

---

# Schur Nets: effectively exploiting local structure for equivariance in higher order graph neural networks

---

Qingqi Zhang<sup>1\*</sup>, Ruize Xu<sup>2\*</sup> and Risi Kondor<sup>1,2</sup>

<sup>1</sup>Computational and Applied Mathematics

<sup>2</sup>Department of Computer Science

University of Chicago

{qingqi,richard1xur,risi}@uchicago.edu

## Abstract

Several recent works have shown that extending the message passing paradigm to subgraphs communicating with other subgraphs, especially via higher order messages, can boost the expressivity of graph neural networks. In such architectures, to faithfully account for local structure such as cycles, the local operations must be equivariant to the automorphism group of the local environment. However, enumerating the automorphism groups of all subgraphs of interest and finding appropriate equivariant operations for each one of them separately is generally not feasible. In this paper we propose a solution to this problem based on spectral graph theory that bypasses having to determine the automorphism group entirely and constructs a basis for equivariant operations directly from the graph Laplacian. We show that empirically this approach can boost the performance of GNNs.

## 1 Introduction

Message passing neural networks (MPNNs) are the most popular paradigm for building neural networks on graphs [19]. While MPNNs have proved to be remarkably effective in a range of domains from program analysis [1] to drug discovery [21], a series of both empirical [3] and theoretical [43, 10] results have shown that the fact that MPNNs are based on just individual vertices sending messages to their neighbors limits their expressive power. This problem is especially acute in domains such as chemistry, where the presence or absence of specific small structural units (functional groups) directly influences the behavior and properties of molecules.

Recent papers proposed to address this issue by extending the message paradigm to also allow for message passing between vertices and edges [25] or between subgraphs, treated as single units [2, 16, 4, 49]. However, if the actual messages remain scalars, the added expressive power of these networks is relatively limited.

Another recent development is *higher order message passing networks*, where subgraphs communicate with each other via not just scalars, but more complex messages indexed by the individual vertices of the sending and receiving subgraphs [32, 35, 15]. For example, in an organic molecule, the internal state of a benzene ring (six carbon atoms arranged in a cycle) can be represented as a matrix  $T \in \mathbb{R}^{6 \times c}$ , in which the rows correspond to the individual carbons. If this benzene ring passes a message to a neighboring benzene ring with which it shares an edge, information relating to the two shared carbons can be sent directly to the same atoms in the second ring, whereas information relating to the other vertices has to be treated differently. Hypergraph neural networks [16, 48, 44]

---

\* denotes equal contribution.

and neural networks on more abstract mathematical structures such as simplicial complexes [7, 6] are closely related, since these also involve higher order generalizations of message passing between combinatorial objects interlocked in potentially complicated ways.

The theory of higher order MPNNs builds on the by now substantial literature on permutation equivariance in neural networks from Deep Sets [46], to various works studying higher order permutation equivariant maps [33], and is also related to the category theoretic approach expounded employed Natural Graph Neural Networks [13]. The recently introduced  $P$ -tensors formalism provides a flexible framework for implementing message passing in higher order MPNNs and enumerating all possible linear maps between subgraphs [22].

One aspect of higher order MPNNs that has garnered relatively little attention, however, is the interaction between the local graph structure and the equivariance constraint. Intuitively, the reason that GNNs must be equivariant to permutations is that when the same, or analogous, graphs are presented to the network with the only difference being that the vertices have been renumbered, the final output must remain the same. However, Thiede et al. [40] observed that at the local level enforcing full permutation equivariance is too restrictive. When considering specific, structurally important subgraphs such as paths or cycles, the operations local to such a subgraph  $S$  should really only be equivariant to the group of automorphisms of  $S$ , rather than all  $|S|!$  permutations of its vertices. The Autobahn architecture described in [40] explicitly takes the automorphism group of two specific types of subgraphs, cycles and paths, into account and defines a bespoke form of equivariant convolution on them, which is shown to improve performance on molecular datasets like ZINC [27].

The drawback of the Autobahn approach is that it requires explicitly identifying the automorphism group of each type of subgraph of interest, and crafting specialized equivariant operations for it based on its representation theory. For more flexible architectures leveraging not just cycles and paths but many other types of subgraphs this quickly becomes infeasible.

**Main contributions.** The main contribution of this paper is a simple algorithm based on spectral graph theory for constructing a basis of automorphism equivariant operations on any possible subgraph *without* explicitly having to determine its automorphism group, let alone derive its irreducible representations. The algorithm is based on imitating the structure of the group theoretical approach to equivariance, which we summarize in Section 3, but bypassing having to use actual group theory. Section 4 describes the algorithm itself and the resulting neural network operations, which we call *Schur layers*. In Section 5 we show that the introduction of Schur layers does indeed improve the performance of higher order graph neural networks on standard molecular benchmarks.

## 2 Background: equivariance with side information

Let  $\mathcal{G}$  be an undirected graph with vertex set  $V = \{1, 2, \dots, n\}$  and edge set  $E \subseteq V \times V$  represented by the adjacency matrix  $A \in \mathbb{R}^{n \times n}$ . Graph neural networks (GNNs) address one of two, closely related tasks: (a) given a function  $f^{\text{in}}: V \rightarrow \mathbb{R}$  on the vertices of  $\mathcal{G}$ , learn to transform it to another function  $f^{\text{out}}: V \rightarrow \mathbb{R}$ ; (b) given not just one graph, but an entire collection of graphs, learn to map each graph  $\mathcal{G}$ , or rather its adjacency matrix  $A$ , to a scalar or vector  $\psi(A)$  that characterizes  $\mathcal{G}$ 's structure.

In both cases, the critical constraint is that the network must behave appropriately under permuting the order in which the vertices are listed. The group of all permutations of  $\{1, 2, \dots, n\}$  is called the *symmetric group* of degree  $n$  and denoted  $\mathbb{S}_n$ . Applying a permutation  $\sigma \in \mathbb{S}_n$  to the vertices changes the adjacency matrix to  $A' = \sigma(A)$  with

$$A'_{i,j} = A_{\sigma^{-1}(i),\sigma^{-1}(j)}.$$

The basic constraint on algorithms that operate on functions on graphs is that if the input is transformed along with the numbering of the vertices,  $f_i^{\text{in}'} = f_{\sigma^{-1}(i)}^{\text{in}}$ , then the output must transform the same way,  $f_i^{\text{out}'} = f_{\sigma^{-1}(i)}^{\text{out}}$ . This property is called *equivariance*. Formally, denoting the mapping from inputs to outputs  $\phi: f^{\text{in}} \mapsto f^{\text{out}}$ , equivariance states that the action  $\sigma: f \mapsto f'$  of permutations on functions given by  $f'_i = f_{\sigma^{-1}(i)}$  must commute with  $\phi$ , that is,

$$\phi(\sigma(f^{\text{in}})) = \sigma(\phi(f^{\text{in}})) \tag{1}$$

for any  $\sigma \in \mathbb{S}_n$  and any input function  $f^{\text{in}} \in \mathbb{R}^V$ . In contrast, for networks that just learn embeddings  $\mathcal{G} \mapsto \psi(A)$ , the constraint is *invariance* to permutations, i.e.,  $\psi(A) = \psi(\sigma(A))$ . In practice, however,

the two cases are closely related because most graph embedding networks work with functions defined on the vertices, and then symmetrize over permutations at the very top of the network by using a readout function such as  $\psi(\mathcal{G}) = \sum_i f_i^{\text{out}}$ .

Enforcing the constraint (1) on the hypothesis space directly would be very limiting, effectively constraining GNNs to be composed of symmetric polynomials of  $f^{\text{in}}$  combined with pointwise nonlinearities. Graph neural networks cleverly get around this problem by using the adjacency matrix itself as *side information* to guide equivariance. For example, in classical (zeroth order) message passing neural networks (MPNNs), the output of each layer is itself a (vector valued) function on the graph, and in the simplest case the update rule from layer  $\ell$  to layer  $\ell + 1$  is

$$f_i^{\ell+1} = \eta\left(W \sum_{j \in \mathcal{N}(i)} f_j^\ell + \theta\right), \quad (2)$$

where  $\mathcal{N}(i)$  denotes the neighbors of node  $i$  in  $\mathcal{G}$ ,  $W$  and  $\theta$  are learnable weight matrices/vectors and  $\eta$  is a suitable nonlinearity. The fact that the summation only extends over the neighbors of vertex  $i$  induces a dependence of  $\phi$  on the adjacency matrix  $A$ . Hence it would be more accurate to write  $\phi$  as  $\phi_A$ , and the equivariance condition as

$$\phi_{\sigma(A)}(\sigma(f^{\text{in}})) = \sigma(\phi_A(f^{\text{in}})). \quad (3)$$

Since vertex permutations transform both  $f^{\text{in}}$  and  $A$ , when the vertices are permuted  $A$  can implicitly provide information about this to the algorithm allowing it to compensate, even if for fixed  $A$ ,  $\phi_A$  is not equivariant to all  $n!$  factorial permutations. Update rules such as (2) work in this way automatically. The adjacency matrix cannot carry information about permutations that leave it invariant however, so  $\phi_A$  must still be equivariant to the group of all such permutations, called the *automorphism group* of  $\mathcal{G}$  and denoted  $\text{Aut}(\mathcal{G})$  (or  $\text{Aut}(A)$ ). This observation about  $A$  acting as a source of side information that can reduce the size of the group that individual GNN operations need to be equivariant to and hence increase the expressive power of the network as a whole motivates much of the rest of this paper.

## 2.1 Higher order GNNs

Despite the great success of message passing networks, a long sequence of empirical as well as theoretical results [43, 35, 31, 14, 38] over the last several years have made it clear that the expressive power of algorithms based on simple update rules like (2) is severely limited. In response, the community has turned to extensions of the MPNN paradigm involving message passing not just from vertices to vertices but also between vertices and edges or between carefully selected subgraphs [19, 35, 6, 8]. While these networks maintain the local and equivariant character of earlier MPNNs, they can more faithfully reflect local topological information, and are particularly well suited to domains such as chemistry, where capturing local structures such as functional groups is critical.

In tandem, researchers have developed *higher order MPNN* architectures, where the output of individual edge or subgraph “neurons” are not just scalars, but vector or tensor quantities indexed by the vertices of the graph involved in the given substructure. For example, in the chemistry setting, if  $n$  is a titular neuron corresponding to a benzene ring in a molecule, the output of  $n$ , assuming we have  $C$  channels, might be a matrix  $T \in \mathbb{R}^{6 \times C}$  or a tensor  $T \in \mathbb{R}^{6 \times 6 \times C}$ , or  $T \in \mathbb{R}^{6 \times 6 \times 6 \times C}$ , etc.. In each of these cases the non-channel dimensions correspond to the six atoms making up the ring, and when  $n$  communicates with other subgraph-neurons, must be treated accordingly. Such higher order representations offer far greater expressive power since  $T$  can effectively represent not just information related to individual vertices, but also relations between pairs of vertices, triples, and so on. The general trend towards higher order message passing is also closely tied to hypergraph neural networks [16], “topological” neural networks and simplicial complex networks [7, 6].

