Relational Pooling for Graph Representations

Ryan L. Murphy [1] **Balasubramaniam Srinivasan** [2] **Vinayak Rao** [1] **Bruno Ribeiro** [2]

Abstract

This work generalizes graph neural networks (GNNs) beyond those based on the Weisfeiler-Lehman (WL) algorithm, graph Laplacians, and diffusions. Our approach, denoted Relational Pooling (RP), draws from the theory of finite partial exchangeability to provide a framework with maximal representation power for graphs. RP can work with existing graph representation models and, somewhat counterintuitively, can make them even more powerful than the original WL isomorphism test. Additionally, RP allows architectures like Recurrent Neural Networks and Convolutional Neural Networks to be used in a theoretically sound approach for graph classification. We demonstrate improved performance of RP-based graph representations over state-of-the-art methods on a number of tasks.

1. Introduction

Applications with relational graph data, such as molecule classification, social and biological network prediction, first order logic, and natural language understanding, require an effective representation of graph structures and their attributes. While representation learning for graph data has made tremendous progress in recent years, current schemes are unable to produce so-called *most-powerful* representations that can provably distinguish all distinct graphs up to graph isomorphisms. Consider for instance the broad class of Weisfeiler-Lehman (WL) based Graph Neural Networks (WL-GNNs) (Duvenaud et al., 2015; Kipf & Welling, 2017; Gilmer et al., 2017; Hamilton et al., 2017a; Velickovic et al., 2018; Monti et al., 2017; Ying et al., 2018; Xu et al., 2019; Morris et al., 2019). These are unable to distinguish pairs of nonisomorphic graphs on which the standard WL isomorphism heuristic fails (Cai et al., 1992; Xu et al., 2019; Morris et al., 2019). As graph neural networks (GNNs) are applied to increasingly more challenging problems, having a most-powerful framework for graph representation learning would be a key development in geometric deep learning (Bronstein et al., 2017).

In this work we introduce *Relational Pooling* (RP), a novel framework with maximal representation power for any graph input. In RP, we specify an idealized most-powerful representation for graphs and a framework for tractably approximating this ideal. The ideal representation can distinguish pairs of nonisomorphic graphs even when the WL isomorphism test fails, which motivates a straightforward procedure using approximate RP – we call this *RP-GNN* – for making GNNs more powerful.

A key inductive bias for graph representations is invariance to permutations of the adjacency matrix (graph isomorphisms), see Aldous (1981); Diaconis & Janson (2008); Orbanz & Roy (2015). Our work differs in its focus on learning representations of *finite but variable-size* graphs. In particular, given a finite but arbitrary-sized graph G potentially endowed with vertex or edge features, RP outputs a representation $\overline{\overline{f}}(G) \in \mathbb{R}^{d_h}$, $d_h > 0$, that is invariant to graph isomorphisms. RP can learn representations for each vertex in a graph, though to simplify the exposition, we focus on learning one representation of the entire graph.

Contributions. We make the following contributions: (1) We introduce *Relational Pooling* (RP), a novel framework for graph representation that can be combined with any existing neural network architecture, including ones not generally associated with graphs such as Recurrent Neural Networks (RNNs). (2) We prove that RP has maximal representation power for graphs and show that combining WL-GNNs with RP can increase their representation power. In our experiments, we classify graphs that cannot be distinguished by a state-of-the-art WL-GNN (Xu et al., 2019). (3) We introduce approximation approaches that make RP computationally tractable. We demonstrate empirically that these still lead to strong performance and can be used with RP-GNN to speed up graph classification when compared to traditional WL-GNNs.

2. Relational Pooling

Notation. We consider graphs endowed with vertex and edge features. That is, let $G = (V, E, \boldsymbol{X}^{(v)}, \mathbf{X}^{(e)})$ be a graph with vertices V, edges $E \subseteq V \times V$, vertex fea-

[1]Department of Statistics, and [2]Department of Computer Science, Purdue University, West Lafayette, Indiana, USA. Correspondence to: Ryan L. Murphy <murph213@purdue.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

tures stored in a $|V| \times d_v$ matrix $\boldsymbol{X}^{(v)}$, and edge features stored in a $|V| \times |V| \times d_e$ tensor $\mathbf{X}^{(e)}$. W.l.o.g, we let $V := \{1, \ldots, n\}$, choosing some arbitrary ordering of the vertices. Unlike the vertex features $\boldsymbol{X}^{(v)}$, these vertex *labels* do not represent any meaningful information about the vertices, and learned graph representations should not depend upon the choice of ordering. Formally, there always exists a bijection on V (called a permutation or isomorphism) between orderings so we desire *permutation-invariant*, or equivalently, *isomorphic-invariant* functions.

In this work, we encode G by two data structures: (1) a $|V| \times |V| \times (1 + d_e)$ tensor that combines G's adjacency matrix with its edge features and (2) a $|V| \times d_v$ matrix representing node features $\boldsymbol{X}^{(v)}$. The tensor is defined as $\mathbf{A}_{v,u,\cdot} = \left[\mathbb{1}_{(v,u)\in E} \bowtie \boldsymbol{X}^{(e)}_{v,u}\right]$ for $v, u \in V$ where $[\cdot \bowtie \cdot]$ denotes concatenation along the 3rd mode of the tensor, $\mathbb{1}_{(\cdot)}$ denotes the indicator function, and $\boldsymbol{X}^{(e)}_{v,u}$ denotes the feature vector of edge (v, u) by a slight abuse of notation. A permutation is bijection $\pi : V \to V$ from the label set V to itself. If vertices are relabeled by a permutation π, we represent the new adjacency tensor by $\mathbf{A}_{\pi,\pi}$, where $(\mathbf{A}_{\pi,\pi})_{\pi(i),\pi(j),k} = \boldsymbol{A}_{i,j,k}\ \forall i, j \in V,\ k \in \{1, \ldots, 1 + d_e\}$; the index k over edge features is not permuted. Similarly, the vertex features are represented by $\boldsymbol{X}^{(v)}_{\pi}$ where $(\boldsymbol{X}^{(v)}_{\pi})_{\pi(i),l} = X^{(v)}_{i,l},\ \forall i \in V$ and $l \in \{1, \ldots, d_v\}$. The Supplementary Material shows a concrete example and Kearnes et al. (2016) use a similar representation.

For bipartite graphs (e.g., consumers $\times$ products), V is partitioned by $V^{(\mathrm{r})}$ and $V^{(\mathrm{c})}$ and a separate permutation function can be defined on each. Their encoding is similar to the above and we define RP for the two different cases below.

