNN}^+{}_0$ is the representation we will learn from the true true induced homomorphism densities $\omega(\cdot, \cdot)$. This is possible because GNN^+ is proven to be a most expressive k-vertex graph representation, thus it is able to learn any function on the graph C_k. Then

$$\mathbb{E}\left[\sum_{C_{k'} \in \mathcal{C}_{\leq k}} \hat{\omega}(C_{k'}, G_n)\mathbf{1}^{\mathrm{T}}(\text{GNN}^+(C_{k'}))\right] = \sum_{C_{k'} \in \mathcal{C}_{\leq k}} \omega(C_{k'}, G_n)\mathbf{1}^{\mathrm{T}}(\text{GNN}^+{}_0(C_{k'})),$$

where $\mathbf{1}^{\mathrm{T}}(\text{GNN}^+(C_{k'}))$ is the sum of the node embeddings given by the GNN^+ if it is an equivariant representation of the graph. □

E Related work

This section provides a more in-depth discussion placing our work in the context of existing literature. We explain why existing state-of-the-art graph learning methods will struggle to extrapolate, subgraph methods, and explore perspectives of causality and extrapolation at large as well as in the context of graph classification.

Causal reasoning *Counterfactual inference and invariances.* Recent efforts have brought counterfactual inference to machine learning models. *Independence of causal mechanism (ICM)* methods (Bengio et al., 2019; Besserve et al., 2018; Johansson et al., 2016; Louizos et al., 2017; Raj et al., 2020; Schölkopf, 2019), *Causal Discovery from Change (CDC)* methods (Tian & Pearl, 2001), and *representation disentanglement* methods (Bengio et al., 2019; Goudet et al., 2017; Locatello et al., 2019). Invariant risk minimization (IRM) (Arjovsky et al., 2019) is a type of ICM (Schölkopf, 2019). Broadly, these efforts look for representations (or mechanism descriptions) that are invariant across multiple environments observed in the training data. In our work, we are interested in techniques that can work with a single training environment —a common case in graph data. Moreover, these works are not specifically designed for graphs, and it unclear how they can be efficiently adapted for graph tasks. To the best of our knowledge there is no clear effort for counterfactual graph extrapolations from a single environment.

Extrapolation There are other approaches for conferring models with extrapolation abilities. These ideas have started to permeate graph literature, which we touch on here, but remain outside the scope of our systematic counterfactual modeling framework.

Incorporating domain knowledge is an intuitive approach to learn a function that predicts adequately outside of the training distribution, data collection environment, and heuristic curation. This has been used, for example, in time series forecasting (Scott Armstrong & Collopy, 1993; Armstrong et al., 2005). This can come in the form of re-expressing phenomena in a way that can be adequately and accurately represented by machine learning methods (Lample & Charton, 2020) or specifically augmenting existing general-purpose methods to task (Klicpera et al., 2020). In the context of graphs, it has been used to pre-process the graph input to make a learned graph neural network model a less complex function and thus extend beyond training data (Xu et al., 2020), although this does not necessarily fall into the framework we consider here.

Another way of moving beyond the training data is *robustness*. Relevant for deep learning systems are adversarial attacks (Papernot et al., 2017). Neural networks can be highly successful classifiers

on the training data but become wildly inaccurate with small perturbations of those training examples (Goodfellow et al., 2015). This is important, say, in self-driving cars (Sitawarin et al., 2018), which can become confused by graffiti. This becomes particularly problematic when we deploy systems to real-world enviornments outside the training data. Learning to defend against adversarial attacks is in a way related to performing well outside the environment and curation heuristics encountered in training. An interesting possibility for future work is to explore the relationships between the two approaches.

We would like to point out that representation learning on dynamic graphs (Kazemi et al., 2020) is a separate vein of work from what we consider here. In these scenarios, there is a direct expectation that the process we model will change and evolve. For instance, knowledge bases – a form of graph encoding facts and relationships – are inevitably incomplete (Sun et al., 2018). Simply put, developments in information and society move faster than they can be curated. Another important example is recommendation systems (Kumar et al., 2019) based on evolving user-item networks. These also relate to the counterfactuals on graphs (Eckles et al., 2016a). This is fundamentally different from our work where we do graph-wide learning and representation of a dataset of many graphs rather than one constantly evolving graph.