Recently, [22] proposed a general formalism for describing such higher order architectures using so-called  $P$ -tensors. In this formalism, given a neuron  $n$  attached to a subgraph  $S$  with  $m$  vertices, if the output of  $n$  is a *zeroth order P-tensor*  $T$  with  $C$  channels then that means that it is simply a vector  $T \in \mathbb{R}^C$ . The elements of this vector are scalars in the sense that they give a global description of  $S$ , which is invariant to permutations of the vertices of  $S$ . If  $T$  is a *first order P-tensor* then that means that it is a matrix  $T \in \mathbb{R}^{m \times C}$  whose columns transform under permutations of  $S$  similarly to how  $f^{\text{in}}$  and  $f^{\text{out}}$  transform under global permutations of the full graph:

$$\sigma: T \mapsto T' \quad T'_{i,c} = T_{\sigma^{-1}(i),c} \quad \sigma \in \mathbb{S}_m. \quad (4)$$

If  $T$  is a *second order P-tensor*  $T \in \mathbb{R}^{m \times m \times C}$ , then each slice of it corresponding to a given channel transforms according to the second order action of the symmetric group, like the adjacency matrix:

$$\sigma: T \mapsto T' \quad T'_{i,j,c} = T_{\sigma^{-1}(i), \sigma^{-1}(j), c} \quad \sigma \in \mathbb{S}_m. \quad (5)$$

Continuing this pattern, a *k'th order P-tensor*  $T \in \mathbb{R}^{m \times m \times \dots \times m \times C}$  transforms under local permutations as

$$\sigma: T \mapsto T' \quad T'_{i_1, \dots, i_k, c} = T_{\sigma^{-1}(i_1), \dots, \sigma^{-1}(i_k), c} \quad \sigma \in \mathbb{S}_m. \quad (6)$$

[22] derived the general rules for equivariant message passing between such  $P$ -tensors in the cases that the sending and receiving subgraphs  $S$  resp.  $S'$  are (a) the same (b) partially overlap (c) are disjoint. In all these cases however it was assumed that  $T$  needs to be equivariant to all permutations of the underlying subgraphs  $S$  and  $S'$ . As we discussed above, this is an overly restrictive condition that limits the extent to which a higher order subgraph neural network can exploit the underlying topology. In the following sections we describe how the  $P$ -tensor framework can be extended so that the  $S \mapsto S'$  type of message passing steps (called *linmaps* in the  $P$ -tensors nomenclature) are equivariant to just  $\text{Aut}(S)$  rather than the full symmetric group  $\mathbb{S}_m$ .

### 3 Equivariance to local permutations: the group theoretic approach

Recall that a *representation* of a finite group  $G$  such as  $\mathbb{S}_m$  or  $\text{Aut}(S)$  is a (complex) matrix valued function  $\rho: G \rightarrow \mathbb{C}^{d_\rho \times d_\rho}$  such that the matrices multiply the same way as the corresponding group elements do

$$\rho(\sigma_2 \sigma_1) = \rho(\sigma_2) \rho(\sigma_1) \quad \sigma_1, \sigma_2 \in G.$$

Two representations  $\rho$  and  $\rho'$  are said to be *equivalent* if there is an invertible matrix  $Q$  such that  $\rho'(\sigma) = Q^{-1} \rho(\sigma) Q$  for all group elements. A representation of a finite group is said to be *reducible* if there is some invertible matrix  $Q$  that reduces it to a block diagonal form

$$\rho(\sigma) = Q^{-1} \left( \begin{array}{c|c} \rho_1(\sigma) & \\ \hline & \rho_2(\sigma) \end{array} \right) Q.$$

Some fundamental results in representation theory tell us that  $G$  only has a finite number of inequivalent irreducible representations (*irreps*, for short), these irreps can be chosen to all be unitary, and that any representation of  $G$  is reducible to some combination of them [39]. These facts give rise to a type of generalized Fourier analysis on finite groups, consisting of decomposing vectors that  $G$  acts on into parts transforming according to the unitary irreps of the group.

The most general approach to defining  $\text{Aut}(S)$ -equivariant maps for first, second, and higher order subgraph neurons uses exactly this machinery. In particular, defining permutation matrices as usual as

$$[P_\sigma]_{i,j} = \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{otherwise,} \end{cases}$$

dropping the channel indices w.l.o.g., and writing our  $P$ -tensors in vectorized form  $\bar{T} = \text{vec}(T) \in \mathbb{R}^{m^k}$ , we note that (4)–(6) can be written in a unified form

$$\bar{T}' = P^k(\sigma) \bar{T}$$

where  $P^k(\sigma)$  is the  $k$ -fold Kronecker product matrix  $P^k(\sigma) = P_\sigma \otimes P_\sigma \otimes \dots \otimes P_\sigma$ . Crucially,  $P^k(\sigma)$ , regarded as a function  $\text{Aut}(S) \rightarrow \mathbb{C}^{m^k \times m^k}$  is a unitary representation of the automorphism group.

According to representation theory,  $P^k$  must then be decomposable into a direct sum of irreps  $\rho_1, \dots, \rho_p$  of  $\text{Aut}(S)$  with corresponding multiplicities  $\kappa_1, \dots, \kappa_p$ . The same unitary matrix  $Q$  that accomplishes this can also be used to decompose  $\bar{T}$  into a combinations of smaller vectors  $(\mathbf{t}_j^i \in \mathbb{R}^{d_{\rho_i}})_{i,j}$ :

$$Q \bar{T} = \bigoplus_i \bigoplus_{j=1}^{\kappa_i} \mathbf{t}_j^i,$$

where each  $\mathbf{t}_j^i$  now transforms independently under the action of the group as  $\mathbf{t}_j^i \mapsto \rho_i(\sigma) \mathbf{t}_j^i$ . Alternatively, stacking all  $\mathbf{t}_j^i$  vectors transforming according to the same irrep  $\rho_i$  together in a matrix  $\mathbf{T}^i \in \mathbb{R}^{d_{\rho_i} \times \kappa_i}$ , we arrive at a sequence of matrices  $\mathbf{T}^1, \mathbf{T}^2, \dots, \mathbf{T}^p$  transforming as

$$\mathbf{T}^1 \mapsto \rho_1(\sigma) \mathbf{T}^1 \quad \mathbf{T}^2 \mapsto \rho_2(\sigma) \mathbf{T}^2 \quad \dots \quad \mathbf{T}^p \mapsto \rho_p(\sigma) \mathbf{T}^p. \quad (7)$$

Now it is very easy to see how to construct learnable linear operations that are equivariant to these actions: simply multiplying each  $\mathbf{T}^i$  from the right by an appropriate weight matrix  $W^i$ .

In summary, this general, group theoretic approach to constructing automorphism group equivariant linear maps between  $k$ 'th order  $P$ -tensors can be seen as a special case of e.g. [30, 41], and involves the following sequence of steps:

1. Find the unitary matrix  $Q$  that decomposes  $P^k(\sigma) = P_\sigma \otimes \dots \otimes P_\sigma$  into a direct product of irreducible representations of  $\text{Aut}(S)$ .
2. Use  $Q$  to decompose the input  $P$ -tensor into a sequence of matrices  $\mathbf{T}^1 \dots \mathbf{T}^p$  transforming as (7).
3. Multiply each  $\mathbf{T}^i$  by an appropriate learnable weight matrix  $W^i \in \mathbb{R}^{\kappa_i \times \kappa_i}$ .
4. Use the inverse map  $Q^{-1} = Q^\dagger$  to reassemble  $\mathbf{T}^1, \mathbf{T}^2, \dots, \mathbf{T}^p$  into the output  $P$ -tensor  $T^{\text{out}}$ .

Notwithstanding its elegance and power, the representation theoretic approach to implementing automorphism group equivariance has some major disadvantages, namely that it requires explicitly determining the automorphism group of each subgraph of interest, which is in and of itself a combinatorial problem, and explicitly finding its representations, which is also not trivial. The underlying mathematical structure however is important because it forms the basis to generalizing the approach to a much simpler framework in the next section:

1. We have a collection of (orthogonal) linear maps  $\{P^k(\sigma): U \mapsto U\}_{\sigma \in \text{Aut}(S)}$  (with  $U = \mathbb{R}^{m^k}$ ) that the neuron's operation needs to be equivariant to.
2.  $U$  is decomposed into an orthogonal sum of subspaces  $U = U_1 \oplus \dots \oplus U_p$  corresponding to the different irreps featured in the decomposition of  $P^k$ .
3. Each  $U_i$  is further decomposed into an orthogonal sum of subspaces  $U_i = V_1^i \oplus \dots \oplus V_{\kappa_i}^i$  corresponding to the different columns of the  $\mathbf{T}^i$  matrices.
4. The decomposition is such that the  $\{P^k(\sigma)\}$  maps fix each  $V_j^i$  subspace. Moreover, for a fixed  $i$ ,  $\{P^k(\sigma)\}$  acts the same way on each  $V_j^i$  subspace by  $\rho_i(\sigma)$ .
5. This structure implies that any linear map that linearly mixes the  $V_1^i, \dots, V_{\kappa_i}^i$  subspaces but does not mix information across subspaces with different values of  $i$  is equivariant.

## 4 Equivariance via spectral graph theory: Schur layers

Instead of the representation theoretical approach described in the previous section, for practical purposes we advocate a simpler way of implementing automorphism group equivariance based on just spectral graph theory. The cornerstones of this approach are the following two theorems. The proofs can be found in the Appendix.

**Theorem 1.** *Let  $G$  be a finite group acting on a space  $U$  by the linear action  $\{g: U \rightarrow U\}_{g \in G}$ . Assume that we have a decomposition of  $U$  into a sequence of spaces of the form*

$$U = U_1 \oplus \dots \oplus U_p$$

*such that each  $U_i$  is invariant under  $G$  in the sense that for any  $g \in G$  and  $v \in U_i$ , we have  $g(v) \in U_i$ . Let  $\phi: U \rightarrow U$  be a linear map that is a homothety on each  $U_i$ , i.e.,  $\phi(w) = \alpha_i w$  for some fixed scalar  $\alpha_i$  for any  $w \in U_i$ . Then  $\phi$  is equivariant to the action of  $G$  on  $U$  in the sense that  $\phi(g(u)) = g(\phi(u))$  for any  $u \in U$  and any  $g \in G$ .*

The representation theoretic result of the previous section actually corresponds to a refinement of this result involving a further decomposition of each  $U_i$  space into a sequence of smaller subspaces.