Joint RP. Inspired by joint exchangeability (Aldous, 1981; Diaconis & Janson, 2008; Orbanz & Roy, 2015), we define a *joint RP* permutation-invariant function of non-bipartite graphs, whether directed or undirected, as

$$\overline{\overrightarrow{f}}(G) = \frac{1}{|V|!} \sum_{\pi \in \Pi_{|V|}} \overrightarrow{f}(\mathbf{A}_{\pi,\pi}, \boldsymbol{X}^{(v)}_{\pi}), \tag{1}$$

where $\Pi_{|V|}$ is the set of all distinct permutations of V and $\overrightarrow{f}$ is an arbitrary (possibly *permutation-sensitive*) function of the graph with codomain $\mathbb{R}^{d_h}$. Following Murphy et al. (2019), we use the notation $\overline{\cdot}$ to denote permutation-invariant function. Since Equation 1 averages over all permutations of the labels V, $\overline{\overrightarrow{f}}$ is a permutation-invariant function and can theoretically represent any such function $\overline{\overrightarrow{g}}$ (consider $\overrightarrow{f} = \overrightarrow{g}$). We can compose $\overline{\overrightarrow{f}}$ with another function ρ (outside the summation) to capture additional signal in the graph. This can give a maximally expressive, albeit intractable, graph representation (Theorem 2.1). We later discuss tractable approximations for $\overline{\overrightarrow{f}}$ and neural network architectures for $\overrightarrow{f}$.

Separate RP. RP for bipartite graphs is motivated by *separate* exchangeability (Diaconis & Janson, 2008; Orbanz & Roy, 2015) and is defined as

$$\overline{\overrightarrow{f}}(G) = C \sum_{\pi \in \Pi_{|V^{(\mathrm{r})}|}} \sum_{\sigma \in \Pi_{|V^{(\mathrm{c})}|}} \overrightarrow{f}\big(\mathbf{A}_{\pi,\sigma}, \boldsymbol{X}^{(\mathrm{r},v)}_{\pi}, \boldsymbol{X}^{(\mathrm{c},v)}_{\sigma}\big) \tag{2}$$

where $C = (|V^{(\mathrm{r})}|!|V^{(\mathrm{c})}|!)^{-1}$ and π, σ are permutations of $V^{(\mathrm{r})}$, $V^{(\mathrm{c})}$, respectively. Results that apply to joint RP apply to separate RP.

2.1. Representation Power of RP

Functions $\overline{\overrightarrow{f}}$ should be *expressive* enough to learn distinct representations of nonisomorphic graphs or graphs with distinct features. We say $\overline{\overrightarrow{f}}(G)$ is *most-powerful* or *most-expressive* when $\overline{\overrightarrow{f}}(G) = \overline{\overrightarrow{f}}(G')$ iff G and G' are isomorphic and have the same vertex/edge features up to permutation. If $\overline{\overrightarrow{f}}$ is not most-powerful, a downstream function ρ may struggle to predict different classes for nonisormorphic graphs.

Theorem 2.1. *If node and edge attributes come from a finite set, then the representation $\overline{\overrightarrow{f}}(G)$ in Equation 1 is the most expressive representation of G, provided $\overrightarrow{f}$ is sufficiently expressive (e.g., a universal approximator).*

All proofs are shown in the Supplementary Material. This result provides a key insight into RP; one can focus on building expressive functions $\overrightarrow{f}$ that need not be permutation-invariant as the summation over permutations assures that permutation-invariance is satisfied.

2.2. Neural Network Architectures

Since $\overrightarrow{f}$ may be permutation sensitive, RP allows one to use a wide range of neural network architectures.

RNNs, MLPs. A valid architecture is to vectorize the graph (concatenating node and edge features, as illustrated in the Supplementary Material) and learn $\overrightarrow{f}$ over the resulting sequence. $\overrightarrow{f}$ can be an RNN, like an LSTM (Hochreiter & Schmidhuber, 1997) or GRU (Cho et al., 2014), or a feed-forward neural network (multilayer perceptron, MLP) with padding if different graphs have different sizes. Concretely,

$$\overline{\overrightarrow{f}}(G) = \frac{1}{|V|!} \sum_{\pi \in \Pi_{|V|}} \overrightarrow{f}\big(\mathrm{vec}(\mathbf{A}_{\pi,\pi}, \boldsymbol{X}^{(v)}_{\pi})\big).$$

CNNs. Convolutional neural networks (CNNs) can also be directly applied over the tensor $\mathbf{A}_{\pi,\pi}$ and combined with the node features $\boldsymbol{X}^{(v)}_{\pi}$, as in

$$\overline{\overrightarrow{f}}(G) = \frac{1}{|V|!} \sum_{\pi \in \Pi_{|V|}} \mathrm{MLP}\left(\Big[\mathrm{CNN}(\mathbf{A}_{\pi,\pi}) \bowtie \mathrm{MLP}(\boldsymbol{X}^{(v)}_{\pi})\Big]\right), \tag{3}$$

where CNN denotes a 2D (LeCun et al., 1989; Krizhevsky et al., 2012) if there are no edge features and a 3D CNN (Ji et al., 2013) if there are edge features, $[\cdot \bowtie \cdot]$ is a concatenation of the representations, and MLP is a multilayer perceptron. Multi-resolution 3D convolutions (Qi et al., 2016) can be used to map variable-sized graphs into the same sized representation for downstream layers.

GNNs. The function $\vec{f}$ can also be a graph neural network (GNN), a broad class of models that use the graph G itself to define the computation graph. These are permutation-invariant by design but we will show that their integration into RP can (1) make them more powerful and (2) speed up their computation via theoretically sound approximations. The GNNs we consider follow a message-passing (Gilmer et al., 2017) scheme defined by the recursion

$$\mathbf{h}_u^{(l)} = \phi^{(l)}\left(\mathbf{h}_u^{(l-1)}, \text{JP}\big((\mathbf{h}_v^{(l-1)})_{v\in\mathcal{N}(u)}\big)\right), \qquad (4)$$

where $\phi^{(l)}$ is a learnable function with distinct weights at each layer $1 \le l \le L$ of the computation graph, JP is a general (learnable) permutation-invariant function (Murphy et al., 2019), $\mathcal{N}(u)$ is the set of neighbors of $u \in V$, and $\mathbf{h}_u^{(l)} \in \mathbb{R}^{d_h^{(l)}}$ is a vector describing the embedding of node u at layer l. $\mathbf{h}_u^{(0)}$ is the feature vector of node u, $(\boldsymbol{X}^{(v)})_{u,\cdot}$ or can be assigned a constant c if u has no features. Under this framework, node embeddings can be used directly to predict node-level targets, or all node embeddings can be aggregated (via a learnable function) to form an embedding $\mathbf{h}_G$ used for graph-wide tasks.

There are several variations of Equation 4 in the literature. Duvenaud et al. (2015) proposed using embeddings from all layers $l \in \{1, 2, \ldots, L\}$ for graph classification. Hamilton et al. (2017a) used a similar framework for node classification and link prediction tasks, using the embedding at the last layer, while Xu et al. (2018) extend Hamilton et al. (2017a) to once again use embeddings at all layers for node and link prediction tasks. Other improvements include attention (Velickovic et al., 2018). This approach can be derived from spectral graph convolutions (e.g., (Kipf & Welling, 2017)). More GNNs are discussed in Section 3.

