Characterizing Schur Net's Expressivity Via Homomorphism Expressivity
NOTICE: THIS IS THE LATE VERSION AND WAS SUBMITTED AFTER THE DEADLINE

Michael Newman Fortunato
University of Chicago, Chicago, USA
michaelfortunato@uchicago.edu

Abstract

Graph spectral invariants and group-theoretic constructions each offer complementary pathways toward building expressive Graph Neural Networks (GNNs) [1] Recent work on higher-order message passing has shown that subgraphs communicating with subgraphs can capture complex local structures (e.g., cycles) by leveraging equivariance to the automorphism group of each local environment [1] However, explicitly enumerating these automorphism groups is often infeasible. In response to these challenges, Schur Net provides a spectral approach that circumvents direct group enumeration, yet its precise expressive power relative to group-based methods remains an open question [2]

Quite recently, a theoretical framework for measuring the expressivity of GNNs was introduced, dubbed *Homorphism Expressivity* [3] Crucially, Homomorphism Expressivity is the first measure of a GNN's expressivity that does not fall under a Weisfeiler-Lehman test [3] In this proposal, we layout a plan for theoretically measuring the expressivity of Schur Nets [2] using Homomorhism Expressivity [3] Specifically, we could characterize how Schur Net's spectral invariants compare to, and sometimes replicate, the subgraph-counting capabilities typically attributed to automorphism-based frameworks. Our analysis quantifies which families of graphs (and which substructures within them) each paradigm can homomorphism-count, and under what conditions they coincide or diverge. Furthermore, we extend our results to higher-order GNNs, clarifying how spectral techniques and group-theoretic constructions each handle more complex local symmetries. By illuminating these trade-offs, we aim to guide the design of hybrid GNN architectures that harness both the computational simplicity of spectral methods and the robust theoretical grounding of group-theoretic approaches.

Contents

1 Background

Designers of graph neural networks want their functions to treat a graph with one choice of node labels the same as the graph under another choice of vertex labels. Designers initially achieved this by ensuring that the first $\ell - 1$ layers were equivariant to permutations and then then ensured the last layer was invariant. This because most neural network φ are composed of as a series of layers $\varphi = \varphi^{\ell} \circ \varphi^{\ell-1} \circ ... \circ \varphi^{1}$, and it is known that the composition of two equivariant functions is equivariant, so networks φ invariant to permutations of their input graphs can be constructed a series of equivariant layers, until the last layer is made to be invariant.

However, just as the GNN community realized historically that invariant internal layers could be swapped for more expressive equivariant layers, it has recently been shown [1] that, in order to construct a neural network φ that is *invariant to the entire input graph $\mathcal{G}$*, internal layers of a GNN φ^{i} need not be equivariant to the entire input graph $\mathcal{G}$, but rather need just be equivariant to the automorphism group of a template graph $\mathcal{T}$.

As we will see in Section 3.1, the template graph $\mathcal{T}$ is chosen based on the problem domain. And so, designers using Autobahn based neural networks need to enumerate and manually choose the proper template graph. Schur Nets [2] was introduced to address this drawback in Autobahn. Schur Nets uses a spectral approach construct a Autobahn-style neural network. This allows Schur Nets to avoid having to explicitly state the template graph automorphism.

While Schur Nets approach to automorphism sub-graph equivariance avoids the overhead imposed by the purely group theoretic approach, Schur Nets' does not necessary yield an irreducible representation for each graph [4]. Therefore, Schur Net-based constructions of equivariance do not necessarily yield the most generalizable forms of automorphism (Autobahn-type) neural networks. Furthermore, the extent of the gap between Schur Nets and group theoretic based automorphism networks remains an open question and area of research.

One of the primary challenges to exploring the expressivity gap between Autobahn and Schur Nets is that Schur Nets is not amendable to Weisfeiler-Lehman [5] tests of expressivity. In fact, the authors of Schur Nets intuit that

"…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 … [than WL type tests]. It'll be an interesting research direction to design suitable criteria for the expressive power of such higher order MPNNs in general." [2]

This notion, that measuring the ability of GNN to count structures like sub-graph structures, has been recognized recently by other researchers as well, and has been formalized by [6] in a new framework for expressivity measurement, termed *Homomorphism Expressivity*. This initial work established Homomorphism Expressivity for sub-graph counting GNNs. Therefore, such a framework would *not* be directly applicable to Schur Nets, because it is a spectral invariant based GNN. However, following up on that work, Homomorphism Expressivity was extended to spectral invariant GNNs such as Schur Nets [3], giving us the tools necessary to compare pure group theoretic models (Autobahn) with spectral models (Schur Nets) using this framework.

