Subgraph Permutation Equivariant Networks

Anonymous authors
Paper under double-blind review

Abstract

In this work we develop a new method, named Sub-graph Permutation Equivariant Networks (SPEN), which provides a framework for building graph neural networks that operate on sub-graphs, while using a base update function that is permutation equivariant, that are equivariant to a novel choice of automorphism group. Message passing neural networks have been shown to be limited in their expressive power and recent approaches to over come this either lack scalability or require structural information to be encoded into the feature space. The general framework presented here overcomes the scalability issues associated with global permutation equivariance by operating more locally on sub-graphs. In addition, through operating on sub-graphs the expressive power of higher-dimensional global permutation equivariant networks is improved; this is due to fact that two non-distinguishable graphs often contain distinguishable sub-graphs. Furthermore, the proposed framework only requires a choice of k-hops for creating ego-network sub-graphs and a choice of representation space to be used for each layer, which makes the method easily applicable across a range of graph based domains. We experimentally validate the method on a range of graph benchmark classification tasks, demonstrating statistically indistinguishable results from the state-of-the-art on six out of seven benchmarks. Further, we demonstrate that the use of local update functions offers a significant improvement in GPU memory over global methods.

1 Introduction

Machine learning on graphs has received much interest in recent years with many graph neural network (GNN) architectures being proposed. One such method, which is widely used, is the general framework of message passing neural networks (MPNN). These provide both a useful inductive bias and scalability across a range of domains (Gilmer et al., 2017).

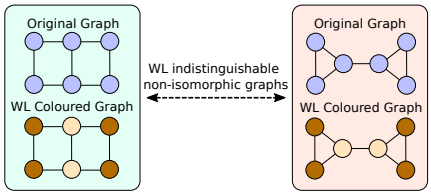

Figure 1: The initial graph on the left is non-isomorphic to the graph on the right. Despite this the WL graph isomorphism test cannot distinguish between the two graphs. Adapted from Bevilacqua et al. (2022).

However, Xu et al. (2019); Morris et al. (2019b) showed that models based on a message passing framework with permutation invariant aggregation functions have expressive power at most that of the Weisfeiler-Lehman (WL) graph isomorphism test (Weisfeiler & Leman, 1968). Therefore, there exist many non-isomorphic graphs that a model of this form cannot distinguish between. Figure 1 provides an example of two non-isomorphic graphs which to a message passing update function are indistinguishable.

Many methods have been proposed to design a GNN that improves the expressive power of MPNNs. Although—most often—an increase in expressivity must be traded off against scalability. We present the background into existing methods which attempt to tackle this question in Section 2.

Our approach. We design a framework to create provably more expressive and scalable graph networks. We achieve this through incorporating symmetry structures in graphs, by considering a graph equivariant update function which operates over sub-graphs. Our framework, *Subgraph Permutation Equivariant Networks (SPEN)*, is developed from the idea that operating on subgraphs could both improve the scalability and

expressive power of higher-dimensional GNNs, whilst unlocking a natural choice of automorphism groups which further increases the expressive power of the network. Our framework consists of:

1. encoding the graph as a bag of bags of sub-graphs,
2. utilising a k-order permutation equivariant base encoder, and
3. constraining the linear map to be equivariant to the automorphism groups of the bags of sub-graphs.

Sub-graphs each have a symmetry group and our framework captures this in two ways. Each sub-graph has a permutation symmetry, which is induced by a permutation of the nodes in the graph. In addition, there is a symmetry across sub-graphs whereby sub-graphs are associated to an automorphism group. We therefore construct a neural network comprising of layers that are equivariant to both permutations of nodes and the automorphism groups of sub-graphs. We achieve this by utilising a permutation equivariant base encoder with feature space constrained by the direct sum of different order permutation representations. Further, we constrain the linear map comprising each layer to be equivariant to the automorphism groups of the bags of sub-graphs. This necessitates that sub-graphs belonging to different automorphism groups are processed by a kernel with different weights, while for sub-graphs belonging to the same automorphism group the kernel shares weights. This leads to us creating a sub-graph extraction policy which generates a bag of bags of sub-graphs, where each bag of sub-graphs corresponds to a different sub-graph automorphism group.

Our contributions

1. An automorphism equivariant compatible subgraph extraction method.
2. A novel choice of automorphism groups with which to constrain the linear map to be equivariant to.
3. A more scalable framework for utilising higher-dimensional permutation equivariant GNNs.
4. A more expressive model than higher-dimensional permutation equivariant GNNs and sub-graph MPNNs.
5. A theoretical analysis of the proposed model in terms of scalability and an application of the theoretical analysis from Bevilacqua et al. (2022) and de Haan et al. (2020) to demonstrate the expressivity of our model.
6. A demonstration that our method is statistically indistinguishable from state-of-the-art methods on benchmark graph classification tasks.

Throughout this paper we are following the notation presented by (Bevilacqua et al., 2022) for how we present what a subgraph is and for the expressivity analysis.[1]

2 Background

More expressive graph neural networks (GNNs) exist which can be grouped into three different groups: (1) those which design higher-dimensional GNNs, (2) those which use positional encodings through pre-coloring nodes, and (3) those which use sub-graphs/local equivariance. Several architectures have been proposed of the type (1) which design a high-order GNN equivalent to the hierarchy of k-WL tests (Maron et al., 2018; 2019; Morris et al., 2019b;a; Keriven & Peyré, 2019; Azizian & Lelarge, 2021; Zhang & Li, 2021). Despite being equivalent to the k-WL test, and hence having provably strong expressivity; these models lose the advantage of locality and linear complexity. As such, the scalability of such models poses an issue for their practical use, with Maron et al. (2018) showing that the basis space for permutation equivariant models of order k is equal to the $2k^{th}$ Bell number, which results in a basis space of size 2 for order-1 tensors, 15 for order-2 tensors, 203 for order-3 tensors, and 4140 for order-4 tensors, demonstrating the practical challenge of using higher-dimensional GNNs. Several architectures have also been proposed of type (2) where authors seek to

[1]One of the contributions of our paper, namely operating on subgraphs which inherit ids from the original graph, was developed concurrently to (Bevilacqua et al., 2022), see `https://openreview.net/forum?id=7oyVOECcrt` for the initial version of our work in September 2021.