Recently, Xu et al. (2019); Morris et al. (2019) showed that these architectures are at most as powerful as the Weisfeiler-Lehman (WL) algorithm for testing graph isomorphism (Weisfeiler & Lehman, 1968), which itself effectively follows a message-passing scheme. Accordingly, we will broadly refer to models defined by Equation 4 as WL-GNNs. Xu et al. (2019) proposes a WL-GNN called *Graph Isomorphism Network* (GIN) which is as powerful as the WL test in graphs with discrete features.

Can a WL-GNN be more powerful than the WL test? WL-GNNs inherit a shortcoming from the WL test (Cai et al., 1992; Arvind et al., 2017; Fürer, 2017; Morris et al., 2019); node representations $\mathbf{h}_u^{(l)}$ do not encode whether two nodes have the *same neighbor* or *distinct neighbors with the same features*, limiting their ability to learn an expressive representation of the entire graph. Consider a task where graphs represent molecules, where node features indicate atom type and edges denote the presence or absence of bonds. Here, the first WL-GNN layer cannot distinguish that two (say) carbon atoms have a bond with the same carbon atom or a bond to two distinct carbon atoms. Successive layers of the WL-GNN update node representations and the hope is that nodes eventually get unique representations (up to isomorphisms), and thus allow the WL-GNN to detect whether two nodes have the same neighbor based on the representations of their neighbors. However, if there are too few WL-GNN layers or complex cycles in the graph, the graph and its nodes will not be adequately represented.

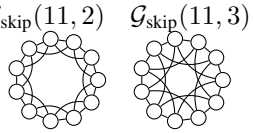


Figure 1: The WL test incorrectly deems these isomorphic.

To better understand this challenge, consider the extreme case illustrated by the two graphs in Figure 1. These are cycle graphs with $M = 11$ nodes where nodes that are $R \in \{2, 3\}$ 'hops' around the circle are connected by an edge. These highly symmetric graphs, which are special cases of circulant graphs (Vilfred, 2004) are formally defined in Definition 2.1 but the key point is that the WL test, and thus WL-GNNs, cannot distinguish these two nonisomorphic graphs.

Definition 2.1: [Circulant Skip Links (CSL) graphs] Let R and M be co-prime natural numbers[1] such that $R < M - 1$. $\mathcal{G}_{\text{skip}}(M, R)$ denotes an undirected 4-regular graph with vertices $\{0, 1, \ldots, M-1\}$ whose edges form a cycle and have skip links. That is, for the cycle, $\{j, j+1\} \in E$ for $j \in \{0, \ldots, M-2\}$ and $\{M-1, 0\} \in E$. For the skip links, recursively define the sequence $s_1 = 0$, $s_{i+1} = (s_i + R) \bmod M$ and let $\{s_i, s_{i+1}\} \in E$ for any $i \in \mathbb{N}$. $\Diamond$

We will use RP to help WL-GNNs overcome this shortcoming. Let $\vec{f}$ be a WL-GNN that we make permutation sensitive by assigning each node an identifier that depends on π. Permutation sensitive IDs prevent the RP sum from collapsing to just one term but more importantly help distinguish neighbors that otherwise appear identical. In particular, given any $\pi \in \Pi_{|V|}$, we append to the rows of $\boldsymbol{X}_\pi^{(v)}$ one-hot encodings of the row number before computing $\vec{f}$. We can represent this by an augmented vertex attribute ma-

[1] Two numbers are co-primes if their only common factor is 1.

trix $\left[\boldsymbol{X}_\pi^{(v)} \bowtie I_{|V|}\right]$ for every $\pi \in \Pi_{|V|}$, where $I_{|V|}$ is a $|V| \times |V|$ identity matrix and $[B \bowtie C]$ concatenates the columns of matrices B and C. RP-GNN is then given by

$$\overline{\vec{f}}(G) = \frac{1}{|V|!}\sum_{\pi\in\Pi_{|V|}} \vec{f}\left(\mathbf{A}_{\pi,\pi}, \left[\boldsymbol{X}_\pi^{(v)} \bowtie I_{|V|}\right]\right) \quad (5)$$
$$= \frac{1}{|V|!}\sum_{\pi\in\Pi_{|V|}} \vec{f}\left(\mathbf{A}, \left[\boldsymbol{X}^{(v)} \bowtie (I_{|V|})_\pi\right]\right),$$

where the second holds since $\vec{f}$ a GNN and thus invariant to permutations of the adjacency matrix. The following theorem shows that $\overline{\vec{f}}(G)$ in Equation 5 is strictly more *expressive* than the original WL-GNN; it can distinguish all nodes and graphs that WL-GNN can in addition to graphs that the original WL-GNN cannot.

Theorem 2.2. *The RP-GNN in Equation 5 is strictly more expressive than the original WL-GNN. Specifically, if $\vec{f}$ is a GIN (Xu et al., 2019) and the graph has discrete attributes, its RP-GNN is more powerful than the WL test.*

Equation 5 is computationally expensive but can be made tractable while retaining expressive power over standard GNNs. While all approximations discussed in Section 2.3 for RP in general are applicable to RP-GNN, a specific strategy is to assign permutation-sensitive node IDs in a clever way. In particular, if vertex features are available, we only need to assign enough IDs to make all vertices unique and thereby reduce the number of permutations we need to evaluate. For example, in the molecule CH_2O_2, if we create node features with one-hot IDs $(C,0,1)$,$(H,0,1)$,$(H,1,0)$,$(O,0,1)$,$(O,1,0)$, then we need only consider $1!\cdot\, 2!\cdot 2! = 4$ permutations. For unattributed graphs, we assign $i \bmod m$ to node i; setting m=1 reduces to a GNN and m=$|V|$ is the most expressive. More examples are in the Supplementary Material.

2.3. RP Tractability

2.3.1. Tractability via canonical orientations

Equation 1 is intractable as written and calls for approximations. The most direct approximation is to compose a permutation-sensitive $\vec{f}$ with a canonical orientation function that re-orders $\mathbf{A}$ such that CANONICAL($\mathbf{A}, \boldsymbol{X}^{(v)}$)= CANONICAL($\mathbf{A}_{\pi,\pi}, \boldsymbol{X}_\pi^{(v)}$), $\forall\pi \in \Pi_{|V|}$. For instance, vertices can be sorted by centrality scores with some tie-breaking scheme (Montavon et al., 2012; Niepert et al., 2016). This causes the sum over all permutations to collapse to just an evaluation of $\vec{f} \circ$ CANONICAL. Essentially, this introduces a fixed component into the permutation-invariant function $\overline{\vec{f}}$ with only the second stage learned from data. This simplifying approximation to the original problem is however only useful if CANONICAL is related to the true function, and can otherwise result in poor representations (Murphy et al., 2019).