**Theorem 2.** *Let  $G$  and  $U$  be as in Theorem 1, but now assume that each  $U_i$  further decomposes into an orthogonal sum of subspaces in the form  $U_i = V_1^i \oplus \dots \oplus V_{\kappa_i}^i$  such that*

- (a) *Each  $V_j^i$  subspace is individually invariant by  $G$ ;*
- (b) *For any fixed value of  $i$ , the spaces  $V_1^i, \dots, V_{\kappa_i}^i$  are isomorphic and there is a set of canonical isomorphisms  $\iota_{j \rightarrow j'}^i: V_j^i \rightarrow V_{j'}^i$  between them such that*

$$g(\iota_{j \rightarrow j'}^i(v)) = \iota_{j \rightarrow j'}^i(g(v)) \quad \forall v \in V_j^i.$$

*Let  $\phi: U \rightarrow U$  be a map of the form*

$$\phi(v) = \sum_{j'} \alpha_{j,j'}^i \iota_{j \rightarrow j'}^i(v) \quad v \in V_j^i$$

*for some fixed set of coefficients  $\{\alpha_{j,j'}^i\}$ . Then  $\phi$  is equivariant to the action of  $G$  on  $U$ .*

In the matrix language of the previous section,  $U_1, \dots, U_p$  correspond to the  $T^1, T^2, \dots, T^p$  matrices, whereas the  $V_1^i, \dots, V_{\kappa_i}^i$  subspaces of  $U_i$  correspond to individual columns of  $T^i$ . For any fixed  $i$ , the  $(\alpha_{j,j'}^i)_{j,j'}$  scalars correspond to the individual matrix entries of the learnable weight matrix  $W^i$ .

#### 4.1 Automorphism invariance the simple way via spectral graph theory

It remains to find the actual systems of invariant subspaces that can be plugged into Theorems 1 and 2. The following lemma suggests such a system of subspaces can be found directly from the graph Laplacian, effectively bypassing the need for group representation theory.

**Lemma 1.** *Let  $S$  be an undirected graph with  $m$  vertices,  $\text{Aut}_S$  its automorphism group, and  $L$  its combinatorial graph Laplacian. Assume that  $L$  has  $p$  distinct eigenvalues  $\lambda_1, \dots, \lambda_p$  and corresponding subspaces  $U_1, \dots, U_p$ . Then each  $U_i$  is invariant under the first order action (4) of  $\text{Aut}_S$  on  $\mathbb{R}^m$ .*

*Proof.* Since  $\text{Aut}_S$  is a subgroup of the full group of vertex permutations, its action on  $\mathbb{R}^m$  is just  $\mathbf{v} \mapsto \sigma(\mathbf{v}) = P_\sigma \mathbf{v}$  with  $\sigma \in \text{Aut}_S$ .  $L$  is a real symmetric matrix, so its eigenspaces  $U_1, \dots, U_p$  are mutually orthogonal and  $U_1 \oplus \dots \oplus U_p = \mathbb{R}^m$ . Furthermore,  $\mathbf{v} \in U_i$  if and only if  $L\mathbf{v} = \lambda_i \mathbf{v}$ . By definition,  $\text{Aut}_S$  is the set of permutations that leave the adjacency matrix, and consequently the Laplacian invariant, so, in particular,  $P_\sigma L P_{\sigma^{-1}} = L$ . Therefore, for any  $\mathbf{v} \in U_i$  and any  $\sigma \in \text{Aut}_S$ ,

$$L \underbrace{P_\sigma \mathbf{v}}_{\sigma(\mathbf{v})} = P_\sigma L P_{\sigma^{-1}} P_\sigma \mathbf{v} = P_\sigma L \mathbf{v} = \lambda_i \underbrace{P_\sigma \mathbf{v}}_{\sigma(\mathbf{v})}$$

showing that  $\sigma(\mathbf{v}) \in U_i$  and hence that  $U_i$  is an invariant subspace.  $\square$

The following Corollary puts this lemma to use and gives a surprisingly easy way of creating locally automorphism equivariant neurons. We define a *Schur* layer as a neural network module that applies this operation to every instance of a given subgraph in the graph, for example, every single benzene ring in a molecule. Clearly, for the sake of global permutation equivariance, the weight matrices for the same subgraph  $S$  must be shared across the network.

**Corollary 1.** *Consider a GNN on a graph that involves a neuron  $\mathfrak{n}_S$  corresponding to a subgraph  $S$  with  $m$  vertices. Assume that the input of  $\mathfrak{n}_S$  is a matrix  $T \in \mathbb{R}^{m \times c_{in}}$ , the rows of which transform covariantly with permutations of  $S$  and  $c_{in}$  is the number of channels. Let  $L$  be the combinatorial Laplacian of  $S$ ,  $U_1, \dots, U_p$  be the eigenspaces of  $L$ , and  $M_i$  an orthogonal basis for the  $i$ 'th eigenspace stacked into an  $\mathbb{R}^{n \times \dim(U_i)}$  dimensional matrix. Then for any collection of learnable weight matrices  $W_1, \dots, W_p \in \mathbb{R}^{c_{in} \times c_{out}}$ ,*

$$\phi: T \mapsto \sum_{i=1}^p M_i M_i^\top T W_i \quad (8)$$

is a permutation equivariant linear operation.

The spectral approach also generalizes to higher order permutation equivariance, and in that case we can take advantage of the more refined two-level subspace structure implied by Theorem 2.

**Theorem 3.** *Let  $S$ ,  $L$  and the  $M_i$ 's be as in Corollary 1. Given a multi-index  $\mathbf{i} = (i_1, \dots, i_k) \in \{1, \dots, p\}^k$ , we define its type as the tuple  $\mathbf{n} = (n_1, \dots, n_p)$ , where  $n_j$  is the number of occurrences of  $j$  in  $\mathbf{i}$  and we define  $\mathcal{I}_{\mathbf{n}}$  as the set of all multi-indices of type  $\mathbf{n}$ . Assume that the input to neuron  $\mathfrak{n}_S$  is a  $k$ 'th order permutation equivariant tensor  $T \in \mathbb{R}^{m \times \dots \times m \times c_{in}}$ , as defined in Section 3. For any given  $\mathbf{i}$ , define the  $k$ 'th order eigen-projector*

$$\Pi_{\mathbf{i}} = \mathcal{P}_{\mathbf{i}}(M_{i_1}^\top \otimes M_{i_2}^\top \otimes \dots \otimes M_{i_k}^\top \otimes I): \mathbb{R}^{m \times \dots \times m \times c} \rightarrow \mathbb{R}^{m \times \dots \times m \times c}$$

where  $\mathcal{P}_{\mathbf{i}}$  is a permutation map that canonicalizes the form of the projection, as defined in the Appendix. Let  $T \circ W$  denote multiplying  $T$  by the matrix  $W$  only along its last dimension. Then for any collection of weight matrices  $\{W_{\mathbf{i}', \mathbf{i}} \in \mathbb{R}^{c_{out} \times c_{in}}\}$  the map

$$\phi: T \mapsto \sum_{\mathbf{n}} \sum_{\mathbf{i}' \in \mathcal{I}_{\mathbf{n}}} \sum_{\mathbf{i} \in \mathcal{I}_{\mathbf{n}}} \Pi_{\mathbf{i}'}^\top (\Pi_{\mathbf{i}}(T \circ W_{\mathbf{i}', \mathbf{i}}))$$

is a permutation equivariant map.

As mentioned earlier, this gives sufficient but not necessary conditions for equivariant maps w.r.t. the automorphism group. A discussion of potential gap between our approach and the group theoretical approach can be found in appendix D.



## 5 Experiments

To empirically evaluate *Schur* net, we implement the first order case in corollary 1 and compare it with other higher order MPNNs. Note that, the theorem only gives the equivariant maps to transform local representation on a subgraph, i.e.,  $\phi : T^{\text{in}} \mapsto T^{\text{out}}$  where  $T^{\text{in}} \in R^{m \times c_{\text{in}}}$  and  $T^{\text{out}} \in R^{m \times c_{\text{out}}}$  are representations of the subgraph. Another key component that is missing is the message passing between different subgraphs, which as mentioned in the previous section, is already given by [22] (*P*-tensor). In *P*-tensor framework, the most general linear equivariant message passing is defined based on an extension of equivariant maps by considering different domains. More details can be found in appendix E. We implement our *Schur* net based on the *P*-tensor framework but use *Schur* layer to transform local representation.

There are several design details about the architecture that are worth mentioning: (1) We chose cycles of lengths three to eight in most of the experiments and also added branched cycles to show our algorithm’s scalability. The reason to do so is in the chemical dataset we used, cycles are indeed the most important functional group to the property to be predicted, other subgraphs such as *m*-stars, and *m*-paths didn’t help the performance. (2) While having first-order representation on the cycles, we also maintain 0th-order representation (just scalar) on node and edges, as in [6, 22]. Those node and edge representations captures more elementary information about the graph, such as *k*-hop neighborhood, and is still important in higher order MPNNs. The representations pass messages with each other by intersection rule as defined in *P*-tensor framework. (3) As discussed in appendix G, we view *Schur* layer’s operation as a spectral convolution filter [5] applied to the subgraph and the number of channels to indicate how many times it expands the input feature to the output feature. The codes used to run our experiments can be found at <https://github.com/risilab/SchurNet>.

### 5.1 *Schur* layer improves over *Linmaps*

First of all, we want to show that considering more possible equivariant maps on the subgraph can indeed boost the performance. To this end, we performed controlled experiments to compare *Schur* layer with the equivariant maps w.r.t.  $S_m$  (following [22] we called it *Linmaps*). To make a fair comparison, we use the same architecture including MLPs and message passing defined by *P*-tensor and only replace the equivariant maps used in *Linmaps* by what is defined in corollary 1. We didn’t compare with Autobahn [40] because it didn’t use the *P*-tensor framework in the original paper and it’s hard for us to implement the convolution w.r.t. automorphism group for all the subgraphs we chose. But we note that for the case of the cycles (see table 4), the equivariant maps given by our approach are equal to that given by the group theoretical approach.

We’ll present the results on the commonly used molecular benchmark ZINC-12K [27] dataset in the main text. Results on TUDatasets as well as runtime comparison can be found in appendix H. The task on ZINC-12K is to regress the  $\log P$  coefficient of each molecule and the Mean absolute error (MAE) between the prediction and the ground truth is used for evaluation.

We design experiments to compare *Linmaps* and *Schur* layers in various scenarios, to showcase the robustness of the improvement. Table 1 shows under various message passing schemes between edges and cycles, *Schur* Layer consistently outperforms *Linmaps*. Those are indications of the added expressive power of extra equivariant maps in *Schur* layer, and they’re effective in various architectural design settings. Another ablation study regarding different cycle sizes can be found in appendix H.

Layer	Pass message when overlap $\geq 1$	Pass message when overlap $\geq 2$
<i>Linmaps</i> (baseline)	0.074 $\pm$ 0.008	0.074 $\pm$ 0.005
<i>Schur</i> layer	<b>0.070 <math>\pm</math> 0.006</b>	<b>0.071 <math>\pm</math> 0.003</b>

Table 1: Comparison between *Schur* Layer and *Linmaps* with different message passing schemes. The message passing scheme is a design choice in *P*-tensor framework, where the user can set when two subgraph’s representations communicate. The mostly common use case is to require at least *k* vertices in the intersection of two subgraphs for them to communicate. Experiments on ZINC-12k dataset and all scores are test MAE. Cycle sizes of {3,4,5,6,7,8,11} are used.