In this paper we seek to kick of the theoretical exploration of this gap. First, we expound both the Autobahn and Schur Nets construction by formulating both approaching on example graphs. Not only does this formulation demonstrate this author's understanding and faculty with both frameworks[1], but also suggests a intuitive understanding of both the characters of the frameworks, which we will enforce by making qualitative observations of both behaviors.

With these preliminaries and intuitions established, we finally get to the main contribution of this paper, which is a formal treatment and classification of both frameworks under Homomorphism Expressivity [3]. First, we show that, in the language of *Homomorphism Expressivity*, Schur Nets can be described as a *spectral invariant graph neural network*, and that Autobahn can be classified as a *sub graph counting graph neural network*. Next, using the main result from [3], we compute Schur Nets' Homomorphism Expressivity and compare it to Autobahn.

2 Preliminaries

In this paper, if we refer to a group action of the symmetric group on a k-ranked tensor without mentioning the particular action, assume it is the action defined below, dubbed *the permutation action*.

Definition 2.1 *(Permutation action on kth order tensors)* Let $\mathbf{T} \in \mathbb{R}^{n^k}$ be kth order (rank) tensor. Let $\sigma \in \mathbb{S}_n$ be a permutation in the symmetric group $\mathbb{S}_n$. Then we say that the *action of the permutation group* $\mathbb{S}_n$ *on* $\mathbf{T}$, also called the permutation action on $\mathbf{T}$, is

$$\sigma \cdot \mathbf{T}$$
$$[\sigma \cdot \mathbf{T}]_{i_1,\dots i_k} = [\mathbf{T}]_{\sigma^{-1}(i_1),\dots,\sigma^{-1}(i_k)} \tag{1}$$

Remark 2.2 For a given graph with n nodes, represented by the rank two tensor $A \in \{0,1\}^{n\times n}$, we say that $\mathbb{S}_n$ permutes the graph by the action defined in Definition 2.1. Conceptually, permuting the graph A by Definition 2.1 can is equivalent to relabelling the nodes of the graph.

Graph neural network designers want their models to be invariant to all permutations on the input graph $\mathcal{G}$.

Definition 2.3 *(Permutation Invariance)* Let $f : \mathbb{R}^{n^{k_1}} \to \mathbb{R}^{n^{k_2}}$ be a function. We say that f *is permutation invariant to any permutation* if, for any input $A^{n^{k_1}}$,

$$f(\sigma \cdot A) = \sigma \cdot f(A) \tag{2}$$

for any $\sigma \in \mathbb{S}_n$.

Definition 2.4 *(Permutation Equivariance)* Similarly, we say that $f : \mathbb{R}^{n^{k_1}} \to \mathbb{R}^{n^{k_2}}$ is *permutation equivariant* We say that f *is permutation invariant to any permutation* if, for any input $A^{n^{k_1}}$,

$$f(\sigma \cdot A) = \sigma \cdot f(A) \tag{3}$$

for any $\sigma \in \mathbb{S}_n$.

Corollary 2.5 *(An Equivalent Definition of Permutation Equivariance)* Note that the permutation action of $\mathbb{S}_n$ given by Definition 2.1, immediately implies a homomorphism between $\mathbb{S}_n$ and $\mathrm{GL}(n,\mathbb{R})$. This can be used to redefine permutation equivariance when considering linear functions. Let $f : \mathbb{R}^n \to \mathbb{R}^n$, so f is in $\mathrm{GL}(n,\mathbb{R})$ and can be written as the matrix $L \in \mathbb{R}^{n\times n}$. Next, let $\rho : \mathbb{S}_n \to \mathrm{GL}(n,\mathbb{R})$, then we can say that $f \in \mathrm{GL}(n,\mathbb{R})$ is permutation equivariant if and only if

$$\rho(\sigma) \star f = f \star \rho(\sigma) \tag{4}$$

for any σ, where $\star$ is the binary operation of the group $\mathrm{GL}(n,\mathbb{R})$. Note that $\star$ is just matrix multiplication if we represent f and $\rho(\sigma)$ with matrices. One can also that that f is permutation equivariant to $\mathbb{S}_n$ if f *commutes* with every element of $\mathbb{S}_n$.

In general, I believe a group action $\varphi : G \times \mathcal{X} \to \mathcal{X}$ always implies a homomorphism between G and the group of functions which map $\mathcal{X}$ to $\mathcal{X}$ under composition (sometimes termed the automorphism group). It is sometimes useful to think of permutation equivariance for a given internal layer φ^i this way as it generalizes its particular input.