A more flexible approach collapses the set of all permutations into a smaller set of *equivalent permutations* which we denote as *poly-canonical orientation*. Depth-First Search (DFS) and Breadth-First Search (BFS) serve as two examples. In a DFS, the nodes of the adjacency matrix/tensor $\mathbf{A}_{\pi,\pi}$ are ordered from 1 to $|V|$ according to the order they are visited by a DFS starting at $\pi(1)$. Thus, if G is a length-three path and we consider permutation functions defined (elementwise) as $\pi(1,2,3) = (1,2,3)$, $\pi'(1,2,3)=(1,3,2)$, DFS or BFS would see respectively ①-②-③ and ①-③-② (where vertices are numbered by permuted indices), start at $\pi(1)$=1 and result in the same 'left-to-right' orientation for both permutations. In disconnected graphs, the search starts at the first node of each connected component. Learning orientations from data is a discrete optimization problem left for future work.

2.3.2. Tractability via π-SGD

A simple approach for making RP tractable is to sample *random* permutations during training. This offers the computational savings of a single canonical ordering but circumvents the need to learn a good canonical ordering for a given task. This approach is only approximately invariant, a tradeoff we make for the increased power of RP.

For simplicity, we analyze a supervised graph classification setting with a single sampled permutation, but this can be easily extended to sampling multiple permutations and unsupervised settings. Further, we focus on joint invariance but the formulation is similar for separate invariance. Consider N training data examples $\mathcal{D} \equiv \{(G(1), \boldsymbol{y}(1)), \ldots, (G(N), \boldsymbol{y}(N))\}$, where $\boldsymbol{y}(i) \in \mathbb{Y}$ is the target output and graph $G(i)$ its corresponding graph input. For a parameterized function $\vec{f}$ with parameters $\boldsymbol{W}$,

$$\overline{\vec{f}}(G(i); \boldsymbol{W}) = \frac{1}{|V(i)|!}\sum_{\pi\in\Pi_{|V(i)|}} \vec{f}(\mathbf{A}_{\pi,\pi}(i), \boldsymbol{X}_\pi^{(v)}(i); \boldsymbol{W}),$$

our (original) goal is to minimize the empirical loss

$$\overline{\overline{L}}(\mathcal{D}; \boldsymbol{W}) = \sum_{i=1}^{N} L\left(\boldsymbol{y}(i)\,,\, \overline{\vec{f}}(G(i); \boldsymbol{W})\right), \quad (6)$$

where L is a convex loss function of $\overline{\vec{f}}(\cdot;\cdot)$ such as cross-entropy or square loss. For each graph $G(i)$, we sample a permutation $\mathbf{s}_i \sim \text{Unif}(\Pi_{|V(i)|})$ and replace the sum in Equation 1 with the estimate

$$\hat{\overline{\vec{f}}}(G(i); \boldsymbol{W}) = \vec{f}(\mathbf{A}_{\mathbf{s}_i,\mathbf{s}_i}(i), \boldsymbol{X}_{\mathbf{s}_i}^{(v)}(i); \boldsymbol{W}). \quad (7)$$

For separate invariance, we would sample a distinct permutation for each set of vertices. The estimator in Equation 7

is unbiased: $E_{\mathbf{s}_i}[\hat{\overline{\overline{f}}}(G_{\mathbf{s}_i,\mathbf{s}_i}(i);\boldsymbol{W})] = \overline{\overline{f}}(G(i);\boldsymbol{W})$, where $G_{\mathbf{s}_i,\mathbf{s}_i}$ is shorthand for a graph that has been permuted by $\mathbf{s}_i$. However, this is no longer true when $\overline{\overline{f}}$ is chained with a nonlinear loss L: $E_{\mathbf{s}_i}[L(\boldsymbol{y}(i), \hat{\overline{\overline{f}}}(G_{\mathbf{s}_i,\mathbf{s}_i}(i);\boldsymbol{W}))] \neq L(\boldsymbol{y}(i), E_{\mathbf{s}_i}[\hat{\overline{\overline{f}}}(G_{\mathbf{s}_i,\mathbf{s}_i}(i);\boldsymbol{W})])$. Nevertheless, as we will soon justify, we follow Murphy et al. (2019) and use this estimate in our optimization.

Definition 2.2: [π-SGD for RP] Let $\mathcal{B}_t = \{(G(1), \boldsymbol{y}(1)), \ldots, (G(B), \boldsymbol{y}(B))\}$ be a mini-batch i.i.d. sampled uniformly from the training data $\mathcal{D}$ at step t. To train RP with π-SGD, we follow the stochastic gradient descent update

$$\boldsymbol{W}_t = \boldsymbol{W}_{t-1} - \eta_t \mathbf{Z}_t, \tag{8}$$

where $\mathbf{Z}_t = \frac{1}{B}\sum_{i=1}^{B} \nabla_{\boldsymbol{W}} L\left(\boldsymbol{y}(i), \hat{\overline{\overline{f}}}(G(i);\boldsymbol{W}_{t-1})\right)$ is the random gradient with the random permutations $\{\mathbf{s}_i\}_{i=1}^{B}$, (sampled independently $\mathbf{s}_i \sim \text{Unif}(\Pi_{|V(i)|})$ for all graphs $G(i)$ in batch $\mathcal{B}_t$), and the learning rate is $\eta_t \in (0,1)$ s.t. $\lim_{t\to\infty}\eta_t = 0$, $\sum_{t=1}^{\infty}\eta_t = \infty$, and $\sum_{t=1}^{\infty}\eta_t^2 < \infty$. $\Diamond$

Effectively, this is a Robbins-Monro stochastic approximation algorithm of gradient descent (Robbins & Monro, 1951; Bottou, 2012) and optimizes the modified objective

$$\begin{aligned}\overline{\overline{J}}(\mathcal{D};\boldsymbol{W}) &= \frac{1}{N}\sum_{i=1}^{N} E_{\mathbf{s}_i}\left[L\left(\boldsymbol{y}(i), \hat{\overline{\overline{f}}}(G_{\mathbf{s}_i,\mathbf{s}_i}(i);\boldsymbol{W})\right)\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|V(i)|!}\sum_{\pi\in\Pi_{|V(i)|}} L\left(\boldsymbol{y}(i), \hat{\overline{\overline{f}}}(G_{\pi,\pi}(i);\boldsymbol{W})\right). \end{aligned} \tag{9}$$

Observe that the expectation over permutations is now outside the loss function (recall $\overline{\overline{f}}(G(i);\boldsymbol{W})$ in in Equation 6 is an expectation). The loss in Equation 9 is also permutation-invariant, but π-SGD yields a result sensitive to the random input permutations presented to the algorithm. Further, unless the function $\vec{f}$ itself is permutation-invariant ($\overline{\overline{f}} = \vec{f}$), the optima of $\overline{\overline{J}}$ are different from those of the original objective function $\overline{\overline{L}}$. Instead, if L is convex in $\overline{\overline{f}}(\cdot;\cdot)$, $\overline{\overline{J}}$ is an upper bound to $\overline{\overline{L}}$ via Jensen's inequality, and minimizing this bound forms a tractable surrogate to the original objective in Equation 6.