Overfitting will compromise even generalization (interpolation). Regularization schemes such as explicit penalization are a well known and broadly applicable strategy (Hastie et al., 2012). Another implicit approach is data augmentation (Hernández-García & König, 2018), and the recent GraphCrop method proposes a scheme for graphs that randomly extracts subgraphs from certain graphs in a minibatch during training (Wang et al., 2020b). These directions differ from our own in that we seek a formulation for extrapolation even when overfitting is not necessarily a problem but the two approaches are both useful in the toolbox of an analyst.

Subraph methods and Graphlet Counting Kernels A foundational principle here is that exploiting subgraphs confers graph classifications models with both the ability to fit the training data and extrapolate to graphs generated from a different environment. As detailed in Section 3, this insight follows from the Aldous-Hoover representation exchangeable distributions over graphs (Hoover, 1979; Aldous, 1981; Kallenberg, 2006; Orbanz & Roy, 2014) and work on graph limits (Lovász, 2012). We discuss the large literature using subgraphs in machine learning.

Counting kernels (Shervashidze et al., 2009) measure the similarity between two graphs by the dot product of their normalized counts of connected induced subgraphs (graphlet). This can be used for classification via kernalized methods like Support Vector Machines (SVM). Yanardag & Vishwanathan (2015) argue that the dot product does not capture dependence between subgraphs and extend to a general bilinear form over a learned similarity matrix. These approaches are related to the Reconstruction Conjecture, which posits graphs can be determined through knowledge of their subgraphs (Kelly et al., 1957; Ulam, 1960; Hemminger, 1969; McKay, 1997). It is known that computing a maximally expressive graph kernel, or one that is injective over the class of graphs, is as hard as the Graph Isomorphism problem, and thus intractible in general (Gärtner et al., 2003; Kriege et al., 2020). Kriege et al. (2018) demonstrate graph properties that subgraph counting kernels fail to capture and propose a method to make them more expressive, but only for graphs without vertex attributes. Most applications of graphlet counting do not exploit vertex attributes, and even those that do (e.g. (Wale et al., 2008)) are likely to fail under a distribution shift over attributes; recording a count for each type of attributed subgraph (e.g. red clique, blue clique) is sensitive to distribution shift. In comparison, our use of Relational Pooling Graph Neural Networks confers our framework with the ability learn a compressed representation of different attributed subgraphs, tailored for the task, and extrapolate even under attribute shift. We demonstrate this in synthetic experiments below. Last, a recent work of Ye et al. (2020) propose to pass the attributed subgraph counts to a downstream neural network model to better compress and represent the high dimensional feature space. However, with extreme attribute shift, it may be that the downstream layers did not see certain attributed subgraph types in training enough to learn how to correctly represent them. We feel that it is better to *compress* the attributed signal *in the process of* representing the graph to handle these vertex features, the approach we take here.

There are many graph kernel methods that do not leverage subgraph counts but other features to measure graph similarity, such as the count of matching walks, e.g. (Kashima et al., 2003; Borgwardt et al., 2005; Borgwardt & Kriegel, 2005). The WL Kernel uses the WL algorithm to compare

graphs (Shervashidze et al., 2011) and will inherit the limitations of WL GNNs like inability to represent cycles. Rieck et al. (2019) propose a persistent WL kernel that uses ideas from Topological Data Analysis (Munch, 2017) to better capture such structures when comparing graphs. Methods that do not count subgraphs will not inherit properties regarding a graph-size environment change – from our analysis of asymptotic graph theory – but all extrapolation tasks require an assumption and our framework can be applied to studying the ability of various kernel methods to extrapolate under different scenarios. Those relying on attributes to build similarities are also likely to suffer from attribute shift.