Then we studied the possibilities of adding *Schur* layer in different places of higher-order message passing scheme. In table 2, we observed that the more condensed the higher-order feature is, the

more improvement that the *Schur* Layer brings to us over *Linmaps*. We attribute the improvements of adding/replacing *Schur* layer in various scenarios over *Linmaps* the benefit gained from utilizing the subgraph structure and increased number of equivariant maps. We also tried other ways to use *Schur* layer, the result is summarized in G.

Model	Test MAE
<i>Linmaps</i>	0.071 $\pm$ 0.004
Simple <i>Schur</i> -Net	0.070 $\pm$ 0.005
Linmap <i>Schur</i> -Net	<b>0.068 <math>\pm</math> 0.002</b>
Complete <i>Schur</i> -Net	<b>0.064 <math>\pm</math> 0.002</b>

Table 2: An experiment demonstrating different ways of using *Schur* layer. "Complete *Schur* Layer" means that we apply *Schur* Layer on the incoming messages together with the original cycle representation. "Linmap *Schur* Layer" means that we just apply the *Schur* Layer on the aggregated subgraph representation feature. "Simple *Schur* Layer" means we directly apply *Schur* Layer on the subgraph features without any preprocessing. We can observe that as the subgraph information diversifies, *Schur* layer tends to decouple the dense information better and results in better performance. The test MAE of *Linmaps* in this table is taken from [22].

## 5.2 Flexibility

The other advantage of *Schur* layer is it computes the feature transform only based on the subgraph’s Laplacian, bypassing a difficult step of finding the automorphisms group and *irreps* of the subgraph it acts on. As discussed in the theory, *Schur* layer constructs equivariant maps only based on the subgraph’s Laplacian and is applicable directly to any subgraphs, making the implementation much easier when different subgraphs are chosen than the group theoretical approach. This allows it to easily extend to any subgraph templates that’re favorable by the user. To demonstrate this, we augment the subgraphs in the model by all the five and six cycles with one to three branches (including in total 16 non-isomorphic subgraph templates), comparing with baseline model where only the cycle itself is considered. Results can be found in appendix G.

## 5.3 Benchmark results

Finally, and most importantly, we compare the *Schur*-Net to several other higher-order MPNNs <sup>1</sup> on ZINC-12k dataset and OGB-HIV dataset [24] in table 3. We included baselines of (1) classical MPNNs: GCN[29], GIN [43], GINE [25], PNA[12], HIMP [17] (2) higher order MPNNs:  $N^2$ -GNN [15] <sup>2</sup>, CIN [6],  $P$ -tensors [22] (3) Subgraph-GNNs: DS-GNN(EGO+) and DSS-GNN(EGO+) [4], GNN-AK+ [49], SUN(EGO+)[18] (4) Autobahn [40].

We find that *Schur* Net ranked second on ZINC-12K and outperformed all other baselines on OGB-HIV dataset. This shows the expressivity of adding more equivariant maps by leveraging the subgraph topology. Furthermore, note that while  $N^2$ -GNN outperforms *Schur* Net on ZINC-12K, it’s a second-order model whereas in our experiment, we only used first-order activation. Also, the partial reason *Autobahn* didn’t perform well is in the original implementation, the author didn’t use  $P$ -tensor framework and used only a part of all possible linear message passing schemes. This shows to get the full power of equivariant maps w.r.t. subgraph automorphism group, we need to combine it with a general message passing framework between subgraphs as well.

## 6 Limitations

Unlike the representation theoretic approach, the spectral approach is not guaranteed to give the finest possible decomposition into invariant subspaces. Hence, equations like (8) do not necessarily define the most general possible automorphism-equivariant linear maps. In this paper we did not investigate from a theoretical point of view the potential gap in expressivity due to this fact. In general, being able to craft automorphism-equivariant layers of any order for any types of subgraphs opens up

<sup>1</sup>A discussion of related work can be found in appendix C.

<sup>2</sup>The works [32, 35] don’t have results in those two datasets.



Model	ZINC-12K MAE( $\downarrow$ )	OGB-HIV ROC-AUC(% $\uparrow$ )
GCN	$0.321 \pm 0.009$	$76.07 \pm 0.97$
GIN	$0.408 \pm 0.008$	$75.58 \pm 1.40$
GINE	$0.252 \pm 0.014$	$75.58 \pm 1.40$
PNA	$0.133 \pm 0.011$	$79.05 \pm 1.32$
HIMP	$0.151 \pm 0.002$	$78.80 \pm 0.82$
$N^2$ -GNN	<b><math>0.059 \pm 0.002</math></b>	-
CIN	$0.079 \pm 0.006$	$80.94 \pm 0.57$
P-tensors	$0.071 \pm 0.004$	$80.76 \pm 0.82$
DS-GNN (EGO+)	$0.105 \pm 0.003$	$77.40 \pm 2.19$
DSS-GNN (EGO+)	$0.097 \pm 0.006$	$76.78 \pm 1.66$
GNN-AK+	$0.091 \pm 0.011$	$79.61 \pm 1.19$
SUN (EGO+)	$0.084 \pm 0.002$	$80.03 \pm 0.55$
Autobahn	$0.106 \pm 0.004$	$78.0 \pm 0.30$
Schur-Net	$0.064 \pm 0.002$	<b><math>81.6 \pm 0.295</math></b>

Table 3: Comparison of different models on the ZINC-12K and OGBG-MOLHIV datasets.

a host of possibilities for making GNNs more powerful. Our experiments are limited to some of the simplest cases, such as exploiting cycles and edges. We also only used first order activations.

## 7 Conclusions

Enforcing equivariance to the automorphism group of subgraphs in higher order neural networks seemingly requires the use of advanced tools from group representation theory. This is likely a large part of the reason why automorphism-based architectures such as [40] are not used more commonly in practical applications. In this paper we have shown that a simpler approach based on spectral graph theory, following the same underlying logic as the group theoretic approach but bypassing having to enumerate all irreducible representations of the automorphism group, can lead to an architecture that is almost as expressive. Our algorithm, called Schur Nets, easily generalizes to higher order activations, as well as incorporating other types of side information such as vertex labels. In a practical setting, Schur Nets is easiest to deploy in conjunction with another algorithm like P-tensors [22] which hides the complexities of the higher order message passing component. The empirical performance of Schur Nets on the ZINC 12K dataset is superior to all other comparable (non-transformer based) architectures that we are aware of.

Our approach exposes heretofore unexplored connections between permutation equivariance and spectral GNNs such as [9, 23]. It also highlights the fact that while permutation equivariance is a fundamental constraint on graph neural networks, the key to building high performing, expressive GNNs is to use as much side information as possible, whether it be the adjacency matrix, vertex degrees or something else, to reduce the size of the group that the network needs to be equivariant to.

## Acknowledgements

We would like to thank Andrew Hands for many valuable pieces of advice and much practical assistance that he has given to the experimental side of this project. We also gratefully acknowledge the Toyota Technological Institute of Chicago and the Data Science Institute at the University of Chicago for making their computational resources available for our use, as well as NSF MRI-1828629 for additional infrastructure that was used in the course of this project.

## References

- [1] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. Learning to Represent Programs with Graphs. In *International Conference on Learning Representations (ICLR)*, 2018.
- [2] Emily Alsentzer, Samuel Finlayson, Michelle Li, and Marinka Zitnik. Subgraph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [3] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, and et al. Relational Inductive Biases, Deep Learning, and Graph Networks. 2018.
- [4] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant Subgraph Aggregation Networks. In *International Conference on Learning Representations (ICLR)*, 2022.
- [5] Deyu Bo, Xiao Wang, Yang Liu, Yuan Fang, Yawen Li, and Chuan Shi. A Survey on Spectral Graph Neural Networks. 2023.
- [6] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Liò, Guido Montufar, and Michael Bronstein. Weisfeiler and Lehman Go Cellular: CW Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [7] Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montúfar, Pietro Liò, and Michael Bronstein. Weisfeiler and Lehman Go Topological: Message Passing Simplicial Networks. *arXiv:2103.03212*, 2021.
- [8] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M. Bronstein. Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting. In *International Conference on Learning Representations (ICLR)*, 2020.
- [9] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral Networks and Locally Connected Networks on Graphs. In *International Conference on Learning Representations (ICLR)*, 2014.
- [10] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can Graph Neural Networks Count Substructures? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [11] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge Equivariant Convolutional Networks and the Icosahedral CNN. In *International Conference on Machine Learning (ICML)*, 2019.
- [12] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Velickovic. Principal Neighbourhood Aggregation for Graph Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [13] Pim de Haan, Taco S. Cohen, and Max Welling. Natural Graph Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [14] Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [15] Jiarui Feng, Lecheng Kong, Hao Liu, Dacheng Tao, Fuhai Li, Muhan Zhang, and Yixin Chen. Extending the Design Space of Graph Neural Networks by Rethinking Folklore Weisfeiler-Lehman. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [16] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph Neural Networks. *33rd AAAI Conference on Artificial Intelligence*, 2019.
- [17] Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical Inter-Message Passing for Learning on Molecular Graphs. *arXiv:2006.12179*, 2020.
- [18] Fabrizio Frasca, Beatrice Bevilacqua, Michael M. Bronstein, and Haggai Maron. Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [19] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. *International Conference on Machine Learning (ICML)*, 2017.
- [20] Martin Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Cambridge University Press, 2017.