The following convergence result follows from the π-SGD formulation of Murphy et al. (2019).

Proposition 2.1. *π-SGD stochastic optimization enjoys properties of almost sure convergence to optimal $\boldsymbol{W}$ under conditions similar to SGD (listed in Supplementary).*

Remark 2.1. *Given fixed point $\boldsymbol{W}^\star$ of the π-SGD optimization and a new graph G at test time, we may exactly compute $E_{\mathbf{s}}[\hat{\overline{\overline{f}}}(G_{\mathbf{s},\mathbf{s}};\boldsymbol{W}^\star)] = \overline{\overline{f}}(G;\boldsymbol{W}^\star)$ or estimate it with $\frac{1}{m}\sum_{j=1}^{m}\vec{f}(G_{\mathbf{s}_j,\mathbf{s}_j};\boldsymbol{W}^\star)$, where $\mathbf{s}_1\ldots,\mathbf{s}_m \overset{\text{i.i.d.}}{\sim} \text{Unif}\big(\Pi_{|V|}\big)$.*

2.3.3. Tractability via k-ary dependencies

Murphy et al. (2019) propose k-ary pooling whereby the computational complexity of summing over all permutations of an input sequence is reduced by considering only permutations of subsequences of size k. Inspired by this, we propose k-ary Relational Pooling which operates on k-node induced subgraphs of G, which corresponds to patches of size $k \times k \times (d_e + 1)$ of $\mathbf{A}$ and k rows of $\boldsymbol{X}^{(v)}$. Formally, we define k-ary RP in joint RP by

$$\overline{\overline{f}}^{(k)}(G;\boldsymbol{W}) = \frac{1}{|V|!}\sum_{\pi\in\Pi_{|V|}} \vec{f}\Big(\mathbf{A}_{\pi,\pi}[1{:}k, 1{:}k, :], \boldsymbol{X}_{\pi}^{(v)}[1{:}k, :]; \boldsymbol{W}\Big), \tag{10}$$

where $\mathbf{A}[\cdot,\cdot,\cdot]$ denotes access to elements in the first, second, and third modes of $\mathbf{A}$; $a:b$ denotes selecting elements corresponding to indices from a to b inclusive; and ":" by itself denotes all elements along a mode. Thus, we permute the adjacency tensor and select fibers along the third mode from the upper left $k \times k \times (d_e+1)$ subtensor of $\mathbf{A}$ as well as the vertex attributes from the first k rows of $\boldsymbol{X}_{\pi}^{(v)}$. An illustration is shown in Figure 2. The graph on the right is numbered by its 'original' node indices and we assume that it has no vertex features and one-dimensional edge features. This 'original' graph would be represented by a $5 \times 5 \times 2$ tensor $\mathbf{A}$ where, for all pairs of vertices, the front slice holds adjacency matrix information and the back slice holds edge feature information (not shown). Given the permutation function $\pi^\dagger \in \Pi_{|V|}$ defined as $\pi^\dagger(1,2,3,4,5) = (3,4,1,2,5)$, the permuted $\mathbf{A}_{\pi^\dagger,\pi^\dagger}$ is shown on the left. Its entries show elements from $\mathbf{A}$ shuffled appropriately by $\pi^\dagger$. For $k=3$ RP, we select the upper-left 3×3 region from $\mathbf{A}_{\pi^\dagger,\pi^\dagger}$, shaded in red, and pass this to $\vec{f}$. This is repeated for all permutations of the vertices. For separate RP, the formulation is similar but we can select k_1 and k_2 nodes from $V^{(\text{r})}$ and $V^{(\text{c})}$, respectively.

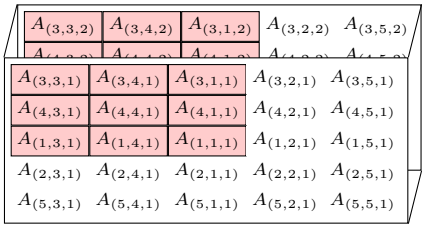
In practice, the relevant k-node induced subgraphs can be selected without first permuting the entire tensor $\mathbf{A}$ and matrix $\boldsymbol{X}^{(v)}$. Instead, we enumerate all subsets of size k from index set V and use those to index $\mathbf{A}$ and $\boldsymbol{X}^{(v)}$.

More generally, we have the following conclusion:

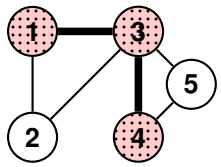
Proposition 2.2. *The RP in Equation 10 requires summing over all k-node induced subgraphs of G, thus saving computation when $k < |V|$, reducing the number of terms in the sum from $|V|!$ to $\frac{|V|!}{(|V|-k)!}$.*

Fewer computations are needed if $\vec{f}$ is made permutation-invariant over its input k-node induced subgraph. We now show that the expressiveness of k-ary RP increases with k.

Proposition 2.3. *$\overline{\overline{f}}^{(k)}$ becomes strictly more expressive as k increases. That is, for any $k\in\mathbb{N}$, define $\mathcal{F}_k$ as the set of all permutation-invariant graph functions that can be represented by RP with k-ary dependencies. Then, $\mathcal{F}_{k-1} \subset \mathcal{F}_k$.*



Adjacency tensor $\mathbf{A}_{\pi^\dagger,\pi^\dagger}$ where $\pi^\dagger(1,2,3,4,5) = (3,4,1,2,5)$ (elementwise): the top-left $3 \times 3 \times 2$ subtensor is passed to $\vec{f}^{(3)}$.



An example five-node graph encoded by $\mathbf{A}$. We select a 3-node induced subgraph, corresponding to the top-left of $\mathbf{A}_{\pi^\dagger,\pi^\dagger}$ indicated by shaded nodes and thickened edges.

Figure 2: Illustration of a k-ary ($k = 3$) RP on a 5-node graph with one-dimensional edge attributes ($d_e = 1$) and no vertex attributes. The graph is encoded as a $5 \times 5 \times 2$ tensor $\mathbf{A}$. k-ary RP selects the top-left $k \times k$ corner of a permuted tensor $\mathbf{A}_{\pi,\pi}$.

Further computational savings. The number of k-node induced subgraphs can be very large for even moderate-sized graphs. The following yield additional savings.