introduce a pre-coloring or positional encoding that is permutation invariant. These comprise of pre-coloring nodes based on pre-defined substructures (Bouritsas et al., 2020) or lifting graphs into simplicial- (Bodnar et al., 2021b) or cell complexes (Bodnar et al., 2021a). These methods require a pre-computation stage, which in the worst-case finding substructures of size k in a graph of n nodes is $\mathcal{O}(n^k)$. Finally sub-graphs/local equivariance of type (3) have been considered to find more expressive GNNs. Local graph equivariance requires a (linear) map that satisfies an automorphism equivariance constraint. This is due to the nature of graphs having different local symmetries on different nodes/edges. This has been considered by de Haan et al. (2020) though imposing an isomorphism/automorphism constraint on edge neighbourhoods and by Thiede et al. (2021) by selecting specific automorphism groups and lifting the graph to these. Although the choice of automorphism group chosen by de Haan et al. (2020) leads to little weight sharing and requires the automorphism constraint to be parameterized, while those proposed by Thiede et al. (2021) and Xu et al. (2021) do not guarantee to capture the entire graph and requires a hard-coded choice of automorphism group. Xu et al. (2021) also propose a method for searching across different sub-graph templates, although this still requires some hard-coding. Operating on sub-graphs has been considered as a means to improve GNNs by dropping nodes (Papp et al., 2021; Cotta et al., 2021), dropping edges (Rong et al., 2019), utilising ego-network graphs (Zhao et al., 2021), and considering the symmetry of a bag of sub-graphs (Bevilacqua et al., 2022).

3 Subgraph Permutation Equivariant Networks (SPEN)

We first outline necessary definitions that our method builds upon. Following this we present the two main concepts of our SPEN model. Firstly, SPEN has a k-ego net subgraph extraction method, which are collected into bags determined by their automorphism group. Secondly, SPEN has an automorphism equivariant graph neural network. This section presents the core concepts of the model which contribute to the improved expressivity and classification performance. The overall architecture of the SPEN model is presented in Figure 2. Further, the breakdown of an automorphism equivariant layer within our framework is presented in Figure 3. In addition, in Figure 4 we detail an example breakdown of one of the mapping functions used within our model. Finally, Figure 5 shows a visualisation of some of the bases used within the mapping functions in the automorphism equivariant functions. In addition, we present a more general overview of the architectural details of the SPEN framework in Appendix A.5. The key definitions required to understand our framework are provided below, with further useful definitions provided in Appendix A.1.

3.1 Definitions

In this work we consider graphs as concrete graphs and utilise sub-concrete graphs in our framework. The sub-graphs are extracted as k-ego network sub-graphs. This leads us to define the graphs and sub-graphs.

Definition 3.1. A *Concrete Graph* (de Haan et al., 2020) G is a finite set of nodes $\mathcal{V}(G) \subset \mathbb{N}$ and a set of edges $\mathcal{E}(G) \subset \mathcal{V}(G) \times \mathcal{V}(G)$.

The set of node ids may be non-contiguous and we make use of this here as we extract overlapping sub-graphs and perform the graph update function on this bag of sub-graphs. "The natural numbers of the nodes are essential for representing the graphs in a computer, but hold no actual information about the underlying graph" (de Haan et al., 2020). Therefore, the same underlying graph can be given in may forms by a permutation of the ordering of the natural numbers of the nodes. Throughout the paper we refer to concrete graphs as graphs to minimise notation.

Definition 3.2. In tensor format the values of G are encoded in a tensor $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}(G)| \times |\mathcal{V}(G)| \times d}$.

The node features are encoded along the diagonal and edge features encoded in off-diagonal positions.

Definition 3.3. A *sub-Concrete Graph* H is created by taking a node $i \in \mathcal{V}(G)$, and extracting the nodes $j \in \mathcal{V}(G)$ and edges $(i, j) \subset \mathcal{V}(G) \times \mathcal{V}(G)$, according to some sub-graph selection policy.

In this work we consider the sub-graph selection policy as a k-ego-network policy. For brevity we refer to sub-concrete graphs as sub-graphs throughout the paper.

Definition 3.4. A *k-Ego Network* of a node is its k-hop neighbourhood with induced connectivity.

In this work we are interested in the symmetries between graphs. For this we considered the automorphism symmetries of graphs through consideration of the automorphism group. Here we consider the automorphism group of the permutation group and therefore make some necessary definitions for the consideration of groups and their representations.

Definition 3.5. A *group representation* ρ of the group G is a homomorphism $\rho : G \to \mathrm{GL}(V)$ of G to the group of automorphisms of V (Fulton & Harris, 2013). A group representation associates to each $g \in G$ an invertible matrix $\rho(g) \in \mathbb{R}^{n \times n}$. This can be understood as specifying how the group acts as a transformation on the input.

Definition 3.6. A *feature space* is a vector space V with a group representation ρ acting on it. The choice of group representations on the input and output vector spaces of a linear map constrains the possible forms the linear map can take.

Definition 3.7. A *tensor representation* can be built up from some base group representations $\rho(g)$ through the tensor operations dual ($*$), direct sum ($\oplus$), and tensor product ($\otimes$). This allows for tensor representations to be constructed that are of increasing size and complexity.

Definition 3.8. A *kernel constraint* is taken to mean a restriction of the space a kernel or linear map can take between two vector spaces. The symmetric subspace of the representation is the space of solutions to the constraint $\forall g \in G : \rho(g)v = v$, which provides the space of permissible kernels.

As was said previously, we are interested in the symmetries between graphs. Here we explore the automorphism symmetries of graphs and hence define what an automorphism group is. Further, we define a naturality constraint for a linear map as this governs a symmetry condition up to graph isomorphism.

Definition 3.9. Automorphism groups.
Let $\mathcal{X}$ be some mathematical object for which we can formulate the notion of homomorphism (or isomorphism). Then an automorphism of $\mathcal{X}$ is an isomorphism $\theta : \mathcal{X} \to \mathcal{X}$; in other words, it is a permutation of $\mathcal{X}$ which happens also to be a homomorphism satisfying

$$(x \circ y)\theta = x\theta \circ y\theta.$$

Let $\mathrm{Aut}(\mathcal{X})$ be the set of all automorphisms of $\mathcal{X}$. Then $\mathrm{Aut}(\mathcal{X})$ is a group, the automorphism group of $\mathcal{X}$. This can also be considered for a group rather than a general object $\mathcal{X}$ and therefore we can talk about the automorphism group of a group.