- [21] Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules. *ACS Central Science*, 2018.
- [22] Andrew R. Hands, Tianyi Sun, and Risi Kondor. P-Tensors: A General Framework for Higher Order Message Passing in Subgraph Neural Networks. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024.
- [23] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep Convolutional Networks on Graph-Structured Data. *arXiv:1506.05163*, 2015.
- [24] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open Graph Benchmark: Datasets for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [25] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. Strategies for Pre-Training Graph Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2020.
- [26] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning (ICML)*, 2015.
- [27] John J. Irwin, Khanh G. Tang, Jennifer Young, Chinzorig Dandarchuluun, Benjamin R. Wong, Munkhzul Khurelbaatar, Yurii S. Moroz, John Mayfield, and Roger A. Sayle. ZINC20—A Free Ultralarge-Scale Chemical Database for Ligand Discovery. *Journal of Chemical Information and Modeling*, 2020.
- [28] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, 2015.
- [29] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [30] Risi Kondor and Shubhendu Trivedi. On the Generalization of Equivariance and Convolution in Neural Networks to the Action of Compact Groups. In *International Conference on Machine Learning (ICML)*, 2018.
- [31] Andreas Loukas. How Hard is it to Distinguish Graphs with Graph Neural Networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [32] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably Powerful Graph Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [33] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and Equivariant Graph Networks. In *International Conference on Learning Representations (ICLR)*, 2019.
- [34] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. TUDataset: A Collection of Benchmark Datasets for Learning with Graphs. *arXiv:2007.08663*, 2020.
- [35] 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 Thirty-Third AAAI Conference*, 2019.
- [36] Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance Through Parameter-Sharing. In *International Conference on Machine Learning (ICML)*, 2017.
- [37] B.E. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Springer New York, 2013.
- [38] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation Ratios of Graph Neural Networks for Combinatorial Problems. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [39] Jean-Pierre Serre. *Linear Representations of Finite Groups*. Springer-Verlag, 1977.
- [40] Erik Thiede, Wenda Zhou, and Risi Kondor. Autobahn: Automorphism-Based Graph Neural Nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

- [41] Erik Henning Thiede, Truong-Son Hy, and Risi Kondor. The General Theory of Permutation Equivariant Neural Networks and Higher Order Graph Variational Encoders. *arXiv:2004.03990*, 2020.
- [42] Boris Weisfeiler and Andrei Leman. The Reduction of a Graph to a Canonical Form and the Algebra Which Appears Therein. *Nauchno-Technicheskaya Informatsiya*, 1968.
- [43] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2019.
- [44] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. HyperGCN: A New Method for Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [45] Jiaxuan You, Jonathan Gomes-Selman, Rex Ying, and Jure Leskovec. Identity-Aware Graph Neural Networks. *arXiv:2101.10320*, 2021.
- [46] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R. Salakhutdinov, and Alexander J. Smola. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [47] Muhan Zhang and Pan Li. Nested Graph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
- [48] Ruochi Zhang, Yuesong Zou, and Jian Ma. Hyper-SAGNN: A Self-Attention Based Graph Neural Network for Hypergraphs. In *International Conference on Learning Representations (ICLR)*, 2020.
- [49] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness. In *International Conference on Learning Representations (ICLR)*, 2022.

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background: equivariance with side information</b>	<b>2</b>
2.1	Higher order GNNs . . . . .	3
<b>3</b>	<b>Equivariance to local permutations: the group theoretic approach</b>	<b>4</b>
<b>4</b>	<b>Equivariance via spectral graph theory: <i>Schur</i> layers</b>	<b>5</b>
4.1	Automorphism invariance the simple way via spectral graph theory . . . . .	6
<b>5</b>	<b>Experiments</b>	<b>7</b>
5.1	<i>Schur</i> layer improves over <i>Linmaps</i> . . . . .	7
5.2	Flexibility . . . . .	8
5.3	Benchmark results . . . . .	8
<b>6</b>	<b>Limitations</b>	<b>8</b>
<b>7</b>	<b>Conclusions</b>	<b>9</b>
<b>A</b>	<b>Additional details</b>	<b>14</b>
<b>B</b>	<b>Proofs</b>	<b>14</b>
B.1	Alternative Proof for Corollary 1 . . . . .	14
<b>C</b>	<b>Related works</b>	<b>15</b>
<b>D</b>	<b>Analysis of <i>Schur</i> layer</b>	<b>16</b>
<b>E</b>	<b>A brief overview about <i>P</i>-tensor framework</b>	<b>16</b>
E.1	Message passing between <i>P</i> -tensors with the same reference domain . . . . .	17
E.2	Message passing between <i>P</i> -tensors with the different reference domains . . . . .	17
<b>F</b>	<b>Group Representation Theory Background</b>	<b>18</b>
<b>G</b>	<b>Ways to Use Schur Neuron</b>	<b>20</b>
<b>H</b>	<b>More experiment results</b>	<b>21</b>
<b>I</b>	<b>Architecture and experiment details</b>	<b>22</b>

## A Additional details

The purpose of the permutation map  $\mathcal{P}_i$  in Theorem 3 is to remap the tensor product space  $\mathbb{R}^{\dim(U_{i_1}) \times \dots \times \dim(U_{i_k}) \times C}$  into a canonical form so that those dimensions that correspond to  $i_j = 1$  are mapped to the first  $n_1$  slots, those dimensions with  $i_j = 2$  are mapped to the second  $n_2$  slots, and so on. Given the permutation  $\tau: \{1, 2, \dots, k\} \rightarrow \{1, 2, \dots, k\}$  that canonicalizes the indices in this way,  $\mathcal{P}_i$  is just the corresponding permutation map, i.e.,

$$\mathcal{P}_i(x_1 \otimes \dots \otimes x_k \otimes x_c) = x_{\tau(1)} \otimes \dots \otimes x_{\tau(k)} \otimes x_c.$$

## B Proofs

**Proof of Theorem 1.** Let  $\{g \downarrow_{U_i}: U_i \rightarrow U_i\}_{g \in G}$  be the restriction of the action of  $G$  to  $U_i$ . Since  $G$  fixes each  $U_i$ , the action of  $G$  on the full space decomposes in the form

$$g(u) = \sum_{i=1}^p g \downarrow_{u_i}(u \downarrow_{U_i}) \quad u \in U.$$

Now since multiplication by scalars commutes with linear maps,

$$\phi(g(u)) = \sum_{i=1}^p \alpha_i g \downarrow_{u_i}(u \downarrow_{U_i}) = \sum_{i=1}^p g \downarrow_{u_i}(\alpha_i u \downarrow_{U_i}) = g(\phi(u))$$

for any  $u \in U$  and any  $g \in G$ . ■

**Proof of Theorem 2.** Each of the  $V_j^i$  subspaces is invariant, hence a homothety on each is an equivariant operation for the same reason as in Theorem 1. In addition, since  $G$  acts the same way on any pair of subspaces  $(V_j^i, V_{j'}^i)$ , any map of the form  $\xi_{j,j'}^i: V_j^i \rightarrow V_{j'}^i$  that is just a scaling is also equivariant. The composition of equivariant linear maps is an equivariant linear map, hence any map  $\phi$  of the given form is equivariant. ■

**Proof of Theorem 3.** For any multi-index  $\mathbf{i}$ , the subspace  $\text{Im}(M_{i_1}^\top \otimes M_{i_2}^\top \otimes \dots \otimes M_{i_k}^\top \otimes I_c)$  is invariant to permutations. Further after applying the permutation map  $\mathcal{P}_i$ , all such subspaces of the same type  $\mathbf{n}$ , the action of  $\mathbb{S}_k$  on all such subspaces will be the same. Hence we can directly apply Theorem 2 to prove this theorem. ■

### B.1 Alternative Proof for Corollary 1

Here we give a more direct proof for Corollary 1.

*Proof.* Without loss of generality, we assume  $c_{in} = c_{out} = 1$ . Since the  $M_i$ 's and  $\lambda_i$ 's are eigenspaces and eigenvectors of the Laplacian  $L$ ,  $L = M\Lambda M^T$ , where  $M = [M_1, \dots, M_p]$  and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_m)$ .

The neuron  $n_S$  is a linear transform  $\phi: R^m \rightarrow R^m$ , with  $\phi(T) = \sum_i^p M_i M_i^T T W_i = M V M^T T$ , where  $V = \text{diag}(W_1 I_{n_1}, \dots, W_p I_{n_p})$ . Here the  $W_i$ 's are just scalars and  $I_{n_i}$  is an identity matrix of the same size as the corresponding eigenspace.

Given a permutation  $\sigma \in S_n$ , we want to show that  $\phi_\sigma(\sigma \circ T) = \sigma \circ \phi_e(T)$ , which is the definition of equivariance. Note that  $\phi_\sigma$  actually depends on  $\sigma$ , since we're doing eigenvalue decomposition on the Laplacian of the subgraph transformed by  $\sigma$ . This is the key to proving equivariance. We use  $e$  to denote the identity permutation.

Now  $L^\sigma = P_\sigma L P_\sigma^T = M_\sigma \Lambda M_\sigma^T$ , where  $M_\sigma = P_\sigma M$ , while the eigenvalues remain invariant after permutation and eigenspaces are transformed in the same manner as  $L$ . Thus,

$$\begin{aligned} \phi_\sigma(\sigma \circ T) &= M_\sigma V M_\sigma^T P_\sigma T \\ &= P_\sigma M V M_\sigma^T P_\sigma^T P_\sigma T \\ &= P_\sigma \phi_e(T) \end{aligned}$$



which is the desired result. □

**Remark 1.** *The key to the proof is that the transform  $\phi$  depends on an object transformed with permutation  $\sigma$ , specifically, graph Laplacian  $L$ . And all we need is that the object  $L$  is transformed equivariantly with  $\sigma$ . We can also add side informatino such as node labels or degrees to further constrain the automorphism group and increase the number of distinct eigenvalues. One easy way to do that is to set  $L' = L + V$ , where  $V = \text{diag}(v_1, \dots, v_m)$  captures the node labels or degrees, and build the Schur neuron  $n_S$  from  $L'$ .*

## C Related works

Higher order MPNNs have become a popular research area in graph representation learning since the seminal work by [33], which characterized the space of equivariant map w.r.t.  $S_n$  for  $k$ 'th order tensors. The following works have mainly been divided into two streams, the one is based on  $k$ -Weisfeiler Leman test ( $k$ -WL test) [42, 20], for instances, [32] proposed  $k$ -order graph neural networks that is as powerful as  $k$ -WL test; [35] proposed  $k$ -GNN to approximate (or simulate)  $k$ -WL test; [15] further designed a  $(k, t)$ -FWL+ hierarchy that extends the  $k$ -WL test and implemented the corresponding neuron version. A common feature of these approaches is they all work on all  $k$ -tuples of vertices for  $k$ -th order neural networks and thus make their space and time complexity at least  $\theta(n^k)$ .

The other line of work is seeking ways to choose subgraphs in the graph, that are representative or contain important information, such as functional groups in a molecule. [7] chooses simplicial complex and [6] further extends this to any cell complex such as cycles. [22] incorporates any subgraph template and uses the equivariant maps defined [33] for local feature transform and devised a high order message passing scheme between those subgraphs, called  $P$ -tensor. This is arguably the most general framework in this line of work. This line of work is kind of independent of the  $k$ -WL test since the  $k$ -WL test is aimed to distinguish any pair of non-isomorphic graphs while chosen subgraphs are meant to work in some specific regions with domain prior. It's possible to still compare this kind of network with the distinguishing power of  $k$ -WL test, for example, [6] shows if they include cycles of size at most  $k$  (including nodes and edges), their network are as powerful as WL test. However, such comparisons are not very meaningful to indicate the expressive power of such a domain-specific approach. In our opinion, the ability to learn certain features (such as count cycles of certain sizes or learn some function related to specific functional groups) might be a better criterion. It'll be an interesting research direction to design suitable criteria for the expressive power of such higher order MPNNs in general.