Subgraphs are studied to understand underlying mechanisms of graphs like gene regulatory networks, food webs, and the vulnerability of networks to attack, and sometimes used prognostically. A popular example investigates *motifs*, subgraphs that appear more frequently than under chance (Stone & Roberts, 1992; Shen-Orr et al., 2002; Milo et al., 2002; Mangan & Alon, 2003; Sporns & Kötter, 2004; Bascompte & Melián, 2005; Alon, 2007; Chen et al., 2013; Benson et al., 2016; Stone et al., 2019; Dey et al., 2019; Wang et al., 2020a). Although the study of motifs is along a different direction and often focus on one-graph datasets, our framework learns rich latent representations of subgraphs; interesting future work could include leveraging our learned $\overline{\overline{f}}$ functions (pre-trained) and estimated counts to glean scientific understanding. Another line of work uses subgraph counts as graph similarity measures, an example being matching real-world graphs to their most similar random graph generation models (Pržulj, 2007).

Other machine learning methods based on subgraphs have also been proposed. Methods like mGCMN (Li et al., 2020), HONE (Rossi et al., 2018), and MCN (Lee et al., 2018) learn representations for vertices by extending classical methods over edges to a new neighborhood structure based on subgraphs; for instance, mGCMN runs a GNN on the new graph. These methods do not exploit all subgraphs of size k and will not learn subgraph representations in a manner consistent with our extrapolation framework. Teru et al. (2020) use subgraphs around vertices to predict missing facts in a knowledge base. Further examples include the Subgraph Prediction Neural network (Meng et al., 2018) that predicts subgraph classes in one dynamic heterogeneous graph; counting the appearance of edges in each type of subgraph for link prediction tasks (AbuOda et al., 2019); and SEAL(Zhang & Chen, 2018) runs a GNN over subgraphs extracted around candidate edges to predict whether an edge exists. While these methods exploit small subgraphs for their effective balance between rich graph information and computational tractibility, they are along an orthogonal thread of work.

Graph Neural Networks Among the many approaches for graph representation learning and classification, which include methods for vertex embeddings that are subsequently read-out into graph represenations (Belkin & Niyogi, 2002; Perozzi et al., 2014; Niepert et al., 2016; Ou et al., 2016; Kipf & Welling, 2016; Grover & Leskovec, 2016; Yu et al., 2018; Qiu et al., 2018; Maron et al., 2019b;a; Wu et al., 2020; Hamilton, 2020; Chami et al., 2020) we focused our discussion and modeling of $\overline{\overline{f}}$ on graph neural network methods (Kipf & Welling, 2017; Atwood & Towsley, 2016; Hamilton et al., 2017; Gilmer et al., 2017; Velickovic et al., 2018; Xu et al., 2018a; Morris et al., 2019; You et al., 2019; Liu et al., 2019; Chami et al., 2019). GNNs are trained end-to-end, can straightforwardly provide latent subgraph representations, easily handle vertex/edge attributes, are computationally efficient, and constitute a state-of-the-art method. However, GNNs lack extrapolation capabilities due to their inability to learn latent representations that capture the topological structure of the graph (Xu et al., 2018a; Morris et al., 2019; Garg et al., 2020; Sato, 2020). Relevantly, many cannot count the number of subgraphs such as triangles (3-cliques) in a graph (Arvind et al., 2020; Chen et al., 2020). In general, our theory of extrapolating in graph tasks requires properly capturing graph structure. Moreover, if GNNs cannot exploit structure in subgraphs they may be distracted by vertex features and fail to extrapolate under attribute shift, as demonstrated in our experiments. Relational Pooling (Murphy et al., 2019b) and rGIN (Sato et al., 2020) employ random features as a straightforward way to overcome this limitation; whereas rGIN does not respect isomorphic invariance of graphs, we compare against RP-GNN. There, we show that state-of-the-art GIN (Xu et al., 2018a) and RP-GIN (Murphy et al., 2019b) are expressive in-distribution but fail to extrapolate.