Definition 3.10. A *Graph isomorphism*, $\phi : G \to G'$ is a bijection between the vertex sets of two graphs G and G', such that two vertices u and v are adjacent in G if and only if $\phi(u)$ and $\phi(v)$ are adjacent in G'. This mapping is edge preserving, i.e. satisfies for all $(i, j) \in \mathcal{E}(G)$ (de Haan et al., 2020):

$$(i, j) \in \mathcal{E}(G) \iff (\phi(i), \phi(j)) \in \mathcal{E}(G').$$

An isomorphism from the graph to itself is known as an automorphism.

Definition 3.11. The *naturality* constraint for a linear map states that for a graph G and linear map $f_G : \rho(G) \to \rho'(G)$ the following condition holds for every graph isomorphism ϕ (de Haan et al., 2020):

$$\rho'(\phi) \circ f_G = f_{G'} \circ \rho(\phi).$$

Following the definitions it is noteworthy that when considering the permutation symmetries of a graph and looking at the automorphism group, there can be no automorphic mapping between a graph with four nodes and a graph with five nodes. This is due to there being no permutation of the four nodes features such that they yield the five nodes features.

In this paper we are interested in the symmetries of the symmetric group S_n. This constraint can be solved for different order tensor representations (Maron et al., 2018; Finzi et al., 2021). We present the space of linear layers mapping from k-order representations to k'-order representations in Figure 5.

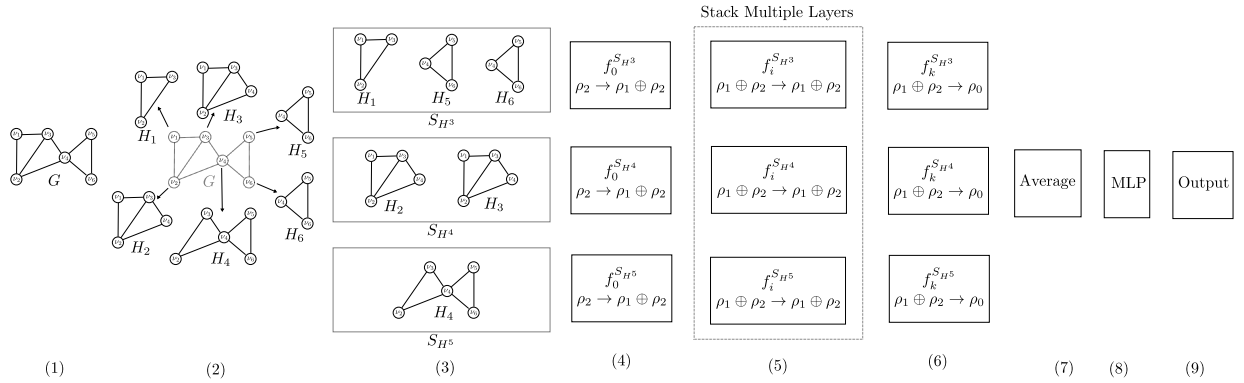

Figure 2: (1-2) Splitting the graph into sub-graphs. (3) Place sub-graphs into bags , where each bag holds sub-graphs of a specific size. (4) Process the bags of sub-graphs with an automorphism equivariant linear map. (5) Stack multiple layers each comprising of an automorphism equivariant mapping function. (6) Add a final automorphism equivariant mapping function. (7) Each automorphism group is averaged. (8) An MLP is used to update the feature space.

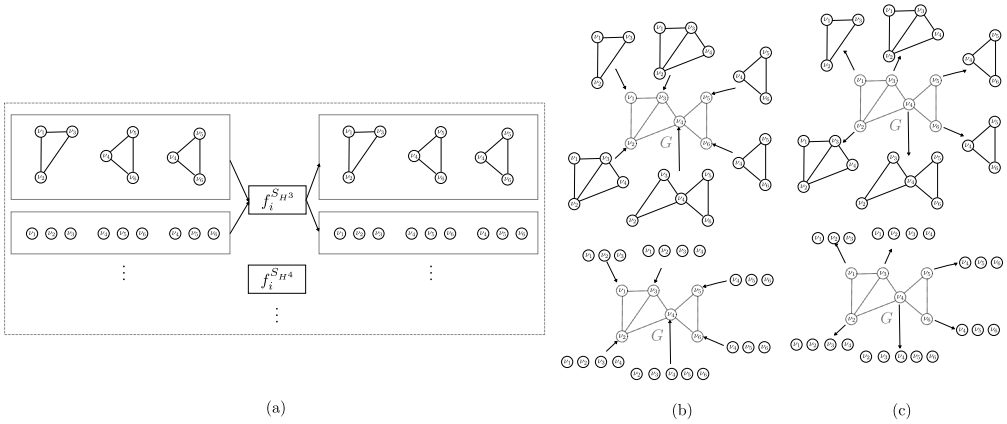

Figure 3: This figure breaks down what a single automorphism equivariant layer within our model looks like. In Figure 2 this corresponds to looking inside a single dashed box in (5). Here we see in (a) that the input to the layer is a vector space transforming under the group representation $\rho_1 \oplus \rho_2$ corresponding to graphs and sets as inputs. These processed by automorphism equivariant update functions f, where there is an $f^{S_{H^i}}$ for each automorphism group. (b) Following the automorphism equivariant update function we re-insert the vector space features back into their respective nodes in the original graph, and (c) re-extract back into the respective sub-graphs. This can be seen as a form of narrowing and promotion and allows information to propagate between sub-graphs.

3.2 Sub-graph Selection Policy

Sub-graphs can be extracted from a graph in a number of ways, by removing nodes, by removing edges, extracting connectivity graphs at nodes, or extracting connectivity graphs at edges to name a few. In this work we focus on k-ego network sub-graphs. These are sub-graphs extracted by considering the k-hop connectivity of the graph at a selected node and extracting the induced connectivity. The sub-graph selection policy of k-ego networks therefore extracts a sub-graph for each node in the original graph.

In this work we process graphs as bags of sub-graphs, where the notation of a bag of subgraphs was introduced by Bevilacqua et al. (2022). In general the size of the sub-graphs, $|H_n|$, extracted for a graph are not all the same size, and thus $|H_n|$ varies from sub-graph to sub-graph. We therefore go further than representing each graph as a bag of sub-graphs and represent each graph as a bag of bags of sub-graphs, where each bag of sub-graphs hold sub-graphs of the same size, i.e. S_{H^i} is the bag of sub-graphs

for sub-graphs of size $|H_n| = i$. The graph is therefore represented as the bag of bags of sub-graphs $G \equiv \{\{S_{H^i} \dots S_{H^k}\}\} \equiv \{\{\{\{H_1^i, ..., H_a^i\}\}, ..., \{\{H_1^k, ..., H_c^k\}\}\}\}$, for sub-graphs H, with bags of sub-graphs which are each of size $a, ..., c$ containing sub-graphs of sizes $i, .., k$ respectively.