[1]Facetious

Definition 2.6 *(Automorphism of a graph $A \in \{0,1\}^{n\times n}$)* Let $\mathcal{G}$ be a graph with n nodes. Let the adjacency matrix $A \in M(2,\mathbb{R})$ be $\mathcal{G}$'s representation. Given the representation A, consider the set of permutations in $\mathbb{S}_n$, that, when acting on $\mathcal{G}$ via Definition 2.1 leave $\mathcal{G}$ unchanged. That is the set

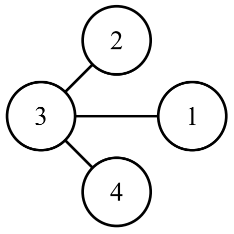
$$\{\sigma \in \mathbb{S}_n \mid \sigma \cdot A = A\}. \tag{5}$$

We denote this set of permutations by $\mathrm{Aut}(\mathcal{G})$, that is the set of permutations that leave $\mathcal{G}$ unchanged.

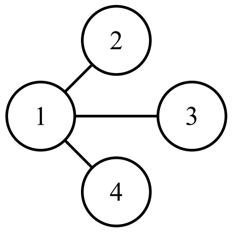
Example 2.7 *(4 node graphs)* It was not initially clear to me why one would be interested in the Automorphism group of a graph at all. Furthermore, For any given graph $A \in \{0,1\}^{n\times n}$, that $\mathbb{S}_n$ was not always essentially e $\mathrm{Aut}(A)$. Clearly though $\mathrm{Aut}(A) \neq \mathbb{S}_n$. We can see this by considering the graph And then considering the permutation $\sigma = (4\ 1) \in \mathbb{S}_n$.

We note that $\sigma \cdot A \neq A$, and so $\sigma \notin \mathbb{S}_n$.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{vs.} \quad \sigma \cdot A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$



(a) The Graph A Given By (6)



(b) The Graph $\sigma \cdot A$

Figure 1: $A \neq \sigma A$ when $\sigma = (3\ 1)$

Of course, A and $\sigma \cdot A$ *look* the same, and that is because their node-edge relations are the same, so invariance over $\mathbb{S}_n$ is typically what we want. *However*, just as $\mathbb{S}_n$ captures symmetries on the entire graph A, automorphisms, capture symmetry up to *sub-graphs*, so it is a finer filter. What I mean is that, notice how the nodes in sub-graph $\{(v_3, v_4), (v_3, v_2)\}$ in A and the nodes in the corresponding sub-graph in $\sigma \cdot A$, i.e. $\{(v_1, v_2), (v_1, v_4)\}$ are not the same across the two sub-graphs (v_3 was swapped out for v_1).

Automorphisms of A, such as $\sigma' = (4 \quad 1)$ would ensure that the nodes involved in the sub-graph $\{(v_3, v_4), (v_3, v_2)\}$ in A would remain the same. So the automorphism group of A, $\mathrm{Aut}(A)$ can be thought of as the set of relabellings that leave the sub-graphs of A in tact.

$$\sigma' \cdot A = (4 \quad 1) \cdot A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

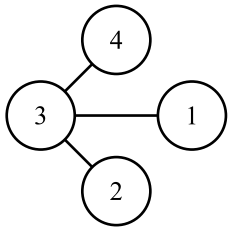


Figure 2: The Automorphism $\sigma' = (4 \quad 1)$ Preserves The Sub-Graph Structure of A

Autobahn works by taking the user-given template graph and classifying functions that are equivariant to the entire input graph while judiciously breaking symmetry with the template graph's automorphism group (see Section 3.1.3). This has advantages and drawbacks, as discussed in Section 3.2.

In contrast, Schur Nets take a spectral approach. In particular, Schur Net layers are described by the Corollary proved in their work. This Corollary gives the core characterization of a non-higher order Schur Net.

In the next section we compare and contrast how Autobahn and Schur Nets how the models compute on a pendant graph with a 5-cycle, in precise detail. This will give us the conceptual basis to give a formal analysis of the expressivity of both frameworks. Moreover, it will suggest that the expressivity of both GNNs frameworks is best measured via sub-graph counting, and as a corollary why WL tests are not a satistifactory measure of expressivity for these frameworks. This will lead us to our major contribution, which is providing a precise characterization of Schur Nets and Autobahn within the measurement framework of Homomorphism Expressivity.

3 Exemplifying The Expressivity Gap Between Schur Nets and Autobahn

With its preliminaries established, the first main contribution of this report is to show in full detail how Autobahn [1] and Schur Nets [2] handle a certain graph.