Teru et al. (2020) use subgraphs around vertices to predict missing facts in a knowledge base. Further examples include Meng et al. (2018) that predicts subgraph classes in one dynamic heterogeneous graph; counting the appearance of edges in each type of subgraph for link prediction tasks (AbuOda et al., 2019); and SEAL(Zhang & Chen, 2018) runs a GNN over subgraphs extracted around candi-

date edges to predict whether an edge exists. While these methods exploit small subgraphs for their effective balance between rich graph information and computational tractability.

F EXPERIMENTS

In this appendix we present the details of the experimental section, discussing the hyperparameters that have been tuned. Note that the search space has been chosen so that all the biggest models have a comparable number of parameters.

F.1 MODEL IMPLEMENTATION

All neural network approaches, including the models proposed in this paper, are implemented in PyTorch (Paszke et al., 2019).

Our GIN (Xu et al., 2018a) implementation is based on the implementation available in Pytorch Geometric (Fey & Lenssen, 2019). For RPGIN (Murphy et al., 2019b), we implement the permutation and concatenation with one-hot identifiers and use GIN as before. To use GIN on unattributed graphs, we follow convention and assign a '1' dummy feature on every vertex. For RPGIN, we assign one-hot identifiers with dimension 10. In the attributed case, GIN simply uses the vertex attributes whereas RPGIN appends one-hot identifiers to the attributes. GIN and RPGIN serve both as baselines and as architectural building-blocks of $\Gamma^{(\text{eq. } 9)}_{\text{GIN}}$and $\Gamma^{(\text{eq. } 10)}_{\text{RPGIN}}$for learning latent vectors of connected induced subgraphs. Other than a few hyperparameters and architectural choices, we use standard choices (e.g. Hu et al. (2020)).

We use the WL graph kernel implementations provided by the *graphkernels* package (Sugiyama et al., 2017). All kernel methods use a Support Vector Machine on scikit-learn (Pedregosa et al., 2011).

The graphlet counting kernel, as well as our own procedure, relies on being able to efficiently count attributed or unattributed connected induced subgraphs. We made use of ESCAPE (Pinar et al., 2017) and R-GPM (Teixeira et al., 2018) as described in the main text. The source code of ESCAPE is available online and the authors of Teixeira et al. (2018) provided us their code.

Our models ($\Gamma^{(\text{eq. } 7)}_{\text{1-hot}}$, $\Gamma^{(\text{eq. } 9)}_{\text{GIN}}$, $\Gamma^{(\text{eq. } 10)}_{\text{RPGIN}}$) were implemented using PyTorch Geometric (Fey & Lenssen, 2019). As discussed in the main text, the choice of subgraph size k is very important hyperparameter for our method, trading off computation, expressive power, and in a way that depends on the characteristics of the graphs at hand. We discuss our choice of k in each of the tasks below, and the same samplers as above to obtain exact or estimated induced subgraph densities.

These models learn graph representations $\Gamma(G)$, which we pass to downstream layers in an end-to-end fashion. For $\Gamma^{(\text{eq. } 9)}_{\text{GIN}}$, and $\Gamma^{(\text{eq. } 10)}_{\text{RPGIN}}$, we use GIN and RP-GIN respectively to obtain latent vectors for each k-sized Connectected Induced Subgraph (CIS) and then sum over the latent CIS representations, each weighted by its corresponding induced homomorphism density. In Appendix F.4, we use an attention mechanism in the sum. For $\Gamma^{(\text{eq. } 7)}_{\text{1-hot}}$, the representation $\Gamma_{\text{one-hot}}(G)$ is a vector containing densities of each (possibly attributed) CIS pattern. To map this into a graph representation, we compute $\Gamma_{\text{one-hot}}(G)^{\mathrm{T}}\boldsymbol{W}$ where $\boldsymbol{W}$ is a learnable weight whose columns. Note that this effectively learns a unique weight vector for each CIS pattern.

The methods GIN, RPGIN, $\Gamma^{(\text{eq. } 9)}_{\text{GIN}}$, $\Gamma^{(\text{eq. } 10)}_{\text{RPGIN}}$, and $\Gamma^{(\text{eq. } 7)}_{\text{1-hot}}$ all produce a latent graph representation vector for each graph. In each case, we use a linear layer on the graph representation to obtain the prediction. To optimize the neural models, we use Adam optimizer. When an in-distribution validation set is available (see below), we use the weights that achieve best validation-set performance for prediction. Otherwise, we train for a fixed number of epochs.