Our work follows this line of work since we're choosing specific subgraphs to learn representation on. But we utilize the subgraph's automorphism group to devise more possible equivariant maps (none of the aforementioned methods are built on the automorphism group). The only other work to our knowledge that uses the automorphism group is Autobahn [40], which is the first to introduce equivariance to the automorphism group to GNNs. The difference between our work and it lines twofold: (1) Autobahn theoretically introduces equivariant maps w.r.t. automorphism group via generalized convolution and Cosets, which is another group theoretical approach to construct equivariant maps, while we're using the decomposition to *irreps* (2) More importantly, Autobahn didn't mention a practical approach to construct those equivariant maps explicitly, hindering the utilization of more diverse subgraphs. In their experiments, they only use cycles of length five and six and paths of length three to six and construct the equivariant maps potentially by hand. In contrast, we give a simple and efficient way to construct equivariant maps w.r.t. automorphism group by EVD, which is very general to be used by any subgraph. Though our approach doesn't give the full set of equivariant maps, experiments show improvement over the traditional approach where only equivariant maps w.r.t.  $S_n$  are used. We believe this algorithm, together with the group theoretical idea about decomposition  $R^{m^k}$  into stable subspaces and *irreps* are beneficial to the research community.

There is a "third" type of higher order GNNs that is called Subgraph GNN, which deviates from the original definition of higher order MPNNs but can still be considered as higher order network. In particular, node-based GNNs such as [47, 49, 4, 45, 18] associate each node with a subgraph (by deleting the node, marking the node or extracting the EGO-network) and do MPNN on each subgraph,

where they get the resulting representation  $X \in R^{n \times n}$  that can be considered as a 2nd order tensor representation on the original graph.

## D Analysis of Schur layer

First of all, we notice that the equivariant map characterized in 1 (or 3) may not be the full set of equivariant maps w.r.t.  $Aut_S$ . Because (1) In the decomposition of  $U = R^{m^k}$  to eigenspaces  $U = U_1 \oplus \dots \oplus U_t$ , while  $U_i$  is stable subspaces, it might not be *irreducible* and finer decomposition may be possible. In other words, there might be two irreps (isomorphic or not) corresponding to the same eigenvalue. (2) The maps defined by 1 didn't take into account the isomorphic subspaces corresponding to the same type of *irrep*, thus ignored the possible equivariant maps between  $V_j^i$  and  $V_{j'}^i$ . So, it is of interest to find out how much the gap would be between our approach and the group theoretical approach. We first look at some examples in the first-order case (see table 4).

Graph	$Aut_S$	# of distinct Eigenvalues (Schur Layer)	$\sum_i (\kappa_i)^2$ irreps approach	$\sum_i \kappa_i$
6-cycle	$D_6$	4	4	4
5-cycle	$D_5$	3	3	3
4-cycle	$D_4$	3	3	3
3-cycle	$D_3$	2	2	2
5-star	$S_4$	3	5	3
4-star	$S_3$	3	5	3
3-path	$S_2$	3	5	3
n-cliques	$S_n$	2	2	2
5-cycle with one branch	$S_2$	6	20	6
6-cycle with one branch	$S_2$	7	29	7

Table 4: Examples on EVD approach vs group theoretical approach towards # of equivariant maps w.r.t.  $Aut_S$ . To calculate the decomposition of  $R^m$  into *irreps*, one can calculate  $\kappa_i = (\phi | \chi_i)$  where  $\phi$  and  $\chi_i$  is the character of the first-order action and  $i$ th *irrep* respectively, and  $(|)$  is inner product. Another quick approach is to use  $\phi(\sigma) = \sum_k \kappa_k \chi_k(\sigma)$  and look at some examples of  $\sigma$  to determine what the  $\kappa_k$  should be.

We note that for cycles in the graph case, there's no gap. However, when we add branches to the cycle to make the automorphism group smaller, the multiplicities of the *irreps* increase, e.g., in 5-cycle with one branch case,  $S_2$  only have two 1-dimensional *irreps*: trivial and sign representation of permutation, and the first-order action decompose to 4 copies of trivial and 2 copies of sign representation, gives in total  $4 * 4 + 2 * 2 = 20$  possible equivariant maps. However, note that # of *irreps* (counting multiplicities given by  $\sum_i \kappa_i$ ) is equal to # of distinct eigenvalues, meaning that each eigenspace corresponds to an irreducible subspace and the decomposition of  $R^m$  provided by eigenspaces is indeed a decomposition to the *irreps* in all of the cases listed in the table. Therefore, the multiplicities are the key reason for the gap between our EVD approach and the group theoretical approach, since our EVD approach can't capture the isomorphic property between subspaces. In principle, it is possible to find out which eigenspace is isomorphic to which, but in our current implementation, we didn't take this into account because we found out that only considering cycles is enough for a very good performance in the datasets we used.

Generally, it's complicated to determine the gap between our EVD approach and the group theoretic approach, especially in higher-order cases (where merely determining the  $\kappa_i$ 's is tricky), so we leave this into feature exploration.

## E A brief overview about P-tensor framework

The  $P$ -tensor framework is a framework for linear equivariant maps w.r.t.  $S_n$  both between the same subgraph and across different subgraphs. It is built on the equivariant maps characterized by [33].

**Definition 1** (*P-tensors*). Let  $U$  be a finite set of atoms and  $D = (x_1, \dots, x_d)$  an ordered subset of  $U$ . We say that a  $k$ -th order tensor  $T \in \mathbb{R}^{d \times d \times \dots \times d}$  is a  $k$ -th order permutation covariant tensor (or *P-tensor* for short) with reference domain  $D$  if under reordering by  $\tau \in S_d$   $D$  it transforms to

$$[\tau \circ T]_{i_1, i_2, \dots, i_k} = T_{\tau^{-1}(i_1), \dots, \tau^{-1}(i_k)}.$$

### E.1 Message passing between *P*-tensors with the same reference domain

Consider the equivariant maps sending  $T$  on  $D$  to  $T^{\text{out}}$  on  $D$ . The space of equivariant maps w.r.t  $S_m$  is characterized by [33]:

**Proposition 1.** The space of linear maps  $\phi : \mathbb{R}^{dk_1} \rightarrow \mathbb{R}^{dk_2}$  that is equivariant to permutations  $\tau \in S_d$  is spanned by a basis indexed by the partitions of the set  $\{1, 2, \dots, k_1 + k_2\}$ .

Then the authors designed a straightforward way to write the maps explicitly. Specifically, for each partition of the set  $\{1, 2, \dots, k_1 + k_2\}$ , there are three parts that determines the equivariant map: (1) summing over specific dimensions or diagonals of  $T^{\text{in}}$  (2) transferring  $T^{\text{in}}$  to  $T^{\text{out}}$  by identifying indices of  $T^{\text{in}}$  with indices of  $T^{\text{out}}$  (3) broadcasting the result along certain dimensions of  $T^{\text{out}}$ . These three operations correspond to the three different types of sets that can occur in a given partition  $\mathcal{P}$ : (1) those that only involve the second  $k_2$  numbers, (2) those that involve a mixture of the first  $k_1$  and  $k_2$  and (3) those only involve the first  $k_1$  numbers. The type (1) - (3) corresponds to type (1) - (3) of operations with the dimensions in the sets.

For example, in the case  $k_1 = k_2 = 3$ , the  $\mathcal{P} = \{\{1, 3\}, \{2, 5, 6\}, \{4\}\}$  partition corresponds to (a) summing  $T^{\text{in}}$  along its first dimension (corresponding to  $\{4\}$ ) (b) transferring the diagonal along the second and third dimension of  $T^{\text{in}}$  to the second dimension of  $T^{\text{out}}$  (corresponding to  $\{2, 5, 6\}$ ) (c) broadcasting the result along the diagonal of the first and third dimensions (corresponding to  $\{1, 3\}$ ). Explicitly, this gives the equivariant map:

$$T_{a,b,a}^{\text{out}} = \sum_c T_{c,b,b}^{\text{in}}$$

See table 5 for another example for  $k_1 = k_2 = 2$  case.

$\mathcal{P}$	$\phi$	$\mathcal{P}$	$\phi$
$\{\{1\}, \{2\}, \{3\}, \{4\}\}$	$T_{a,b}^{\text{out}} = \sum_{c,d} T_{c,d}^{\text{in}}$	$\{\{2\}, \{1,3,4\}\}$	$T_{b,a}^{\text{out}} = T_{b,b}^{\text{in}}$
$\{\{1\}, \{2\}, \{3,4\}\}$	$T_{a,b}^{\text{out}} = \sum_c T_{c,c}^{\text{in}}$	$\{\{1,2,3\}, \{4\}\}$	$T_{a,a}^{\text{out}} = \sum_b T_{a,b}^{\text{in}}$
$\{\{1\}, \{2,4\}, \{3\}\}$	$T_{a,b}^{\text{out}} = \sum_c T_{c,b}^{\text{in}}$	$\{\{1,2,4\}, \{3\}\}$	$T_{a,a}^{\text{out}} = \sum_b T_{b,a}^{\text{in}}$
$\{\{1\}, \{2,3\}, \{4\}\}$	$T_{a,b}^{\text{out}} = \sum_c T_{b,c}^{\text{in}}$	$\{\{1,2\}, \{3,4\}\}$	$T_{a,a}^{\text{out}} = \sum_c T_{c,c}^{\text{in}}$
$\{\{2\}, \{1,4,3\}\}$	$T_{b,a}^{\text{out}} = T_{b,b}^{\text{in}}$	$\{\{1,3\}, \{2,4\}\}$	$T_{a,b}^{\text{out}} = T_{a,b}^{\text{in}}$
$\{\{1,3\}, \{2,4\}\}$	$T_{b,a}^{\text{out}} = \sum_c T_{c,b}^{\text{in}}$	$\{\{1,4\}, \{2,3\}\}$	$T_{b,a}^{\text{out}} = T_{b,a}^{\text{in}}$
$\{\{1,2,3,4\}\}$	$T_{a,a}^{\text{out}} = T_{a,a}^{\text{in}}$	$\{\{1,2,3,4\}\}$	$T_{a,a}^{\text{out}} = T_{a,a}^{\text{in}}$

Table 5: The  $\mathcal{B}(4) = 15$  possible partitions of the set  $\{1, 2, 3, 4\}$  and the corresponding permutation equivariant linear maps  $\phi : \mathbb{R}^{k \times k} \rightarrow \mathbb{R}^{k \times k}$ .