The graph we will be applying Autobahn and Schur Nets to is a labelled 5 cycle graph with a pendant edge.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

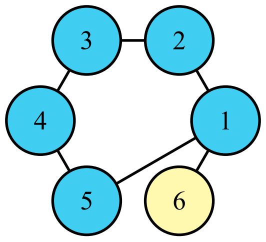


Figure 3: The Graph Given By The Adjacency Matrix In (7)

Remark 3.1 Before we continue, we make a few key observations about our chosen graph (7).

- The graph has a cycle of length 5, $(v_1, v_2, v_3, v_4, v_5, v_1)$
- v_6 is not in the cycle and the edge (v_6, v_1) is why we call this graph a “pendant”.
- We’ve colored (labelled) the nodes v_i that belong to the cycle in blue, the pendant node (v_6) that does not belong to the cycle in yellow.
- Clearly the colors and the pendant node introduce asymmetry.

3.1 Autobahn’s Behavior On The Pendant Graph (7)

Remark 3.1.1 For simplicity, I may gloss over the promotion and narrowing aspects of Autobahn’s algorithm. Although those are the essential contribution, I am primarily interested in treating group theoretic automorphism based networks, as given by [7]

We now observe how Autobahn works on Figure 3.

3.1.1 Step 1: Choose The Template Graph $\mathcal{T}$

Our first step is to choose our template graph $\mathcal{T}$ that aligns with our problem domain. Autobahn is designed to recognized the sub-graphs of the input graph A isomorphic to this sub-graph A and break permutation equivariance across sub-graph boundaries, which gives a more expressive class of neural networks than those whose neurons are permutation equivariant to all of $\mathbb{S}_n$ (i.e MPNNs). Notice that we labelled the nodes with colors corresponding to the selection of our template graph.

Notice the 5-cycle in our node color labelling, so in line with that let us choose the template graph $\mathcal{T} = C_5$.

In particular, the adjacency matrix of $\mathcal{T}$ is given by

$$A_{\mathcal{T}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{8}$$

The automorphism group of $A_{\mathcal{T}}$ is

$$\mathrm{Aut}(A_{\mathcal{T}}) = D_5 \tag{9}$$

The dihedral group of 5 elements, D_5 is the set of rotations and reflections. Visually this makes sense, as rotating or reflecting this sub-graph should not change it.

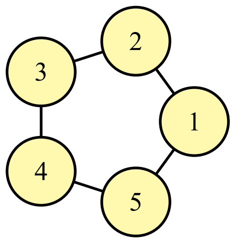


Figure 4: The Template Graph $\mathcal{T}$, A Sub-Graph of A

Now that we have chosen our template graph, Autobahn starts.

3.1.2 Step 2: Determine The Sub-graphs Isomorphic To C_5

- Objective: To find the sub-graph of A that are isomorphic to the template graph $\mathcal{T} = C_5$. In other words, find all sub-graphs in our input graph that contain 5-cycles.
- How: Find all $\sigma \in \mathbb{S}_6$ so that

$$\sigma \cdot A = A_{\mathcal{T}}. \tag{10}$$

3.1.3 Step 3: Construct A Function That Is Equivariant to $\mathrm{Aut}(C_5) = D_5$

Let $f_i^{\ell-1}$ denote the feature vector form the previous layer $(\ell - 1)$ corresponding to the ith vertex. For us, we will consider the first internal layer and let

$$\begin{aligned} f_i^0 &= \deg(v_i) \\ f^0 &= (f_1^0, ..., f_6^0) \\ &= (3, 2, 2, 2, 2, 1) \end{aligned} \tag{11}$$

Let f'^0 be the feature vector but only on the 5-cycle

$$f_i'^0 = (\deg(v_i)) \tag{12}$$

for $i = \{1, ..., 5\}$

We construct our sub-graph neuron so that it is pseudo-equivariant $\mathrm{Aut}(C_5)$ if we only compute the features on the sub-graph, but in actuality is not equivariant to $\mathrm{Aut}(C_5)$. We can construct our sub-graph neuron to be do this by having our sub-graph neuron convolve over all permutations in D_5

$$f'^1 = \sum_{\sigma \in D_5} \sigma \cdot w * f'^0 \tag{13}$$

for some learnable weight w.

Notice that, conceptually, if we only compute f'^0 with respect to the 5-cycle sub-graph, then $f_1^0 = \deg(v_1) = 2$ and (13) is indeed equivariant to D_5. On the other hand, if we compute f'^0 with respect to the entire graph A, then $f_1^0 = \deg(v_1) = 3$, and (13) is not actually equivariant to D_5. This is the crucial insight. In essence, Autobahn breaks permutation equivariance to D_5 of global features of the graph, while maintaining permutation equivariance to D_5 of local features of the graph.