In Chapter 4 we demonstrate how our choice of sub-graph selection policy improves the expressivity of the overall model. Further, in Chapter 4 and 5 we show that the choice of sub-graph selection allows the method to scale better than global approaches. Finally, in Chapter A.6 we show that using k-ego network sub-graphs yields sub-graphs which are typically much smaller than the original graph. This both feeds into the improved ability of our method to scale to larger graphs and provides a smaller more compact automorphism group compared to an approach such as node removed sub-graphs. As a result our method requires less parameterisation of the automorphism group than other automorphism equivariant approaches.

3.3 Automorphism Equivariant Graph Network Architecture

The input data represented as a bag of bags of sub-graphs has a symmetry group of both the individual sub-graphs and of the bags of sub-graphs. We construct a graph neural network that is equivariant to this symmetry. This can be broken down into three parts: (1) the automorphism symmetry of the bags of sub-graphs, (2) the permutation symmetry of sub-graphs within bags of sub-graphs and the original graph permutation symmetry, (3) sub-graph linear maps.

An overview of the architecture is detailed in Figure 2, where each component is described: (1-2) The first component of our SPEN model comprises of splitting the graph G into sub-graphs $H_1 \dots H_n$, for a graph with n nodes. For this we use a k-ego network policy extracting a sub-graph for each node in the input graph. (3) Secondly, we place sub-graphs $H_1 \dots H_n$ into bags $S_{H^i} \dots S_{H^j}$, where each bag holds sub-graphs of a specific size, with $i \dots j$ being the different sizes $|H|$ of sub-graphs. The extracted sub-graphs are used as fully-connected graphs with zero features for non-edges; this results in each bag of sub-graphs representing an automorphism group. Here it is worth noting that the figure shows three bags of sub-graphs or three automorphism groups, while in general there does not have to just be three automorphism groups and this can vary. (4) We then process the bags of sub-graphs with an automorphism equivariant linear map. This comprises of multiple separate functions f, with a different function processing each automorphism group, i.e. $f_0^{S_{H^3}}$ is the function mapping the automorphism group corresponding to the bag S_{H^3} of sub-graphs in layer 0. This is a map from a 2-order permutation representation, i.e. graphs, to the direct sum of a 1-order and 2-order permutation representation. (5) We then stack multiple layers each comprising of an automorphism equivariant mapping function. Each of the automorphism groups is updated with a function mapping from the direct sum of a 1-order and 2-order permutation representation to the direct sum of a 1-order and 2-order permutation representation. (6) The final layer in the model is again an automorphism equivariant mapping function were each automorphism group is mapped from the direct sum of a 1-order and 2-order permutation representation to a 0-order representation. (7) Each automorphism group is averaged. (8) An MLP is used to update the feature space.

3.3.1 Automorphism Symmetry

We have defined the sub-graph selection policy used, which results in the input graph, G, being represented as a bag of bags of sub-graphs, $\{\{S_{H^i} \dots S_{H^k}\}\}$. As each bag of sub-graphs holds sub-graphs of a different size, each forms a different automorphism group. Therefore, we have different feature spaces for different sub-graph sizes, i.e. $\rho(S_{H^i}) \neq \rho(S_{H^j})$. A linear layer acting on sub-graphs should therefore operate differently on sub-graphs from different automorphism groups. This is demonstrated in Figure 2 (4,5,6) by having different linear map f for each bag of sub-graphs ($f_i^{S_{H^3}}$, $f_i^{S_{H^4}}$, $f_i^{S_{H^5}}$). This is defined more rigorously through the concept of naturality. Given a linear layer mapping from a feature space acted upon by ρ_m to a feature space acted upon by ρ'_m for each sub-graph H a (linear) map can be detailed as $f_H : \rho_m(H) \to \rho'_m(H)$. However, given two isomorphic sub-graphs H and H' are the same graph up-to some bijective mapping, we want f_H and $f_{H'}$ to process the feature spaces in an equivalent manner. This naturality constraint is therefore similar to the global naturality constraint on graphs G, for sub-graphs, H:

$$\rho'(\phi) \circ f_H = f_{H'} \circ \rho(\phi). \tag{1}$$

This constraint (Equation 1) says that if we first transition from the input feature space acted upon by $\rho(H)$ to the equivalent input feature space acted upon by $\rho'(H)$ via an isomorphism transformation $\rho(\phi)$ and then apply $f_{H'}$ we get the same thing as first applying f_H and then transitioning from the output feature space acted upon by $\rho'(H)$ to $\rho'(H')$ via the isomorphism transformation $\rho'(\phi)$. Since $\rho(\phi)$ is invertible, if we choose f_H for some H then we have determined $f_{H'}$ for any isomorphic H' by $f_{H'} = \rho'(\phi) \circ f_H \circ \rho(\phi)^{-1}$. Therefore, for any automorphism $\phi : H \to H$, we get an equivariance constraint $\rho'(\phi) \circ f_H = f_H \circ \rho(\phi)$. Thus, a layer in the model must have for each automorphism group a map f_H that is equivariant to automorphisms. Therefore our choice of sub-graph selection policy, extracting bags of sub-graphs, aligns with the naturality constraint, in that we require a mapping function $f^{S_{H^i}}$ for each bag of sub-graphs.

3.3.2 Permutation Symmetries Within Bags of Sub-graphs

The order of sub-graphs in each bag of sub-graphs is arbitrary and changes if the input graph is permuted. This ordering comes from the need to represent the graph in a computer and the ordering is tracked through the use of concrete graphs. It would therefore be undesirable for the output prediction to be dependent upon this arbitrary ordering. This is overcome in the choice of insert and extract functions used to share information between sub-graphs demonstrated in Figure 3. At the end of a linear layer in our model node and edge features from the original graph can be represented multiple times, i.e. it occurs in multiple sub-graphs. We therefore average these features across sub-graphs through and insert and extract function and in doing this ensure the output is invariant to the ordering of sub-graphs in each bag.