### E.2 Message passing between *P*-tensors with the different reference domains

If  $T^{\text{in}}$  has reference domain  $D_1$  and  $T^{\text{out}}$  has  $D_2$ , with  $D_1 \neq D_2$  and  $D_1 \cap D_2 \neq \emptyset$ . We could have more options corresponding to summing either over the intersection or over  $D_1$ , and broadcasting either over the intersection or over  $D_2$ . So the number of maps corresponding to partition of type  $(p_1, p_2, p_3)$  is  $2^{(p_1+p_3)}$ . Let  $D_1 \cap D_2 = d^\cap$ , the previous example would have the following maps in  $D_1 \neq D_2$  case:

$$T_{a,b,a}^{\text{out}} = \begin{cases} \sum_{c=1}^{d^\cap} T_{c,b,b}^{\text{in}} & a, b \leq d^\cap \\ 0 & \text{otherwise,} \end{cases} \quad (a \in \{1, \dots, d^\cap\}, c \in \{1, \dots, d^\cap\})$$

$$T_{a,b,a}^{\text{out}} = \begin{cases} \sum_{c=1}^{d^\cap} T_{c,b,b}^{\text{in}} & b \leq d^\cap \\ 0 & \text{otherwise,} \end{cases} \quad (a \in \{1, \dots, d_2\}, c \in \{1, \dots, d^\cap\})$$

$$T_{a,b,a}^{\text{out}} = \begin{cases} \sum_{c=1}^{d_1} T_{c,b,b}^{\text{in}} & a, b \leq d^\cap \\ 0 & \text{otherwise,} \end{cases} \quad (a \in \{1, \dots, d^\cap\}, c \in \{1, \dots, d_1\})$$

$$T_{a,b,a}^{\text{out}} = \begin{cases} \sum_{c=1}^{d_1} T_{c,b,b}^{\text{in}} & b \leq d^\cap \\ 0 & \text{otherwise,} \end{cases} \quad (a \in \{1, \dots, d_2\}, c \in \{1, \dots, d_1\})$$

## F Group Representation Theory Background

Here, we explain in detail the decomposition of  $k$ -th order permutation to *irreps* of  $S_m$ .

**Proposition 2.** Let  $\rho_\lambda, \lambda \vdash m$  be the irreps of  $S_m$ ,  $U = R^{m^k}$ ,  $\sigma \in S_m$  act on  $U$  in the manner of equation 6, and suppose this action contains irreps  $\rho_{\lambda_1}, \dots, \rho_{\lambda_p}$  with multiplicities  $\kappa_1, \dots, \kappa_p$ , we have:

$$U = U_1 \oplus U_2 \oplus \dots \oplus U_p, \quad U_i = \bigoplus_j^{\kappa_i} V_i^j$$

where the first part gives the canonical decomposition of  $U$  and the second part further decompose each  $U_i$  into irreducible subspaces, where  $V_i^j$  correspond to  $\rho_{\lambda_i}$ . In matrix form,

$$P_\sigma^{(k)} = M \begin{pmatrix} \rho_{\lambda_1}(\sigma) & 0 & \dots & 0 \\ 0 & \rho_{\lambda_2}(\sigma) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \rho_{\lambda_p}(\sigma) \end{pmatrix} M^T, \quad \text{for all } \sigma \in S_m$$

where  $M$  is the orthogonal transformation for basis and we abuse  $\rho_{\lambda_i}(\sigma)$  to denote also the matrix of  $i$ -th irrep in the decomposition.

If we take a closer look at  $M$ , we can find that since  $P_\sigma^{(k)}$  is under the standard basis of  $R^{m^k}$ , thus  $M$  is just the orthonormal basis of each  $U_i$  (thus each  $V_i^j$ ) combined together, we denote  $M = (M_1, \dots, M_p)$  with  $M_i$  a  $m^k \times (d_{\lambda_i} * \kappa_i)$  dimensional matrix, where  $d_{\lambda_i}$  is the degree of  $\rho_{\lambda_i}$ . Therefore,  $P_\sigma^{(k)} T$  becomes:

$$P_\sigma^{(k)} T = \underbrace{[M_1, M_2, \dots, M_p]}_2 \begin{pmatrix} \rho_{\lambda_1}(\sigma) & 0 & \dots & 0 \\ 0 & \rho_{\lambda_2}(\sigma) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \rho_{\lambda_p}(\sigma) \end{pmatrix} \underbrace{\begin{pmatrix} M_1^T \\ M_2^T \\ \vdots \\ M_p^T \end{pmatrix}}_{(1)} T \quad (9)$$

The part (1) is a generalized Fourier transform of  $T$  to its Fourier components (coordinates under

the orthonormal basis)  $\hat{T} = \begin{pmatrix} M_1^T T \\ M_2^T T \\ \vdots \\ M_p^T T \end{pmatrix} \triangleq \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_p \end{pmatrix}$ , and the part (2) is the *irreps*  $\rho_{\lambda_1}, \dots, \rho_{\lambda_p}$  act

independently on  $\hat{T}$  with each component  $B_i \mapsto \begin{pmatrix} \rho_{\lambda_i}(\sigma) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \rho_{\lambda_i}(\sigma) \end{pmatrix} B_i$ . Note that there're  $\kappa_i$

multiple of  $\rho_{\lambda_i}$  act the same on components of  $B_i$  correspond to  $V_i^j$ , with a slight abuse of notation, we can think  $B_i \in R^{d_{\lambda_i} \times \kappa_i}$  (instead of  $R^{(d_{\lambda_i} * \kappa_i) \times 1}$ ) and write this map as  $\rho_{\lambda_i}(\sigma) B_i$ , the fact that the map by *irrep*  $\rho_{\lambda_i}$  act independently on each column of  $B_i$  allow us to identify directly a set of equivariant maps by multiplying  $B_i$  with matrix  $W_i \in R^{\kappa_i \times \kappa_i}$  to the right, and using associativity of matrix multiplication:  $\rho_{\lambda_i}(\sigma)(B_i W_i) = (\rho_{\lambda_i}(\sigma) B_i) W_i$ . This gives in total of  $\sum_i \kappa_i^2$  independent equivariant maps, as stated in the main text. The last part of the above equation maps Fourier components to their original space. In short, we can write:

$$P_\sigma^{(k)} T = \sum_i M_i \rho_{\lambda_i}(\sigma) \underbrace{M_i^T T}_{B_i} \quad (10)$$

with the abuse of notation to rearrange elements in  $B_i$  mentioned above.

Then we present a detailed version of theorem 2 in the main text.

**Theorem 4** (Necessary and sufficient condition for equivariant map). *Let  $G$  be a finite group acting on a vector space  $U$  by the linear action  $\{g : U \rightarrow U\}_{g \in G}$  and assume the action can be decomposed into irreps  $\rho_1, \dots, \rho_p$  with multiplicities  $\kappa_1, \dots, \kappa_p$  and degree  $d_i$ :*

$$U = U_1 \oplus U_2 \oplus \dots \oplus U_p, \quad U_i = \bigoplus_{j=1}^{\kappa_i} V_j^i$$

Then  $\phi : U \rightarrow U$  is an equivariant map w.r.t. this group action if and only if  $\phi$  is of the form:

$$\phi(v) = \sum_{j'} \alpha_{j,j'}^i \tau_{j \rightarrow j'}^i(v) \quad \text{for } v \in V_j^i \quad (11)$$

for some fixed set of coefficient  $\{\alpha_{j,j'}^i : i \in [1, \dots, p], j, j' \in [1, \dots, \kappa_i]\}$ .

In matrix form, suppose the matrix of  $g$  is  $R_g$  under basis  $(e_i)$  and  $\dim(U) = n$ , and it decompose into:

$$R_g = M \begin{pmatrix} \rho_1(g) & 0 & \dots & 0 \\ 0 & \rho_2(g) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \rho_p(g) \end{pmatrix} M^T \quad (12)$$

$$= \sum_i M_i \rho_i(g) M_i^T \quad (13)$$

Then  $\phi : R^n \rightarrow R^n$  is equivariant if and only if it is of the form:

$$\phi(T) = M M^T T \begin{pmatrix} W_1 & 0 & \dots & 0 \\ 0 & W_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & W_p \end{pmatrix} \quad (14)$$

$$= \sum_i M_i \underbrace{M_i^T T W_i}_{B_i} \quad (15)$$

where  $T \in R^n$ ,  $W_i \in R^{\kappa_i \times \kappa_i}$  is the fixed coefficients and  $B_i$  rearrange to  $R^{d_i \times \kappa_i}$  when needed. The coefficients  $\alpha_{j,j'}^i$  corresponds to  $(W_i)_{j,j'}$ .<sup>3</sup>

*Proof. Sufficiency:* firstly,  $\alpha_{j,j'}^i \tau_{j \rightarrow j'}^i(v)$  is equivariant by definition of isomorphism map. And the equivariance of  $\phi$  follows from the fact that sum of equivariant maps are equivariant.

*Necessity:* consider  $\phi$  on  $V_j^i$  and let  $W = \text{Im}(\phi|_{V_j^i})$  the image of  $\phi$  on  $V_j^i$ . Since  $\phi \circ g = g \circ \phi$  and  $g(v) \in V_j^i$  for any  $v \in V_j^i$ , we have  $g(\phi(v)) = \phi(g(v)) \in W$ , so  $W$  is stable under the action. Thus we can decompose  $W$  into irreducible spaces  $W = W_1 \oplus \dots \oplus W_p$ . Apply this decomposition to  $\phi(v)$  for  $v \in V_j^i$ , we get:

$$\phi(v) = \phi_1(v) + \dots + \phi_p(v) \quad \text{where } \phi_i(v) \in W_i$$

In other words  $\phi_k = \text{Proj}_k \circ \phi$  where  $\text{Proj}_k$  is the projection of  $W$  onto  $W_k$  (uniquely defined by the direct sum decomposition). Then  $\phi_k : V_j^i \rightarrow W_k$  and  $\phi_k \circ g = g \circ \phi_k$ . By Schur's lemma [39], for  $\phi_k \neq 0$ , we must have  $W_k$  isomorphic to  $V_j^i$  and  $\phi_k(v) = \theta_k \tau_{V_j^i, W_k}(v)$ . Since only  $V_j^i$  is isomorphic to  $V_j^i$ , we have  $W_k = V_k^i$  and  $\phi_k(v) = \alpha_{j,k}^i \tau_{j,k}(v)$ , where  $\tau_{j,k}(v)$  is the isomorphic map sending  $V_j^i$  to  $V_k^i$ . Thus  $\phi(v) = \sum_k \phi_k(v) = \sum_k \alpha_{j,k}^i \tau_{j,k}(v)$  for  $v \in V_j^i$ .

Finally, by linearity and  $U = \bigoplus_{i,j} V_j^i$ , the only possible equivariant function  $\phi : U \rightarrow U$  is given by 11.

<sup>3</sup>Similar discussions could be found in Section 4 of [41], but without a proof for necessity.