In other words

$$\begin{aligned} f'^0 &= (2,2,2,2,2) \text{ When on considered on the isoloated subgraph} \\ f'^0 &= (3,2,2,2,2) \text{ in actuality. I.e. with respect to the entire graph.} \end{aligned} \tag{14}$$

3.1.4 Step 4: Apply Non-Linearity And Add f_6^1

Finally, we construct the output layer

$$f^1 = f'^1 + f_6^1, \tag{15}$$

where f'^1 is computed with $f'^0 = (\deg(v_1), \deg(v_2), \deg(v_3), \deg(v_4), \deg(v_5)) = (3,2,2,2,2)$

3.2 Summarizing Autobahn

In summary, Autobahn requires you to choose a template graph $\mathcal{T}$ that reflects the input graph's structures and problem domain nicely. For instance, we could have chosen $\mathcal{T} = P_3$, that is, all paths of length 3 in our input graph A.

- Finds the sub-graphs isomorphic to $\mathcal{T}$ on A.
- Constructs neurons that are permutation equivariant to $\mathrm{Aut}(A_{\mathcal{T}})$ with respect the sub-graph
- Incorporates these terms to the neuron's activations on other vertices not in the sub-graph, ensuring global equivariance with respect to the graph.

In summary, by ensuring functions on sub-graphs are permutation equivariant to D_5 **if the features the functions are acting on are computed with respect to the sub-graph**, Autobahn cleverly breaks permutation equivariance to D_5 by applying this function on f'^0 evaluated on the entire input graph.

One apparent drawback of Autobahn is that it requires users to select the template sub-graph. On the other hand, that requirement can actually be considered advantageous as it gives domain practitioners flexibility to choose the template graph that is important to their domain. For instance, chemists applying GNN's on molecules can choose their template graph that is isomorphic to highly functional sub-structures of the molecule [8].

However, the major drawback of Autobahn is that is requires considering all elements of the automorphism group in order to achieve equivariance, via generalized convolution. Moreover, the complexity that arises due to overlapping sub-graphs isomorphic to the input graph introduces complexities and computational overhead. Autobahn addresses these issues by introducing narrowing and promotion, but it is still a source of complexity. This is discussed in Section A.3.

3.3 How Schur Nets Handles The Pendant Graph

The construction of a Schur Net neuron is given by the following corollary proved in their work.

Corollary 3.3.1 *(Schur Net Neuron (Zhang, Xu & Kondor) [2])* Consider a GNN on the input graph $\mathcal{G}$, represented by A. Let $\mathfrak{n}_F$ be a neuron that operates on sub-graph F of $\mathcal{G}$. Let input sub-graph F (and its features) be represented by $T \in \mathbb{R}^m$. Let L be the Laplacian of S, $U_1, \ldots, U_p$ be the eigenspaces of L, and M_i be an orthogonal basis for the ith eigenspace stacked into a $\mathbb{R}^{n\times \dim(U_i)}$ Then for any collection of learnable weight matrices $W_1, ..., W_p \in \mathbb{R}^{a\times b}$

$$\varphi : T \longrightarrow \sum_{i=1}^{p} M_i M_i^T T W_i \tag{16}$$

is a permutation equivariant operation.

Like Autobahn, Schur Net's allows the designer to choose the sub-graph F (see Corollary 3.3.1). Therefore, we choose $F = C_5$, inline with the prior section.

Next, we compute the Laplacian of F. For convenience, recall that the adjacency matrix of the input graph and the 5-cycle template sub-graph stated in (7) is given by

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \text{(Input Graph)}$$

and,

$$A_{\mathcal{T}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \text{(5-Cylcle)}.$$

Then the Laplacian of F is given by $L = D_F - A_F$ so

$$L = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & -1 & 2 \end{bmatrix}. \tag{17}$$

Next, we compute the eigenvalues of L. This is known for 5-cycles [9]. Let λ_i denote the ith eigenvalue. Then for $i = k + 1$, compute $i = 1, ..., 5$

$$\lambda_i = 2 - 2\cos\left(\frac{2\pi k}{5}\right) \tag{18}$$

We also know the eigenvectors are given by the columns of the discrete Fourier matrix. So the kth eigenvector is given by

$$v_j^{(k)} = \frac{1}{\sqrt{n}} e^{\frac{2\pi i k j}{n}},\ j = 0, 1, ..., n-1,\ n = 5. \tag{19}$$

The eigenvalues, their multiplicities, and the eigenvectors are shown in Figure 1.