3.3.3 Sub-graph Linear Maps

Each sub-graph has a symmetry group that is given by permutation of the order of nodes in the graph. This group is denoted S_n for a graph of n nodes. The group S_n acts on on the graph via $(\sigma \cdot A)_{ij} = A_{\sigma^{-1}(i)\sigma^{-1}(j)}$. Sub-graphs, H, therefore have a symmetry group $S_m \leq S_n$ and we are interested in constructing graph neural network layers equivariant to this symmetry group. The graph is an order-2 tensor and the action of the permutation group can be generalised to differing order tensors. For example, the set of nodes in a graph is an order-1 tensor. For the case of a linear mapping from order-2 permutation representations to order-2 permutation representations, the basis space was shown to comprise of 15 elements by Maron et al. (2018). Similarly, the constraint imposed by equivariance to the group of permutations can be solved for different order representation spaces and we provide an example of all mappings between representation spaces from order 0-2 in Figure 5. We are not restricted to selecting a single input-output order permutation representation space and can construct permutation equivariant linear maps between multiple representations separately through the direct sum $\oplus$. For example the direct sum of order 1 and 2 representations is given by $\rho_1 \oplus \rho_2$. We utilise this to build the linear mapping functions, $f^{S_{H^i}}$, shown in Figures 2 (4,5,6) and 3 (a) to use linear maps between feature space acted on by different representations ρ. An example of how this map breaks down into individual functions mapping from and to a graph feature space acted on by 2-order permutation representations and a set feature space acted on by 1-order permutation representations is provided in Figure 4. Further, what the bases utilised within each of these functions mapping between feature spaces acted upon by different order representations is given in Figure 5.

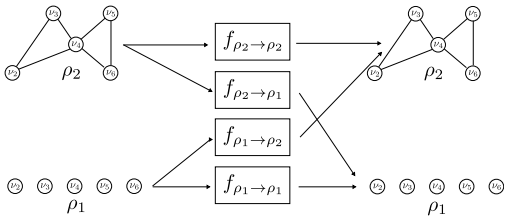

Figure 4: An example of what a function box in Figure 2 breaks down into. This example is for a function $f_i^{S_H}$ mapping from a representation $\rho_1 \oplus \rho_2 \to \rho_1 \oplus \rho_2$. This demonstrates there is a mapping function from ρ_2 to ρ_2, from ρ_2 to ρ_1, from ρ_1 to ρ_2, and from ρ_1 to ρ_1, as well as ρ_2 is a group representation for graphs and ρ_1 is a group representations for sets.

Due to the construction of a sub-graph the sub-graphs inherit node ids from the original graph. Therefore, a permutation of the order of the nodes in the original graph corresponds to an equivalent permutation of the ordering of the nodes in the sub-graphs. In addition, as the permutation action on the graph does not change the underlying connectivity, the sub-graphs exacted are individually unchanged up-to some isomorphism. Therefore, a permutation of the graph only permutes the ordering at which the sub-graphs are extracted.

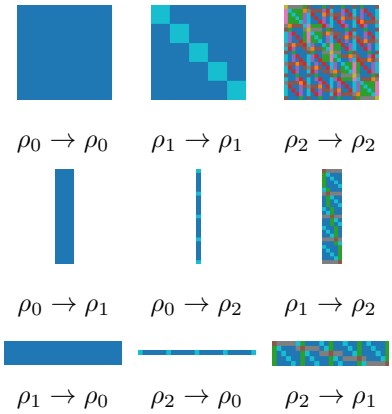

Figure 5: Bases for mappings to and from different order permutation representations, where ρ_k is a k-order representation. Each color in a basis indicates a different parameter. $\rho_2 \to \rho_2$ is a mapping from a graph to a graph, and has 15 learnable parameters. Further, there are mappings between different order representation spaces and higher order representation spaces.

3.4 Related Work

We have largely discussed the related methods to our work in Section 2. Despite this, we provide a more extensive explanation of some other methods in Appendix A.3 and demonstrate how some of these methods can be implemented within our framework in Appendix A.4.

4 Analysis of Expressivity and Scalability

In this section we study both the expressive power of our architecture following the theoretical analysis outlined in Bevilacqua et al. (2022) by its ability to provably separate non-isomorphic graphs and the scalability by its ability to process larger graphs that its predecessor.

4.1 WL Test and Expressive Power

The Weisfeiler-Lehman (WL) test (Weisfeiler & Leman, 1968) is a graph isomorphism test commonly used as a measure of expressivity in GNNs. This is due to the similarity between the iterative color refinement of the WL test and the message passing layers of a GNN. The WL test is a necessary but insufficient condition, which is not able to distinguish between all non-isomorphic graphs. The WL test was extended to the k-WL test, which provides increasingly more powerful tests that operate on k-tuples of nodes.

WL analogue for sub-graphs. One component of our model is the idea of operating on sub-graphs rather than the entire graph, more specifically our architecture operates on ego-network sub-graphs. We follow the theoretical analysis outlined in Bevilacqua et al. (2022) to verify that operating on sub-graphs will improve the expressive power of the base model within our automorphism equivariant model. Therefore, following Bevilacqua et al. (2022), we present a color-refinement variant of the WL isomorphism test that operates on a bag of sub-graphs.

Definition 4.1. The sub-graph-WL test utilises a color refinement of $c^{t+1}_{v,S} = \text{HASH}(c^t_{v,S}, \mathcal{N}^t_{v,S}, C^t_v)$, which is a simplification of that outlined in (Bevilacqua et al., 2022), where $\text{HASH}(\cdot)$ is an injective function, $\mathcal{N}^t_{v,S}$ is the node neighbourhood of v within the ego-network sub-graph S, and C^t_v is the multiset of v's colors across sub-graphs.

Theorem 4.2. *Sub-graph-WL is strictly more powerful than 1&2-WL.*

In Appendix A.2 we provide the proof of Theorem 4.2. Therefore, even for a simple 1-WL expressive function in the GNN, such as message passing, the model is immediately more expressive than 1&2-WL.

Comparing SPEN to the WL test. We have already shown that when considering a graph update function that operates on a bag of k-ego network sub-graphs, even if the update function itself has limited

expressivity, it is more expressive than 1&2-WL. SPEN utilises a natural permutation equivariant update function through operating on a bag of bags of sub-graphs. The naturality constraint of our model states that each automorphism group of sub-graphs should be processed by a different (linear) map. In addition, we utilise higher-dimensional GNNs. Both of these choices are expected to increase the expressive power of our model.