**Connection to matrix form:** first note that the isomorphism  $\tau_{j \rightarrow j'}^i : V_j^i \rightarrow V_{j'}^i$  is given by:

$$\tau_{j \rightarrow j'}^i(u_{j,l}^i) = u_{j',l}^i$$

where  $\{u_{j,l}^i, l = 1, \dots, \kappa_i\}$  and  $\{u_{j',l}^i, l = 1, \dots, \kappa_i\}$  are orthonormal basis for  $V_j^i$  and  $V_{j'}^i$  respectively such that the action of  $g$  is associated with matrix  $\rho_i(g)$ . Therefore,

$$\begin{aligned} \phi(u_{j,l}^i) &= \sum_{j'} \alpha_{j,j'}^i \tau_{j \rightarrow j'}^i(u_{j,l}^i) \\ &= \sum_{j'} \alpha_{j,j'}^i (u_{j',l}^i) \end{aligned}$$

Thus the matrix of  $\phi$  under basis  $(u_{j,l}^i, i = 1, \dots, p, j = 1, \dots, \kappa_i, l = 1, \dots, d_i)$  is

$$\begin{pmatrix} W_1 & & & & \\ & \ddots & & & \\ & & W_2 & & \\ & & & \ddots & \\ & & & & W_p \\ & & & & & \ddots \end{pmatrix} \text{ (which is the same as } \begin{pmatrix} W_1 & 0 & \dots & 0 \\ 0 & W_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & W_p \end{pmatrix} \text{ with } B_i \text{ rear-}$$

range to  $R^{d_i \times \kappa_i}$ ) and  $M$  is just the matrix to transform standard basis  $(e_i)$  to basis  $(u_{j,l}^i)$ . □

## G Ways to Use Schur Neuron

While Schur neuron can be viewed as a generic transform on any subgraph's first-order activation, we provide some thoughts and possibilities to instantiate it.

In our experiment, we primarily view Schur neuron as a way to expand the feature of a subgraph. It can be viewed as an analogue to CNN's convolution filter and it's indeed a constrained version of spectral convolution filter [5] on the subgraph. In light of this, we call number of channels of Schur neuron as how many times it expand the output feature versus the input feature.

Besides, we observe that the linear equivariant map described [33] is a special case of Schur neuron's map. The two possible linear map in [33] is (1) identity map, corresponding to take all weights  $W_i = 1$  (2)  $T \rightarrow \Sigma_i T_i \mathbb{1}$ , where  $\mathbb{1}$  is all ones vector, corresponding to set  $W_1 = 1$ , and  $W_i = 0$  for  $i \neq 1$  since  $\mathbb{1}$  is eigenvector of any graph Laplacian with eigenvalue 0. Consequently, we suggest to keep the two basic but important linear maps and augment them with Schur neuron. Empirically we verified this by an experiment only vary the number of channels. In figure 1, we see that further increase the number of channels beyond 4 wouldn't give us performance gain since cycle 5 and 6 only has 3 and 4 distinct eigenvalues respectively.

Furthermore, we suggest that number of channels of Schur neuron be proportional or approximately equal to number of distinct eigenvalues of the subgraph it attach to. This is because the later is the number of independent linear maps for Schur neuron.

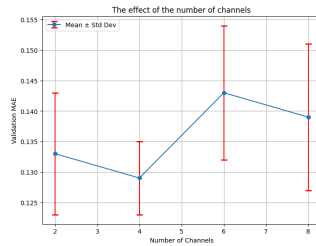


Figure 1: A study on num of channels in Schur layer. Cycle 5 and 6 are included.

Additionally, we're wondering if we can use *Schur* layer in place of MLP's to provide a local structure aware transform to subgraph activations. Therefore, we tried to replace the 2-layer MLP with 2-layer



*Schur* layer in the original *Linmaps* architecture. But it turns out this use case wasn't as effective as the use as learning new feature. Possibly because *Schur* layer learns feature itself while MLP transform the learned feature to some desired space.

Methods	Validation MAE
<i>Linmaps</i>	0.087
<i>Schur</i> layer (in place of MLP)	0.081
<i>Schur</i> layer (as learning new feature)	0.076

Table 6: Comparison about two use cases of *Schur* layer. Experiment on ZINC-12K.

Finally, regarding flexibility as discussed 5.2, in table 7, we see that adding cycles with branches could provide the *Schur* Net with more detailed topological information, thus improving the performance. And the code for adding this actually only requires a single definition of those templates without modification to the neural network.

Model	Validation MAE
<i>Schur</i> -Net on 5,6 cycles(baseline)	0.113 $\pm$ 0.008
<i>Schur</i> -Net on 5,6 cycles with up to three branches	<b>0.105 <math>\pm</math> 0.001</b>

Table 7: Flexibility of *Schur* layer. Experiments on ZINC-12k dataset. All other settings are the same. A smaller network than previous experiments was used.

## H More experiment results

We first start with some molecular datasets in TUDataset [34], which is a small but commonly used benchmark for GNN. In table 8, we see that *Schur* layer consistently improves over *Linmaps* in various bioinformatics and molecule datasets.

Dataset	<i>Linmaps</i>	<i>Schur</i> Layer
Proteins	74.7 $\pm$ 3.8	<b>75.4 <math>\pm</math> 4.8</b>
MUTAG	89.9 $\pm$ 5.5	<b>90.94 <math>\pm</math> 4.7</b>
PTC_MR	61.1 $\pm$ 6.9	<b>64.6 <math>\pm</math> 5.9</b>
NCII	82.1 $\pm$ 1.8	<b>82.7 <math>\pm</math> 1.9</b>

Table 8: Comparison of *Linmaps* and *Schur* Layer performance on TUDatasets. Numbers are binary classification accuracy.

In table 9, we compare the runtime of our *Schur* layer and *Linmaps*, which shows the extra computational cost of *Schur* layer wasn't significant while being able to use more equivariant maps and achieving better accuracy.

Dataset	<i>Linmaps</i>	<i>Schur</i> Layer
Zinc-12k	25.4s	27.6s
NCII	9.5s	11.5s

Table 9: Runtime per epoch with hyper-params num\_layers = 4, rep\_dim = 128, dropout = 0.0, batch\_size = 256, num of channels = 4, cycle\_sizes = 3,4,5,6,7,8. This shows *Schur* Layer didn't add much computational costs to *Linmaps* while being more expressive.

Another ablation study is perform on ZINC-12K dataset. In table 10, we see that *Schur* layer outperforms *Linmaps* when certain cycle sizes are considered, especially when only cycles 5 and 6 are chosen as subgraphs in the neural network.

Layer	cycle size {5,6}	cycle size {3,4,5,6}
<i>Linmaps</i> (baseline)	0.139 ± 0.007	0.103 ± 0.011
<i>Schur</i> layer	<b>0.114 ± 0.014</b>	<b>0.100 ± 0.009</b>

Table 10: Comparison between *Schur* Layer and *Linmaps* with different set of cycles chosen. Experiments on ZINC-12k dataset and all scores are test MAE.

## I Architecture and experiment details

In designing of our architecture, we design a message passing scheme similar to [22] and add Schur layer to each of the convolution layers. A visualization of our architecture can be found in figure ?? In each convolution layer, we maintain 0th-order node representation ( $h_v \in R^{|V|*dim}$ ) and edge representation ( $h_e \in R^{|E|*dim}$ ) where we assume the input graph  $G = (V, E)$ . We further maintain 1st-order representation on cycles ( $h_{D_k} \in R^{(k*c_k)*dim}$ ) where  $D_k$  is cycle of length k and  $c_k$  is the number of such cycle in  $G$ . We did message passing between node and edge representation the same as CIN [6]. The edge and cycle pass message with each other by transfer maps described by [22] and then the incoming message to cycles as well as the original cycle representation is transformed by Schur layer. When updating the edge representation and cycle representation, we combine the original representation with the incoming message by either concatenation or  $\epsilon$ -add described in GIN [43] and feed it into an MLP to get new representations.

**Hyperparameters** For experiments on ZINC-12K and OGB-HIV, we tune representation dimension in {32, 64, 128}, and experiment on cycles up to length 18. In the best-performing model, we use representation dimension 128 and cycle lengths from 3 to 8 and number-of-layers is always 4. For *Schur* layer, we tuned number of channels from 2 to 8 and found 2, 4 are a suitable choice when it is used as an augmentation to *Linmaps*. For MLPs, we either use 2 to 3 layers depending on how dense the input’s information is. We always use a batchnorm [26] after the Linear layer and then do ReLu activation. For training hyperparameters, we use an init learning rate of 0.01 and use ReduceLROnPlateau scheduler in PyTorch with patience of 10 or 30. We use Adam optimizer [28] for all trainings. We train for 500 or 1000 epochs depending on model size. In particular, in table 10, the models have representation dimension 32, so it is trained to only 500 epochs. All other models are trained 1000 epochs. A batch size of 256 is used for all models. All results are run at least three times to calculate the standard deviation.

For experiments on TUDataset (table 8), we chose a set of hyper-params by intuition (we didn’t tune them because this experiment is to demonstrate the effectiveness of *Schur* layer and compare *Schur* layer with *Linmaps* under the same experiment setting. we don’t aim to compare with Sota on this experiment). Hyper-params for both *Schur* Layer and *Linmaps*: num\_layers = 4, rep\_dim = 32,64, dropout = 0.5, batch\_size = 32,128, lr = 0.01, num of channels = {2,4}, cycle sizes = 3,4,5,6. We’re training with StepLR scheduler where reduce lr to 1/2 after every 50 epochs, with a total of 300 epochs. The hyperparams weren’t tuned, we just chose a smaller value for a smaller dataset and a bigger value for bigger datasets by rule-of-thumb. *Linmaps* is implemented on our own according to the description of  $P$ -tensor, then *Schur* Layer replaces *Linmaps* operation by *Schur* operation. We follow the evaluation strategy specified in [43].

**Used Compute Resources** Experiments are all run on one Tesla L4 from PyTorch Lightning Workspace and one NVIDIA RTX 4090. The code is implemented in PyTorch. For the running time, on the ZINC-12k dataset, it takes around  $8.53 \pm 1.2$  hours to finish training for 1000 epochs with 4 layers and 128 dimension representation on an NVIDIA RTX 4090. We’re running on a desktop with a Ryzen 7900 CPU and 64GB of RAM.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The claims made in the paper relate to specific properties of message passing algorithms. The claims that we make about our own algorithm are accurate.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes].

Justification: We have a section on limitations.

### 3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: For each Lemma and Theorem in the paper we provide a proof.

### 4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We document our experiments in detail, including the dataset, the architecture and parameters.

### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have included the link to a github repository containing all the files needed to reproduce the experiments.

### 6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Yes, we give all these details in the Appendix.

### 7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: For each of our results on the benchmarks datasets we provide the standard deviation of the performance.

### 8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We provide this information in the Appendix.

#### 9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have reviewed the code of ethics and our research conforms to it.

#### 10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA] .

Justification: Our work is foundational/theoretical in nature and we not foresee it having any direct social impacts. Indirectly, it could have impact because higher order automorphism-equivariant could be used in drug discovery, for example. Applications with negative social impacts are also possible, but a little more difficult to pinpoint. Space limitations prevented us from discussing this topic in more detail.

#### 11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA] .

Justification: The model itself does not pose such a risk.

#### 12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We provide a citation for every baseline model and benchmark dataset that we use. All assets were used in a way that conforms to their published licenses.

#### 13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA] .

Justification: We do not provide new assets.

#### 14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA] .

Justification: This research did not involve crowdsourcing or human subjects research.

#### 15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA] .

Justification: This research did not involve crowdsourcing or human subjects research.