Figure 1: The Eigenvalues of The C_5 Laplacian L

i	Eigenvalues (λ_i) of $C(5)$	Eigenvectors (v_i) of C_5	Multiplicity
0	0	$\left(\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}}\right)$	1
1	1.381966011250105	$\left(\frac{1}{\sqrt{5}}(1,\cos(2*\frac{\pi}{5}),\cos(4*\frac{\pi}{5}),\cos(4*\frac{\pi}{5}),\cos(2*\frac{\pi}{5}))\right)$	2
2	3.618033988749895	$\left(\frac{1}{\sqrt{5}}(0,-\sin(2*\frac{\pi}{5}),-\sin(4*\frac{\pi}{5}),\sin(4*\frac{\pi}{5}),\sin(2*\frac{\pi}{5}))\right)$	2

Therefore, our eigenspaces, expressed as column vectors (with duplicates), are

$$\begin{aligned}
M_1 &= \left[\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}},\frac{1}{\sqrt{5}}\right]^\top \\
M_2 &= \left[\frac{1}{\sqrt{5}}(1,\cos(2*\frac{\pi}{5}),\cos(4*\frac{\pi}{5}),\cos(4*\frac{\pi}{5}),\cos(2*\frac{\pi}{5}))\right]^\top \\
M_3 &= \left[\frac{1}{\sqrt{5}}(0,-\sin(2*\frac{\pi}{5}),-\sin(4*\frac{\pi}{5}),\sin(4*\frac{\pi}{5}),\sin(2*\frac{\pi}{5}))\right]^\top \\
M_4 &= \left[\frac{1}{\sqrt{5}}(1,\cos(4*\frac{\pi}{5}),\cos(2*\frac{\pi}{5}),\cos(4*\frac{\pi}{5}),\cos(4*\frac{\pi}{5}))\right]^\top \\
M_5 &= \left[\frac{1}{\sqrt{5}}(0,-\sin(\frac{\pi}{5}),\sin(2*\frac{\pi}{5}),-\sin(3*\frac{\pi}{5}),\sin(4*\frac{\pi}{5}))\right]^\top .
\end{aligned} \tag{20}$$

Next we define $T \in \mathbb{R}^{5\times 2}$. Let $T_{i,j} = \begin{cases}\text{The degree of node i} & j{=}1\\ \text{Where node i is next to the pendant} & j{=}2\end{cases}$

$$T = \begin{bmatrix} 3 & 1\\ 2 & 0\\ 2 & 0\\ 2 & 0\\ 2 & 0\end{bmatrix} \tag{21}$$

Finally compute the terms of Corollary 3.3.1. For instance,

$$\begin{aligned}
M_1M_1^\top &= \frac{1}{5}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}\\
M_1M_1^\top T &= \frac{1}{5}\begin{bmatrix}11&1\\11&1\\11&1\\11&1\\11&1\end{bmatrix}
\end{aligned} \tag{22}$$

3.4 Schur Nets Summary

We showed how to construct a Schur Net neuron on the pendant graph. Namely we computed the eigenvalues and eigenspaces of our sub-graph Laplacian via the discrete Fourier basis, which yield orthonormal sub-spaces M_i. The eigenvalues of L had multiplicities, 1, 2, 2, respectively. The corresponding eigenspaces were constructed using the discrete Fourier basis, yielding orthogonal bases M_i. Specifically M_1 corresonded to the eigenvalue of 0 with the eigenvector $\frac{1}{\sqrt{5}}\begin{bmatrix}1&1&1&1&1\end{bmatrix}^\top$, with each M_i being a $5 \times \dim(U_i)$ matrix of our normalized eigenvectors.

The key points to distinguish Schur Nets behavior with that of Autobahn is that Autobahn allowed us to specify how to construct our "equivariant" sub-graph neuron (see (13)), while Schur Nets gave us no such flexibility, instead capturing the asymmetry through spectral analysis. This rigidity in Schur Nets ensures a consistent, mathematically grounded approach, relying on the Laplacian's eigenspaces rather than flexible neuron design, providing a natural characterization of the graph's structure. Whereas Autobahn computes D_5 directly, allowing more flexibility, Schur Nets uses spectral filters to process the node features, where each orthogonal sub-space corresponds to a part of the cycle.

In this example the main limitation in general, as shown by the eigenvalue multiplicity Figure 1, is **not** that the eigenspaces M_i are irreducible, in fact they are for this graph (see section D of the supplement in [2]), but that Schur Net's is **unable** to identify if M_i and M_j are isomorphic. In general however, Schur Nets may not yield a irreducible representation for every graph.