The specifics of hyperparameter grids and downstream architectures are discussed in each section below.

F.2 SCHIZOPHRENIA TASK: SIZE EXTRAPOLATION

These data were provided by the gracious authors of De Domenico et al. (2016), which they pre-processed from publicly available data from The Center for Biomedical Research Excellence[3]. There are 145 graphs which represent the functional connectivity brain networks of 71 schizophrenic patients and 74 healthy controls. Each graph has 264 vertices representing spherical regions of interest (ROIs). Edges represent functional connectivity. Originally, edges reflected a time-series coherence between regions. If the coherence between signals from two regions was above a certain threshold, the authors created a weighted edge. Otherwise, there is no edge. For simplicity, we converted these to un-weighted edges. A key motivation of this paper shares our own. Extensive pre-processing must be done over fMRI data to create brain graphs. This includes discarding signals from certain ROIs. As described by the authors, these choices make highly significant impacts on the resulting graph. We refer the reader to the paper (De Domenico et al., 2016). It is interesting to note that there are numerous methods for constructing a brain graph, and in ways that change the number of vertices. The measurement strategy taken by the lab can result in measuring about 500 ROIs, 1000 ROIs, or 264 as in the case of this paper (Hagmann et al., 2007; Wedeen et al., 2005; De Domenico et al., 2016).

For our purposes, we wish to create an extrapolation task where a change in environment leads to an extrapolation set that contains smaller graphs. We randomly select 20 of the 145 graphs, balanced among the healthy and schizophrenic patients, and we reduce the size of the control-group graphs by removing nodes uniformly at random. Ultimately these graphs have on average 40% fewer vertices. This forms our extrapolation-test set.

We hold out the extrapolation-test. Over the remaining data, we use 5-fold cross-validation to assess interpolation-test accuracy and for hyperparameter tuning. Each of the validation-set folds can be used as interpolation-test sets. We averaged over the validation fold performance of the best-performing hyperparameter configuration and report the mean (standard deviation) as interpolation-test performance. Note that we use stratified sampling within the cross validation folds.

Recall that we must obtain homomorphism densities for $\Gamma_{\text{GIN}}^{(\text{eq. 9})}$, $\Gamma_{\text{RPGIN}}^{(\text{eq. 10})}$, $\Gamma_{\text{1-hot}}^{(\text{eq. 7})}$, and the graphlet counting kernel. We use ESCAPE, to and tune the size in $\{4, 5\}$. Finally, in this section, all GIN modules use the Jumping Knowledge mechanism (Xu et al., 2018b).

For $\Gamma_{\text{GIN}}^{(\text{eq. 9})}$and $\Gamma_{\text{RPGIN}}^{(\text{eq. 10})}$, we tune the network width (of the aggregation MLP) in $\{32, 64, 128, 256\}$ and the number of layers (i.e. recursions of message-passing) in $\{1, 2\}$, the learning rate in $\{0.001, 0.0001\}$, the batch size in $\{32, 64, \text{full-train-size}\}$,

For $\Gamma_{\text{1-hot}}^{(\text{eq. 7})}$, recall that we learn a unique weight vector for each CIS type; we tune the dimension of this vector in $\{32, 64, 128, 256\}$. We tune the learning rate in $\{0.001, 0.0001\}$, and the batch size in $\{32, 64, \text{full-train}\}$.

For the baseline classifiers GIN and RPGIN, we tune the learning rate in $\{0.01, 0.001\}$, the network width in $\{32, 64, 128, 256\}$, the number of layers in $\{1, 2, 3, 4\}$, and the batch size in $\{32, \text{full-train}\}$. For inference, we average over four permutations as described in (Murphy et al., 2019b).