Ignoring some subgraphs: We can encode task- and model-specific knowledge by ignoring certain k-sized induced subgraphs, which amounts to fixing $\vec{f}$ to 0 for these graphs. For example, in most applications the graph structure – and not the node features alone – is important so we may ignore subgraphs of k isolated vertices. Such decisions can yield substantial computational savings in sparse graphs.

Use of π-SGD: We can combine the k-ary approximation with other strategies like π-SGD and poly-canonical orientations. For instance, a forward pass can consist of sampling a random starting vertex and running a BFS until a k-node induced subgraph is selected. Combining π-SGD and k-ary RP can speed up GNNs but will not provide unbiased estimates of the loss calculated with the entire graph. Future work could explore using the MCMC finite-sample unbiased estimator of Teixeira et al. (2018) with RP.

3. Related Work

Our Relational Pooling framework leverages insights from Janossy Pooling (Murphy et al., 2019), which learns expressive permutation-invariant functions over *sequences* by approximating an average over permutation-sensitive functions with tractability strategies. The present work raises novel applications – like RP-GNN – that arise when pooling over permutation-sensitive functions of *graphs*.

Graph Neural Networks (GNNs) and Graph Convolutional Networks (GCNs) form an increasingly popular class of methods (Scarselli et al., 2009; Bruna et al., 2014; Duvenaud et al., 2015; Niepert et al., 2016; Atwood & Towsley, 2016; Kipf & Welling, 2017; Gilmer et al., 2017; Monti et al., 2017; Defferrard et al., 2016; Hamilton et al., 2017a; Velickovic et al., 2018; Lee et al., 2018; Xu et al., 2019). Applications include chemistry, where molecules are represented as graphs and we seek to predict chemical properties like toxicity (Duvenaud et al., 2015; Gilmer et al., 2017; Lee et al., 2018; Wu et al., 2018; Sanchez-Lengeling & Aspuru-Guzik, 2018) and document classification on a citations network (Hamilton et al., 2017b); and many others (cf. Battaglia et al. (2018)).

Recently, Xu et al. (2019) and Morris et al. (2019) show that such GNNs are at most as powerful as the standard Weisfeiler-Lehman algorithm (also known as *color refinement* or *naive vertex classification* (Weisfeiler & Lehman, 1968; Arvind et al., 2017; Fürer, 2017)) for graph isomorphism testing, and can fail to distinguish between certain classes of graphs (Cai et al., 1992; Arvind et al., 2017; Fürer, 2017). In Section 4, we demonstrate this phenomenon and provide empirical evidence that RP can correct some of these shortcomings. Higher-order (k-th order) versions of the WL test (WL[k]) exist and operate on tuples of size k from V rather than on one vertex at a time (Fürer, 2017). Increasing k increases the capacity of WL[k] to distinguish nonisomorphic graphs, which can be exploited to build more powerful GNNs (Morris et al., 2019). Meng et al. (2018), introduce a WL[k]-type representation to predict high-order dynamics in temporal graphs. Using GNNs based on WL[k] may be able to give better $\vec{f}$ functions for RP but we focused on providing a representation for more expressive than WL[1] procedures.

In another direction, WL is used to construct graph kernels (Shervashidze et al., 2009; 2011). CNNs have also been used with graph kernels (Nikolentzos et al., 2018) and some GCNs can be seen as CNNs applied to single canonical orderings (Niepert et al., 2016; Defferrard et al., 2016). RP provides a framework for stochastic optimization over all or poly-canonical orderings. Another line of work derives bases for permutation-invariant functions of graphs and propose learning the coefficients of basis elements from data (Maron et al., 2018; Hartford et al., 2018).

In parallel, Bloem-Reddy & Teh (2019) generalized permutation-invariant functions to group-action invariant functions and discuss connections to exchangeable prob-

ability distributions (De Finetti, 1937; Diaconis & Janson, 2008; Aldous, 1981).Their theory uses a checkerboard function (Orbanz & Roy, 2015) and the left-order canonical orientation of Ghahramani & Griffiths (2006) to orient graphs but it will fail in some cases unless graph isomorphism can be solved in polynomial time. Also, as discussed, there is no guarantee that a hand-picked canonical orientation will perform well on all tasks. On the tractability side, Niepert & Van den Broeck (2014) shows that exchangeabilty assumptions in probabilistic graphical models provide a form of k-ary tractability and Cohen & Welling (2016); Ravanbakhsh et al. (2017) use symmetries to reduce sample complexity and save on computation. Another development explores the universality properties of invariance-preserving neural networks and concludes some architectures are computationally intractable (Maron et al., 2019). Closer to RP, Montavon et al. (2012) discusses random permutations but RP provides a more comprehensive framework with theoretical analysis.

4. Experiments

Our first experiment shows that RP-GNN is more expressive than WL-GNN. The second evaluates RP and its approximations on molecular data. Our code is on GitHub[2].

4.1. Testing RP-GNN vs WL-GNN

Here we perform experiments over the CSL graphs from Figure 1. We demonstrate empirically that WL-GNNs are limited in their power to represent them and that RP can be used to overcome this limitation. Our experiments compare the RP-GNN of Equation 5 using the Graph Isomorphism Network (GIN) architecture (Xu et al., 2019) as $\vec{f}$ against the original GIN architecture. We choose GIN as it is arguably the most powerful WL-GNN architecture.

For the CSL graphs, the "skip length" R effectively defines an isomorphism class in the sense that predicting R is tantamount to classifying a graph into its isomorphism class for a fixed number of vertices M. We are interested in predicting R as an assessment of RP's ability to exploit graph structure. We do not claim to tackle the graph isomorphism problem as we use approximate learning (π-SGD for RP).

RP-GIN. GIN follows the recursion of Equation 4, replacing JP with summation and defining $\phi^{(l)}$ as a function that sums its arguments and feeds them through an MLP:

$$\mathbf{h}_u^{(l)} = \mathrm{MLP}^{(l)}\Big((1+\epsilon^{(l)})\mathbf{h}_u^{(l-1)} + \sum_{v\in\mathcal{N}(u)} \mathbf{h}_v^{(l-1)}\Big),$$

for $l = 1, \ldots, L$, where $\{\epsilon^{(l)}\}_{l=1}^{L}$ can be treated as hyperparameters or learned parameters (we train ϵ). This recursion yields vertex-level representations that can be mapped to a graph-level representation by summing across $\mathbf{h}_u^{(l)}$ at each given l, then concatenating the results, as proposed by Xu et al. (2019). When applying GIN directly on our CSL graphs, we assign a constant vertex attribute to all vertices in keeping with the traditional WL algorithm, as the graph is unattributed. Recall that RP-GIN assigns one-hot node IDs and passes the augmented graph to GIN ($\vec{f}$) (Equation 5). We cannot assign IDs with standard GIN as doing so renders it permutation-sensitive. Further implementation and training details are in the Supplementary Material.