4 A General Characterization Of The Expressivity of Both Autobahn And Schur Nets

4.1 Preliminaries

The main contribution of this project is to provide a precise characterization of the expressivity of automorphism based GNNs given by Autobahn and Natural Graph Networks [7], [10] as opposed to Schur Nets. We are able to do this in general thanks to framework introduced by Zhang et al. [6] and in particular for Schur Nets thanks to the recent results from Zhang and Maron et al [3].

Zhang [6] introduces the concept of *Homomorphism Expressivity*.

Definition 4.1.1 *(Zhang et al [3])* Given two graph F and G, a homomorphism from F to G is a mapping f: $V_F \to V_G$ that preserves edge relations, meaning that, for every $\{u, v\} \in E_F$

$$\{f(u), f(g)\} \in E_G. \tag{23}$$

The set of all homomorphisms form F to G will be denoted as $\text{Hom}(F, G)$

Remark 4.1.2 A question I had, which I believe is valid, is, can the set of homomorphisms from F to G be counted. The answer turns out to be yes in some cases.

Let φ be graph neural network that outputs some graph invariant. Then, the *Homomorphism Expressivity* of φ can be defined.

Definition 4.1.3 *(*Homomorphism Expressivity *Zhang et al [6])* Let φ be a GNN (a function on graph G that outputs some invariant). Let $\mathcal{X}^{\varphi}_G(G)$ be an invariant that φ computes on the graph G. *Homomorphism Expressivity* of φ, denoted by $\mathcal{F}^{\varphi}$, is a family of connected graphs[2] satisfying the following:

- For any two graphs G and H, $\mathcal{X}^{\varphi}_G = \mathcal{X}^{\varphi}_H(H)$ *iff* $\text{Hom}(F, G) = \text{Hom}(F, H)$ for every $F \in \mathcal{F}^{\varphi}$.
- For any graph $F \notin \mathcal{F}^{\varphi}$, there exists a pair of graph G, H so that $\mathcal{X}^{\varphi}_G(G) = \mathcal{X}^{\varphi}_H(H)$ and $\text{Hom}(F, G) \neq \text{Hom}(F, H)$.

Let us compute the homomorphism expressivity for a few simple GNNs.

Example 4.1.4 *(Triangle-Counting GNN)* Let φ be a GNN that outputs the number of triangles in the graph G. For a graph $G = (V_G, E_G)$, $\mathcal{X}^{\varphi}(G) = \|\{\{u, v, w\} \subseteq \mid \{u, v\}, \{v, w\}, \{w, u\} \in E_G\}\|$. This GNN keeps invariance to $\mathbb{S}_n$ by aggregating node features over 3-hop neighborhoods, for example.

Let G be a complete graph, $G = K_4$. This means it has 4 triangles. Let H by a C_4 graph with one extra edge. Say $H = (v_1, v_2, v_3, v_4, v_1)$ and has the edge (v_1, v_3).

Therefore, $\mathcal{X}^{\varphi}$ is the following over the various graphs

[2]A connected graph is a graph $G = (V, E)$ where there is a path v_i to any given node v_j.

- $\mathcal{X}^{\varphi}(G) = 4$ (triangles in K_4)
- $\mathcal{X}^{\varphi}(H) = 1$ (triangle 1-2-3 in C_4)
- $\mathcal{X}^{\varphi}(G) \neq \mathcal{X}^{\varphi}(H)$, so we do not need to check if $\text{Hom}(K_3, G) = \text{Hom}(K_3, H)$, by part 1 of Definition 4.1.3.

Now we consider a new pair G' and H'

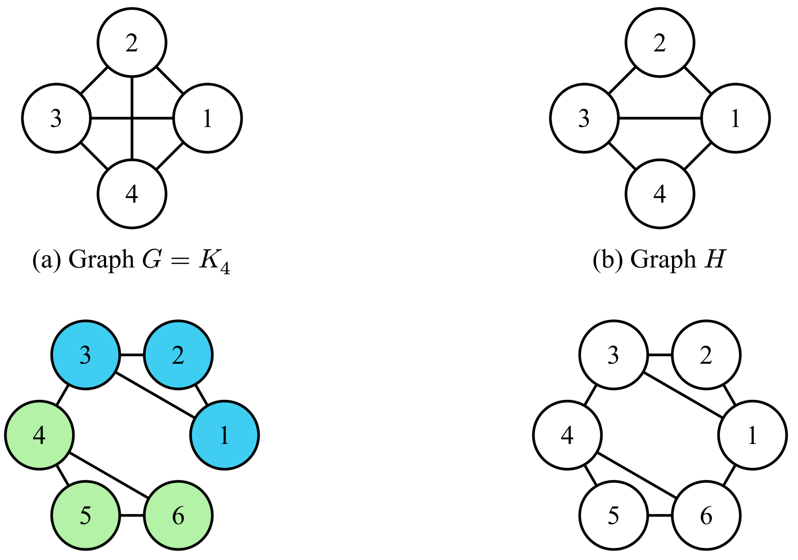
- G' Two disjoint triangles $t_1 = (v_1, v_2, v_3)$ and $t_2 = (v_4, v_5, v_6)$, each triangle forming a K_3.
- H' A cycle C_6 with two triangles. Let us say it has nodes $(v_1, v_2, v_3, v_4, v_5, v_6)$ and edges (v_1, v_3) and (v_4, v_6).
- $\mathcal{X}^{\varphi}(G') = 2$ and $\mathcal{X}^{\varphi} = 2$, so $\mathcal{X}^{\varphi}(G') = \mathcal{X}^{\varphi}(H')$This is a more interesting case because it forces us to consider the homomorphisms from triangle to triangle on G' and H'.