We train all neural models for 400 epochs. Once the best hyperparameter configuration is obtained through cross-validation on the training data with full-sized graphs, we re-train the model on the entire train split before predicting on the extrapolation set where the healthy graphs are smaller. We repeat the training with 10 different initialization seeds, and we report the mean and the standard deviation.

For the graph kernels, following Kriege et al. (2020), we tune the regularization hyperparameter C in SVM over the range from 10^{-3} to 10^4 with steps of 10. We tune the number of iterations for WL kernel in $\{1, 2, 3, 4\}$.

F.3 ERDŐS-RÉNYI CONNECTION PROBABILITY: SIZE EXTRAPOLATION

We simulated Erdős-Rényi graphs (Gnp model) using NetworkX (Hagberg et al., 2008). The graphs in training and interpolation-test varies from $\{20, \ldots, 80\}$, while extrapolation-test graphs vary from

[3] `http://fcon_1000.projects.nitrc.org/indi/retro/cobre`

Table 3: Average (standard deviation) number of five cliques with varying colorations in a graph across training, interpolation test and extrapolation test. The target is the number of 5-cliques without any green vertices, the sum of the clique-types indicated in the first two row headings.

	Train	Interpolation Test	Extrapolation Test
No green, 4 or 5 red	8.55 (10.87)	10.55 (11.82)	3.10 (6.50)
No green, less than 4 red	2.02 (3.11)	1.35 (3.05)	8.56 (9.66)
At least one green	6.79 (8.60)	9.05 (11.06)	7.68 (11.94)
Total number of 5-cliques	17.36 (13.68)	20.95 (13.96)	19.35 (17.07)

$\{140, \ldots, 200\}$, selected uniformly at random. Here, the training, in-environment interpolation-test, and extrapolation-test sets are fixed. They are of sizes 80, 40, and 100 respectively.

Recall that we must obtain homomorphism densities for $\Gamma_{\text{GIN}}^{\text{(eq. 9)}}$, $\Gamma_{\text{RPGIN}}^{\text{(eq. 10)}}$, $\Gamma_{\text{1-hot}}^{\text{(eq. 7)}}$, and the graphlet counting kernel. We use ESCAPE for a fixed size $k = 5$.

For $\Gamma_{\text{GIN}}^{\text{(eq. 9)}}$, and $\Gamma_{\text{RPGIN}}^{\text{(eq. 10)}}$, we tune the network width of the aggregator MLP in $\{16, 32, 64, 128, 256\}$, the number of layers (i.e. recursions of message passing) in $\{1, 2\}$, and the learning rate in $\{0.1, 0.01, 0.001\}$.

For $\Gamma_{\text{1-hot}}^{\text{(eq. 7)}}$, recall that we learn a unique weight vector for each CIS type; we tune the dimension of this vector in $\{16, 32, 64, 128, 256\}$ and the learning rate in $\{0.1, 0.01, 0.001\}$.

For the baseline classifiers GIN and RPGIN, we tune the network width of the MLP aggregator in $\{32, 64, 128, 256\}$, the number of layers (i.e. message passing recursions) in $\{1, 2, 3\}$, and the learning rate in $\{0.1, 0.01, 0.001\}$. We also tune the presence or absence of the Jumping Knowledge mechanism from Xu et al. (2018b).

We train all neural models for 500 epochs. Whenever we evaluate model performance, we do so using the estimated weights from the epoch that attained the best performance on the interpolation-test (i.e. validation) set. We select the hyperparameter configuration that achieved the highest mean accuracy on the interpolation-test, averaged across 10 different random weight initializations. We report the score from the best hyperparameter configuration as interpolation-test performance; we do so for all neural methods. To report training and extrapolation-test set performance, we train the model with best hyperparameter configuration, again using the interpolation-test (validation) set to find the epoch at which to use weights, and predict over both sets. This is also repeated for 10 random initializations.

For the graph kernels, following Kriege et al. (2020), we tune the regularization hyperparameter C in SVM from 10^{-3} to 10^4 with steps of 10 and the number of iterations for WL kernel in $\{2, 3\}$.