Proposition 4.3. *For two non-isomorphic graphs G^1 and G^2 sub-graph-WL can successfully distinguish them if (1) they can be distinguished as non-isomorphic from the multisets of sub-graphs and (2)* $\text{HASH}(\cdot)$ *is discriminative enough that* $\text{HASH}(c^t_{v,S^1}, \mathcal{N}^t_{v,S^1}, C^t_v) \neq \text{HASH}(c^t_{v,S^2}, \mathcal{N}^t_{v,S^2}, C^t_v)$.

This implies that despite the sub-graph policy increasing the expressive power of the model, it is still limited by the ability of the equivalent to the $\text{HASH}(\cdot)$ function's ability to discriminate between the bags of sub-graphs. The naturality constraint of our model processing each automorphism group with a different higher-dimensional GNN is therefore expected to increase the expressive power of our model over sub-graph methods utilising a MPNN.

Theorem 4.4. *SPEN is strictly more powerful than sub-graph MPNN.*

We demonstrate the claim of Theorem 4.4 similarly to Bouritsas et al. (2020); de Haan et al. (2020). We use a neural network with random weights on a graph and compute a graph embedding. We say the neural network finds two graphs to be different if the graph embeddings differ by an ℓ_2 norm of more than $\epsilon = 10^{-3}$ of the mean ℓ_2 norms of the embeddings of the graphs in the set. The network is said to be most expressive if it only finds non-isomorphic graphs to be different. We test this by considering a set of 100 random non-isomorphic non-regular graphs, a set of 100 non-isomorphic graphs, a set of 15 non-isomorphic strongly regular graphs,[2] and a set of 100 isomorphic graphs. Table 1 shows that a simple invariant message passing (GCN) (Kipf & Welling, 2016) as well as a simple invariant message passing model operating on sub-graphs (SGCN), which we created, are unable to distinguish between regular and strongly regular graphs. Further, it is shown that PPGN (Maron et al., 2019) can distinguish regular graphs but not strongly regular graphs, although a variant of PPGN that uses high order tensors should also be able to distinguish strongly regular graphs. On the other hand, our SPEN model is able to distinguish strongly regular graphs. Therefore, our model is able to distinguish non-isomorphic graphs that a sub-graph MPNN cannot and is strictly more powerful.

Table 1: Rate of pairs of graphs in the set of graphs found to be dissimilar in expressiveness experiment. An ideal method only find isomorphic graphs dissimilar. A score of 1 implies the model can find all graphs dissimilar, while 0 implies the model finds no graphs dissimilar.

Model	Random	Regular	Str. Regular	Isom.
GCN	1	0	0	0
SGCN	1	0	0	0
PPGN	1	0.97	0	0
SPEN	1	0.98	0.97	0

4.2 Scalability

Global permutation equivariant models of the form found by Maron et al. (2018) operate over the entire graph. They therefore scale with $\mathcal{O}(n^2)$, for graphs with n nodes. Our method operates on k-ego network sub-graphs where a sub-graph is produced for each node in the original graph. Our method therefore scales with $\mathcal{O}(nm^2)$, where m is the number of nodes in the k-ego network sub-graph. It is therefore clear that if $n = m$, theoretically, our method scales more poorly than global permutation equivariant models, although this would imply the graph is fully-connected and every sub-graph is identical. In this situation extracting sub-graphs is irrelevant and only 1 sub-graph is required (the entire graph) and hence if $n = m$ our method scales with that of global permutation equivariant models. The more interesting situation, which forms

[2]See `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`.

the majority of graphs, is when $n \neq m$. When $m \ll n$ our method scales more closely with methods that scale linearly with the size of the graph and it is for this type of data that our method offers a significant improvement in scalability over global permutation equivariant models.

We empirically show how SPEN and global permutation equivariant methods scale depending on the size of n and m by analysing the GPU memory usage of both models across a range of random regular graphs. We utilise random regular graphs for the scalability test as it allows for precise control over the size of the overall graph and sub-graphs. We compare the GPU memory usage of both models across a range of graph sizes with a sub-graph size of $m = 3$, 6,and 9. Through analysing the graphs in the TUDataset, which we make use of when experimenting on graph benchmarks, we note that the average sub-graph sizes range between 3 and 10 (see Table 3), justifying the choice of sub-graphs in the scalability tests. Figure 6 shows that the Global Permutation Equivariant Network (GPEN) (Maron et al., 2018) cannot scale beyond graphs with 500 nodes. On the other hand, our method (SPEN) scales to larger graphs of over an order of magnitude larger. In the situation where $m = 3$ GPEN can process graphs of size up to 500 nodes, while our SPEN can process graphs of size up to 10,000 nodes using less GPU memory.

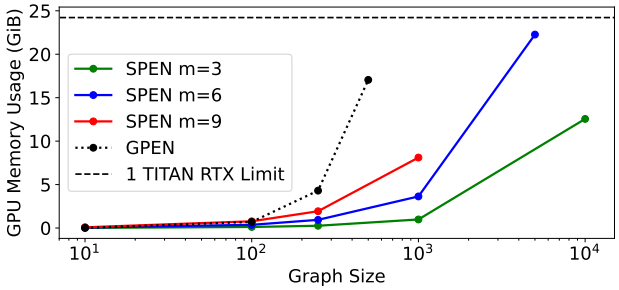

Figure 6: Computational cost of global permutation equivariant model (GPEN) and our (SPEN) model with a very similar number of model parameters for varying average size graphs. For this test we constructed random regular graphs of varying size using the NetworkX package (Hagberg et al., 2008). For SPEN sub-graphs were constructed using a 1-hop ego network policy. As is demonstrated by the log-axis, SPEN can process graphs an order of magnitude larger than global methods.

5 Experiments

5.1 Graph Benchmarks

We perform experiments with our method to compare to leading methods. For this we are looking to see how global permutation equivariant methods compare to our method? How our approach compares in terms of validation accuracy on real graph benchmarks with state-of-the-art methods? How our method scales wen compared with global permutation methods on real benchmark tasks?

TU Datasets. We tested our method on a series of 7 different real-world graph classification problems from the TUDatasets benchmark of (Yanardag & Vishwanathan, 2015). Five of these datasets originate from bioinformatics, while the other two come from social networks. We highlight some interesting features of each dataset. We note that both MUTAG and PTC are very small datasets, with MUTAG only having 18 graphs in the test set when using a 10% testing split. Further, the Proteins dataset has the largest graphs with an average number of nodes in each graph of 39. Also, NCI1 and NCI109 are the largest datasets having over 4000 graphs each, which one would expect to lead to less spurious results. Finally, IMDB-B and IMDB-M generally have smaller graphs, with IMDB-M only having an average number of 13 nodes in each graph. The small size of graphs coupled with having 3 classes appears to make IMBD-M a challenging problem.