[2] https://github.com/PurdueMINDS/RelationalPooling

Table 1: RP-GNN outperforms WL-GNN in 10-class classification task. Summary of validation-set accuracy (%).

model	mean	median	max	min	sd
RP-GIN	37.6	43.3	53.3	10.0	12.9
GIN	10.0	10.0	10.0	10.0	0.0

Classifying skip lengths. We create a dataset of graphs from $\{\mathcal{G}_{\text{skip}}(41, R)\}_R$ where $R \in \{2, 3, 4, 5, 6, 9, 11, 12, 13, 16\}$ and predict R as a discrete response. Note $M = 41$ is the smallest such that 10 nonisomorphic $\mathcal{G}_{\text{skip}}(M, R)$ can be formed; $\exists R_1 \neq R_2$ such that $\mathcal{G}_{\text{skip}}(M, R_1)$ and $\mathcal{G}_{\text{skip}}(M, R_2)$ are isomorphic. For all 10 classes, we form 15 adjacency matrices by first constructing $A^{(R)}$ according to Definition 2.1 and then 14 more as $A^{(R)}_{\pi,\pi}$ for 14 distinct permutations π. This gives a dataset of 150 graphs. We evaluate GIN and RP-GIN with five-fold cross validation – with balanced classes on both training and validation – on this task.

The validation-set accuracies for both models are shown in Table 1 and Figure 3 in the Supplementary Material. Since GIN learns the same representation for all graphs, it predicts the same class for all graphs in the validation fold, and therefore achieves random-guessing performance of 10% accuracy. In comparison, RP-GIN yields substantially stronger performance on all folds, demonstrating that RP-GNNs are more powerful than their WL-GNN and serving as empirical validation of Theorem 2.2.

4.2. Predicting Molecular Properties

Deep learning for chemical applications learns functions on graph representations of molecules and has a rich literature (Duvenaud et al., 2015; Kearnes et al., 2016; Gilmer et al., 2017). This domain provides challenging tasks on which to evaluate RP, while in other applications, different GNN models of varying sophistication often achieve similar performance (Shchur et al., 2018; Murphy et al., 2019; Xu et al., 2019). We chose datasets from the MoleculeNet project (Wu et al., 2018) – which collects chemical datasets and collates the performance of various models – that yield classification tasks and on which graph-

based methods achieved superior performance[3]. In particular, we chose MUV (Rohrer & Baumann, 2009), HIV, and Tox21 (Mayr et al., 2016; Huang et al., 2016), which contain measurements on a molecule's biological activity, ability to inhibit HIV, and qualitative toxicity, respectively.

We processed datasets with DeepChem (Ramsundar et al., 2019) and evaluated models with ROC-AUC per the MoleculeNet project. Molecules are encoded as graphs with 75- and 14-dimensional node and edge features. Table 3 (in Supplementary) provides more detail.

We use the best-performing graph model reported by MoleculeNet as $\vec{f}$ to evaluate k-ary RP and to explore whether RP-GNN can make it more powerful. This is a model inspired by the GNN in Duvenaud et al. (2015), implemented in DeepChem by Altae-Tran et al. (2017), which we refer to as the 'Duvenaud et al.' model. This model is specialized for molecules; it trains a distinct weight matrix for each possible vertex degree at each layer, which would be infeasible in other domains. One might ask whether RP-GNN can add any power to this state-of-the-art model, which we will explore here. We evaluated GIN (Xu et al., 2019) but it was unable to outperform 'Duvenaud et al'. Model architectures, hyperparameters, and training procedures are detailed in the Supplementary Material.

RP-GNN We compare the performance of the 'Duvenaud et al.' baseline to RP-Duvenaud, wherein the 'Duvenaud et al.' GNN is used as $\vec{f}$ in Equation 5. We evaluate $\vec{f}$ on the entire graph but make RP-Duvenaud tractable by training with π-SGD. At inference time, we sample 20 permutations (see Remark 2.1). Additionally, we assign just enough one-hot IDs to make atoms of the same type have unique IDs (as discussed in Section 2.2). To quantify variability, we train over 20 random data splits.

The results shown in Table 2 suggest that RP-Duvenaud is more powerful than the baseline on the HIV task and similar in performance on the others. While we bear in mind the over-confidence in the variability estimates (Bengio & Grandvalet, 2004), this provides support of our theory.

k-ary RP experiments Next we empirically assess the tradeoffs involved in the k-ary dependency models – evaluating $\vec{f}$ on k-node induced subgraphs – discussed in Section 2.3.3. Propositions 2.3 and 2.2 show that expressive power and computation decrease with k. Here, $\vec{f}$ is a 'Duvenaud et al. model' that operates on induced subgraphs of size $k = 10, 20, 30, 40, 50$ (the percentages of molecules with more than k atoms in each dataset are shown in the Supplementary Material). We train using π-SGD (20 inference-time samples) and evaluate using five random train/val/test splits.

[3] `moleculenet.ai/latest-results`, (Dec. 2018)

Table 2: Evaluation of RP-GNN and k-ary RP where $\vec{f}$ is the 'Duvenaud et al.' GNN or a neural-network. We show mean (standard deviation) ROC-AUC across multiple random train/val/test splits. DFS indicates Depth-First Search poly-canonical orientation.

model	**HIV**	**MUV**	**Tox21**
RP-Duvenaud et al.	0.832 (0.013)	0.794 (0.025)	0.799 (0.006)
Duvenaud et al.	0.812 (0.014)	0.798 (0.025)	0.794 (0.010)
$k=50$ Duvenaud et al.	0.818 (0.022)	0.768 (0.014)	0.778 (0.007)
$k=40$ Duvenaud et al.	0.807 (0.025)	0.776 (0.032)	0.783 (0.007)
$k=30$ Duvenaud et al.	0.829 (0.024)	0.776 (0.030)	0.775 (0.011)
$k=20$ Duvenaud et al.	0.813 (0.017)	0.777 (0.041)	0.755 (0.003)
$k=10$ Duvenaud et al.	0.812 (0.035)	0.773 (0.045)	0.687 (0.005)
CNN-DFS	0.542 (0.004)	0.601 (0.042)	0.597 (0.006)
RNN-DFS	0.627 (0.007)	0.648 (0.014)	0.748 (0.055)

Results are shown in Table 2 and Figures 4, 5, and 6 in the Supplementary Material. With the Tox21 dataset, we see a steady increase in predictive performance and computation as k increases. For instance, k-ary with $k = 10$ is 25% faster than the baseline with mean AUC 0.687 (0.005 sd) and with $k = 20$ being 10% faster with AUC 0.755 (0.003 sd), where (sd) indicates the standard deviation over 5 bootstrapped runs. Results level off around $k = 30$. For the other datasets, neither predictive performance nor computation vary significantly with k. Overall, the molecules are quite small and we do not expect dramatic speed-ups with smaller k, but this enables comparing between using the entire graph and its k-sized induced subgraphs.