F.4 Extrapolation performance over attributed graphs

This experiment involves vertex attributes. To simulate graphs, we first randomly create an unattributed graph (i.e. simulate a graph of some topology), and then add vertex attributes. The graph structure is sampled from a Erdős-Rényi (Gnp) model whose number of vertices is selected uniformly at random from $\{20, 21, 22, \ldots, 25\}$. Since our task involves counting 5-cliques (those that have no green vertices), we were careful to specify an edge-creation probability that would create a meaningful number of 5-cliques. In particular, after sampling a graph size, we compute the edge probability such that the expected proportion of 5-cliques is 0.8. For example, the expected number of 5-cliques is 16 for a graph size 20.

The vertices have a one attribute: either red, green, or blue, which is one hot-encoded. As discussed on the main text, we induce an attribute-shift environment change. For training and interpolation-test, the 5-cliques are predominantly red whereas in extrapolation-test, the coloration of 5-cliques is more 'uniform'. The empirical distributions of 5-clique coloration is shown in Table 3. By 'uniform', we do not indicate that the proportion of the types of 5-cliques shown in the extrapolation-test column is uniform. We mean that the coloration is less dominated by mostly-red 5-cliques. The idea behind this task is that it is very challenging in that the target is highly correlated with 'number of mostly-red cliques'. Mostly need to avoid learning to predict the number of red 5-cliques (if they

can count substructures effectively, unlike the WL and GNN methods), and avoid learning some function of the number of red vertices.

Train, Interpolation Test and Extrapolation Test sets are fixed and each respectively contain 80, 20, and 100 graphs.

As these are attributed graphs, we estimate the CIS counts with R-GPM as discussed in the main text. We fixed the subgraph size to $k = 5$. A round of sampling is run for every epoch that we use in training to ensure unbiased estimation.

For $\Gamma_{\text{GIN}}^{(\text{eq. 9})}$, and $\Gamma_{\text{RPGIN}}^{(\text{eq. 10})}$, we use GIN with no jumping knowledge to obtain the CIS representations. Then, a graph representation is constructed by employing the attention mechanism proposed in Ilse et al. (2018) on the CIS representations concatenated to their densities. We tune the width of the MLP in the aggregator in $\{16, 32, 64, 128, 256\}$, the number of layers (i.e. message passing recursions) in $\{1, 2\}$, and the learning rate in $\{0.01, 0.001\}$,

For $\Gamma_{\text{1-hot}}^{(\text{eq. 7})}$, recall that we learn a unique weight vector for each CIS type; we tune the dimension of this vector in $\{8, 16, 32\}$ and the number of layers in $\{1, 2\}$. To avoid overfitting due to the large number of distinct attributed CISs which translates in a large number of parameters, we used an l2 regularization of 0.1.

For standard GIN and RPGIN, we tune the network width of the MLP aggregator in $\{32, 64, 128, 256\}$, the number of layers (i.e. message passing recursions) in $\{1, 2, 3\}$, the learning rate in $\{0.01, 0.001\}$ and dropout in $\{0.0, 0.1\}$,

The models are trained for 400 epochs. The hyperparameter tuning and evaluation scheme are similar that of Appendix F.3.

For the graph kernels, following Kriege et al. (2020), we tune the regularization hyperparameter C in SVM from 10^{-3} to 10^4 with steps of 10 and the number of iterations for WL kernel in $\{2, 3\}$.

G EDGE ATTRIBUTES

To the best of our knowledge, no sampling mechanism handles edge attributes due to the lack of canonical labeling algorithms that consider them. To fill the gap, we propose to modify the canonical labeling algorithms used in the sampling algorithms. Our idea generalizes the method proposed in McKay & Piperno (2014), and consists of transforming a graph with edge and vertex attributes into a larger graph with only vertex attributes that can then be used in the canonical labeling. If the edge attributes are integers in $\{1, 2, \ldots, 2^{d-1}\}$, the transformed graph will have d layers, each with the original number of vertices. The binary expansion of each color number tells us which layers contain edges with that colors. The new attribute for each vertex is an hash of its original attribute and the layer number.