Specific comparisons to prior work. We compare to a wide range of alternative methods, including sub-graph based methods, higher-dimensional GNN methods, and automorphism equivariant methods. We focus specifically on IGN (Maron et al., 2018) as this method uses an order-2 permutation equivariant tensor representation space for the linear map and is therefore the most similar to our base GNN model. The

Table 2: Comparison between our SPEN model and other deep learning methods. Larger mean results are better with the standard deviation around the mean given in (). Methods in comparison are: GDCNN (Zhang et al., 2018), PSCN (Niepert et al., 2016), DCNN (Atwood & Towsley, 2016), ECC (Simonovsky & Komodakis, 2017), DGK (Yanardag & Vishwanathan, 2015), DiffPool (Ying et al., 2018), CCN (Kondor et al., 2018), IGN (Maron et al., 2018), GIN (Xu et al., 2019), 1-2-3 GNN (Morris et al., 2019b), PPGN (Maron et al., 2019), LNGN (GCN) (de Haan et al., 2020), GSN (Bouritsas et al., 2020), SIN (Bodnar et al., 2021b), CIN (Bodnar et al., 2021a), and DSS-GNN (GC) (EGO) (Bevilacqua et al., 2022). All scores statistically indistinguishable (via Welch's ANOVA) from the highest mean in each benchmark have a gray background, whilst the highest mean values are in bold.

Dataset	MUTAG	PTC	PROTEINS	NCI1	NCI109	IMDB-B	IMDB-M
size	188	344	1113	4110	4127	1000	1500
classes	2	2	2	2	2	2	3
avg node #	17.9	25.5	39.1	29.8	29.6	19.7	13
				Results			
GDCNN	85.8 (1.7)	58.6 (2.5)	75.5 (0.9)	74.4 (0.5)	NA	70.0 (0.9)	47.8 (0.9)
PSCN	89.0 (4.4)	62.3 (5.7)	75 (2.5)	76.3 (1.7)	NA	71 (2.3)	45.2 (2.8)
DCNN	NA	NA	61.3 (1.6)	56.6 (1.0)	NA	49.1 (1.4)	33.5 (1.4)
DGK	87.4 (2.7)	60.1 (2.6)	75.7 (0.5)	80.3 (0.5)	80.3 (0.3)	67.0 (0.6)	44.5 (0.5)
CCN	91.6 (7.2)	70.6 (7.0)	NA	76.3 (4.1)	75.5 (3.4)	NA	NA
IGN	83.9 (13.0)	58.5 (6.9)	76.6 (5.5)	74.3 (2.7)	72.8 (1.5)	72.0 (5.5)	48.7 (3.4)
GIN	89.4 (5.6)	64.6 (7.0)	76.2 (2.8)	82.7 (1.7)	NA	75.1 (5.1)	52.3 (2.8)
PPGN v1	90.5 (8.7)	66.2 (6.5)	**77.2** (**4.7**)	83.2 (1.1)	81.8 (1.9)	72.6 (4.9)	50 (3.2)
PPGN v2	88.9 (7.4)	64.7 (7.5)	76.4 (5.0)	81.2 (2.1)	81.8 (1.3)	72.2 (4.3)	44.7 (7.9)
PPGN v3	89.4 (8.1)	62.9 (7.0)	76.7 (5.6)	81.0 (1.9)	82.2 (1.4)	73.0 (5.8)	50.5 (3.6)
LNGN (GCN)	89.4 (1.6)	66.8 (1.8)	71.7 (1.0)	82.7 (1.4)	83.0 (1.9)	74.8 (2.0)	51.3 (1.5)
GSN-e	90.6 (7.5)	68.2 (7.2)	76.6 (5.0)	83.5 (2.3)	NA	**77.8** (**3.3**)	**54.3** (**3.3**)
GSN-v	92.2 (7.5)	67.4 (5.7)	74.6 (5.0)	83.5 (2.0)	NA	76.8 (2.0)	52.6 (3.6)
SIN	NA	NA	76.5 (3.4)	82.8 (2.2)	NA	75.6 (3.2)	52.5 (3.0)
CIN	92.7 (6.1)	68.2 (5.6)	77.0 (4.3)	83.6 (1.4)	**84.0** (**1.6**)	75.6 (3.7)	52.7 (3.1)
DSS (EGO)	91.5 (4.9)	68.0 (6.1)	76.6 (4.6)	83.5 (1.1)	82.5 (1.6)	76.3 (3.6)	53.1 (2.8)
SPEN	**93.3** (**6.5**)	**71.3** (**9.7**)	74.8 (3.2)	**83.7** (**1.5**)	83.4 (1.2)	75.2 (3.1)	48.7 (2.0)

DSS-GNN model introduced in Bevilacqua et al. (2022) is tested on multiple different sub-graph policies and here we compare to the method utilising k-hop ego networks as this is the most similar variant to our method.

Implementation details and comparisons to prior work. We utilise a 1-hop ego network sub-graph policy for all of the experiments. Further, we use a base GNN model that maps between $\rho_1 \oplus \rho_2$ permutation equivariant tensor representation space, with the final layer mapping to a ρ_0 permutation equivariant tensor representation space. We constrain our model to be equivariant to the automorphism groups of the bags of sub-graphs. For this we do not build a model that is equivariant to every possible automorphism group, but rather every automorphism group created by our subgraph selection policy. This was a conscious choice due to it simplifying the construction of the automorphism equivariant layer. This means that the automorphism groups extracted are each a differing size permutation group.

For MUTAG, PTC, NCI1, and NCI109 we directly constrain the model to the automorphism groups of the bags of sub-graphs. For PROTEINS, IMDB-B, and IMDB-M there exists some bags of sub-graphs which comprise of a single sub-graph. As this would lead to no weight sharing between these sub-graphs and and any other sub-graphs we parameterize the automorphism constraint to bunch bags of sub-graphs which contain few sub-graphs. Further implementation and experimental details can be found in Appendix A.6.