RP with CNNs and RNNs. RP permits using neural networks for $\vec{f}$. We explored RNNs and CNNs and report the results in Table 2. Specific details are discussed in the Supplementary Material. The RNN achieves reasonable performance on Tox21 and underperforms on the other tasks. The CNN underperforms on all tasks. Future work is needed to determine tasks where these approaches are better suited.

5. Conclusions

In this work, we proposed the Relational Pooling (RP) framework for graph classification and regression. RP gives ideal most-powerful, though intractable, graph representations. We proposed several approaches to tractably approximate this ideal and showed theoretically and empirically that RP can make WL-GNNs more expressive than the WL test. RP permits neural networks like RNNs and CNNs to be brought to such problems. Our experiments evaluate RP on a number of datasets and show how our framework can be used to improve properties of state-of-the-art methods. Future directions for theoretical study include improving our understanding of the tradeoff between representation power and computational cost of our tractability strategies.

Acknowledgments

This work was sponsored in part by the ARO, under the U.S. Army Research Laboratory contract number W911NF-09-2-0053, the Purdue Integrative Data Science Initiative and the Purdue Research foundation, the DOD through SERC under contract number HQ0034-13-D-0004 RT #206, and the National Science Foundation under contract numbers IIS-1816499 and DMS-1812197.

References

Aldous, D. J. Representations for partially exchangeable arrays of random variables. *J. Multivar. Anal.*, 11 (4):581–598, 1981. ISSN 0047259X. doi: 10.1016/0047-259X(81)90099-3.

Altae-Tran, H., Ramsundar, B., Pappu, A. S., and Pande, V. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

Arvind, V., Köbler, J., Rattan, G., and Verbitsky, O. Graph isomorphism, color refinement, and compactness. *computational complexity*, 26(3):627–685, 2017.

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

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bengio, Y. and Grandvalet, Y. No unbiased estimator of the variance of k-fold cross-validation. *Journal of machine learning research*, 5(Sep):1089–1105, 2004.

Bloem-Reddy, B. and Teh, Y. W. Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*, 2019.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, jul 2017. ISSN 1053-5888. doi: 10.1109/MSP.2017.2693418. URL http://ieeexplore.ieee.org/document/7974879/.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

Cai, J.-Y., Fürer, M., and Immerman, N. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016.

De Finetti, B. La prévision: ses lois logiques, ses sources subjectives. In *Annales de l'institut Henri Poincaré*, volume 7, pp. 1–68, 1937. [Translated into Enlish: H. E. Kyburg and H.E. Smokler, eds. Studies in Subjective Probability. *Krieger* 53-118, 1980].

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

Diaconis, P. and Janson, S. Graph limits and exchangeable random graphs. *Rend. di Mat. e delle sue Appl. Ser. VII*, 28:33–61, 2008. ISSN 1542-7951. doi: 10.1080/15427951.2008.10129166. URL http://arxiv.org/abs/0712.2749.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Fürer, M. On the combinatorial power of the Weisfeiler-Lehman algorithm. In *International Conference on Algorithms and Complexity*, pp. 260–271. Springer, 2017.

Ghahramani, Z. and Griffiths, T. L. Infinite latent feature models and the Indian buffet process. In *Advances in neural information processing systems*, pp. 475–482, 2006.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In Precup, D. and Teh, Y. W. (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. URL http://proceedings.mlr.press/v70/gilmer17a.html.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017a.

Hamilton, W. L., Ying, R., and Leskovec, J. Representation learning on graphs: Methods and applications. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 40(3):52–74, 2017b.

Hartford, J., Graham, D. R., Leyton-Brown, K., and Ravanbakhsh, S. Deep models of interactions across sets. *arXiv preprint arXiv:1803.02879*, 2018.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Huang, R., Xia, M., Nguyen, D.-T., Zhao, T., Sakamuru, S., Zhao, J., Shahane, S. A., Rossoshek, A., and Simeonov, A. Tox21challenge to build predictive models of nuclear receptor and stress response pathways as mediated by exposure to environmental chemicals and drugs. *Frontiers in Environmental Science*, 3:85, 2016.

Ji, S., Xu, W., Yang, M., and Yu, K. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35 (1):221–231, 2013.

Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

Kingma, D. P. and Ba, J. L. ADAM: A Method for Stochastic Optimization. *International Conference on Learning Representations, ICLR*, 2015.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Lee, J. B., Rossi, R., and Kong, X. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1666–1674. ACM, 2018.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. On the universality of invariant networks. *arXiv preprint arXiv:1901.09342*, 2019.

Mayr, A., Klambauer, G., Unterthiner, T., and Hochreiter, S. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

Meng, C., Mouli, S. C., Ribeiro, B., and Neville, J. Subgraph pattern neural networks for high-order graph evolution prediction. In *AAAI*, 2018.

Montavon, G., Hansen, K., Fazli, S., Rupp, M., Biegler, F., Ziehe, A., Tkatchenko, A., Lilienfeld, A. V., and Müller, K.-R. Learning invariant representations of molecules for atomization energy prediction. In *Advances in Neural Information Processing Systems*, pp. 440–448, 2012.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., and Bronstein, M. M. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJluy2RcFm.

Niepert, M. and Van den Broeck, G. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In *AAAI*, pp. 2467–2475, 2014.

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

Nikolentzos, G., Meladianos, P., Tixier, A. J.-P., Skianis, K., and Vazirgiannis, M. Kernel graph convolutional neural networks. In *International Conference on Artificial Neural Networks*, pp. 22–32. Springer, 2018.

Orbanz, P. and Roy, D. M. Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461, 2015.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

Ramsundar, B., Eastman, P., Leswing, K., Walters, P., and Pande, V. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019. https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2892–2901. JMLR. org, 2017.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Rohrer, S. G. and Baumann, K. Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. *Journal of chemical information and modeling*, 49(2):169–184, 2009.

Sanchez-Lengeling, B. and Aspuru-Guzik, A. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *Relational Representation Learning Workshop (R2L 2018), NeurIPS*, 2018.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Shervashidze, N., Schweitzer, P., Leeuwen, E. J. v., Mehlhorn, K., and Borgwardt, K. M. Wisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

Teixeira, C. H., Cotta, L., Ribeiro, B., and Meira, W. Graph pattern mining and learning through user-defined relations. In *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 1266–1271. IEEE, 2018.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *ICLR*, 2018.

Vilfred, V. On circulant graphs. In Balakrishnan, R., Sethuraman, G., and Wilson, R. J. (eds.), *Graph Theory and its Applications*, pp. 34–36. Narosa Publishing House, 2004.

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

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*, 2018. URL http://arxiv.org/abs/1806.03536.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

Younes, L. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4):177–228, 1999.

Yuille, A. L. The convergence of contrastive divergences. In *Advances in neural information processing systems*, pp. 1593–1600, 2005.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.