Table 2 compares our SPEN model to a range of other methods on benchmark graph classification tasks from TUDatasets (Morris et al., 2020). We perform statistical significance analysis using Welch's ANOVA method for comparing multiple means with different variances. We consider the null hypothesis that the means are equal and to reject this null hypothesis the p value is required to be below 0.05. We therefore make bold all the methods indistinguishable from the state-of-the-art method, see Appendix A.7 for more details on the statistical significance analysis. Comparing out method (SPEN) to a higher-dimensional global permutation equivariant (IGN) demonstrates that our method significantly outperforms the base GNN model on four

datasets and is statistically indistinguishable on the remaining three. Table 2 also highlights that our method is statistically indistinguishable from the state-of-the-art result on six out of seven datasets, and achieves a larger mean on three of these datasets. This demonstrates that our method performs competitively across the range of datasets. The strong results produced by our method suggests that our framework's improved expressivity is beneficial for learning on graph classification tasks. Further discussion on the statistical significance is provided in Appendix A.7, which highlights that previous methods which claim state-of-the-art are not achieving this in a statistically meaningful way.

Figure 7 demonstrates that the improved scalability of our methods on regular graphs carries over onto graphs on real-world benchmarks. This demonstrates that our method offers a significant improvement in scalability over global permutation equivariant models.

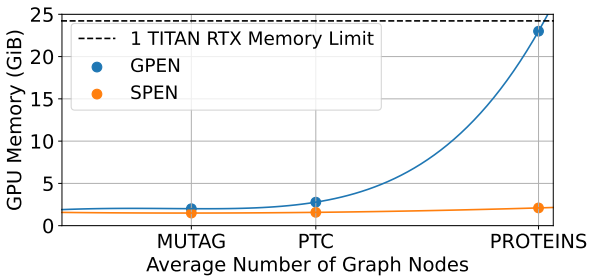

Figure 7: Computational cost of a global permutation equivariant model (GPEN) and our method (SPEN) with a very similar number of model parameters and batch size for datasets with varying average size graphs from the TUDatasets. For SPEN sub-graphs were constructed using a 1-hop ego network policy.

6 Future Work

From Table 2 it is clear that IMDB-M is a dataset for which our method has weaker performance. As stated in Section A.6.2 between hidden layers in our network, for the experiments in this paper, we only make use of order 1 and 2 representations. As it was shown by Maron et al. (2019) that increasing the order of the permutation representation increases the expressivity in line with the k-WL test, the expressivity of our method could be improved through the consideration of higher order permutation representations. Further, we parameterized the automorphism constraint in IMDB-M and therefore exploring alternative parameterizations of this constraint could lead to improved results.

7 Conclusion

We present a graph neural network framework for building models that operate on k-ego network sub-graphs that respects both the permutation symmetries of individual sub-graphs and is equivariant to the automorphism groups across bags of sub-graphs. The choice of sub-graph policy leads to a novel choice of automorphism groups for the bags of sub-graphs. The framework is more scalable than global higher-dimensional GNNs through the use of sub-graphs and we have both theoretically and experimentally demonstrated this. We have shown that SPEN is provably more expressive than the base higher-dimensional permutation equivariant GNN and sub-graph MPNNs through the choice sub-graph selection policy, permutation equivariant base GNN, and automorphism equivariant kernel constraint. We have provided theoretical analysis demonstrating the expressivity of the framework. Finally, we have shown that SPEN performs competitively across multiple graph classification benchmarks, achieving statistically indistinguishable accuracy compared to the state-of-the-art method on six out of seven datasets. We believe that our framework is a step forward in the development of graph neural networks, demonstrating theoretically provable expressivity, scalability, and experimentally achieving strong performances on benchmark datasets.

References

Marjan Albooyeh, Daniele Bertolini, and Siamak Ravanbakhsh. Incidence networks for geometric deep learning. *arXiv preprint arXiv:1905.11460*, 2019.

James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pp. 1993–2001, 2016.

Waiss Azizian and Marc Lelarge. Expressive power of invariant and equivariant graph neural networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=lxHgXYN4bwl`.

Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *ICLR 22*, 2022.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yu Guang Wang, Pietro Liò, Guido Montúfar, and Michael Bronstein. Weisfeiler and Lehman go cellular: CW networks. *arXiv preprint arXiv:2106.12575*, 2021a.

Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Lio, and Michael Bronstein. Weisfeiler and Lehman go topological: Message passing simplicial networks. *arXiv preprint arXiv:2103.03212*, 2021b.

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.

Leonardo Cotta, Christopher Morris, and Bruno Ribeiro. Reconstruction for powerful graph representations. *Advances in Neural Information Processing Systems*, 34, 2021.

Pim de Haan, Taco S Cohen, and Max Welling. Natural graph networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3636–3646. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/2517756c5a9be6ac007fe9bb7fb92611-Paper.pdf`.

Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv preprint arXiv:2104.09459*, 2021.

William Fulton and Joe Harris. *Representation theory: a first course*, volume 129. Springer Science & Business Media, 2013.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272, 2017.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

Jason Hartford, Devon Graham, Kevin Leyton-Brown, and Siamak Ravanbakhsh. Deep models of interactions across sets. In *International Conference on Machine Learning*, pp. 1909–1918. PMLR, 2018.

Nicolas Keriven and Gabriel Peyré. Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32:7092–7101, 2019.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proceedings of the International Conference on Machine Learning*, pp. 2747–2755, 2018.

Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. Covariant compositional networks for learning graphs. *arXiv preprint arXiv:1801.02144*, 2018.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In H. Wallach and H. Larochelle and A. Beygelzimer and F. d'Alché-Buc and E. Fox and R. Garnett (ed.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/bb04af0f7ecaee4aae62035497da1387-Paper.pdf`.

Christopher Morris, Gaurav Rattan, and Petra Mutzel. Weisfeiler and Leman go sparse: Towards scalable higher-order graph embeddings. *arXiv preprint arXiv:1904.01543*, 2019a.

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, 2019b.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *ICML Graph Representation Learning and Beyond (GRL+) Workshop*, 2020.

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

Pál András Papp, Karolis Martinkus, Lukas Faber, and Roger Wattenhofer. DropGNN: random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Gaurav Rattan and Tim Seppelt. Weisfeiler–leman, graph spectra, and random walks. *arXiv preprint arXiv:2103.02972*, 2021.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Erik Henning Thiede, Wenda Zhou, and Risi Kondor. Autobahn: Automorphism-based graph neural nets. *arXiv preprint arXiv:2103.01710*, 2021.

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

Fengli Xu, Quanming Yao, Pan Hui, and Yong Li. Automorphic equivalence-aware graph neural network. *Advances in Neural Information Processing Systems*, 34:15138–15150, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`.

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, 2015.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems*, pp. 4800–4810, 2018.

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

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.