We now consider the homomorphisms $\text{Hom}(K_3, G')$ and $\text{Hom}(K_3, H')$

- A homomorphism from K_3 to a graph maps the triangle to another triangle (see "preserves the edges" in Definition 4.1.1)
- G' has 2 triangles, so $\text{Hom}(K_3, G') = 2$ as each triangle in G' is distinct: $(v_1, v_2, v_3) \neq (v_4, v_5, v_6)$.

So $K_3 \in \mathcal{F}^{\varphi}$ because whenever $\mathcal{X}^{\varphi}(G) = \mathcal{H}^{\varphi}(H)$, the number of homomorphisms from K_3, that is the number of triangles counts, is the same across both graphs.

Conclusion: So triangles $K_3 \in \mathcal{F}^{\varphi}$. In lay terms, our GNN φ can recognize triangles.



(a) Graph $G = K_4$

(b) Graph H

(c) Graph G' With Color-Coded Triangles

(d) Graph H'

Figure 5: The Graphs of G, H, G', and H'

So our GNN φ is expressive for triangles. Let us now see how it does with 3-paths.

Example 4.1.5 *(Homomorphism Expressivity of Triangle GNN for P_3)* We will compute the homomorphism expressivity using our triangle GNN φ, which we defined in the last example.

- Let $F = P_3$, where F is given by Definition 4.1.1.
- So F is a 3-path (v_1, v_2, v_3).
- Let G be a 4-cycle. So G has no triangles, and $\mathcal{X}^{\varphi}(G) = 0$
- Let H be a start graph $\mathbb{S}_4$. No triangles also, so $\mathcal{X}^{\varphi}(H) = 0$
- $\mathcal{X}^{\varphi}(G) = \mathcal{X}^{\varphi}(H) = 0$

Next consider the Homomorphisms $\mathrm{Hom}(P_3, G)$ and $\mathrm{Hom}(P_3, H)$

- $\mathrm{Hom}(P_3, G)$: Map P_3 to a path of length 2 in C_4. C_4 has 4 such paths, and for each path, a P_3 can be mapped in 2 directions: $u \to 1, v \to 2, w \to 3$ or $u \to 3, v \to 2, w \to 1$. So There are $4 \times 2 = 8$ homomorphisms.
- $\mathrm{Hom}(P_3, H)$: In $\mathbb{S}_4$ map v to the center and u, w to two distinct leaves. So we can choose 2 leaves out of 3, and for each choice, we can map u, w in 2 ways ($u \to 1, w \to 2$ or $u \to 2, w \to 1$). Giving us $3 \times 2 = 6$ homomorphisms.
- $\mathrm{Hom}(P_3, G) = 8, \mathrm{Hom}(P_3, H) = 6$, so $\mathrm{Hom}(P_3, G) \neq \mathrm{Hom}(P_3, H)$.

Conclusion: Since $\mathcal{X}^{\varphi}(G) = \mathcal{X}^{\varphi}(H)$ but $\mathrm{Hom}(P_3, G) \neq \mathrm{Hom}(P_3, H)$, this implies that $P_3 \notin \mathcal{F}^{\varphi}$ by Definition 4.1.1.

Before proceeding to the main result, we cite the recent result from Zhang and Maron et al [3], which gives a general way to compute the Homomorphism expressivity of spectral invariant graphs, of which Schur Nets is a part.

Theorem 4.1.6 *(Homomorphism Expressivity of Spectral Invariant GNNs (Zhang & Maron et al) [3])* For any $d \in \mathbb{N}$, the homomorphism expressivity of spectral invariant GNNs with d iterations exists and can be characterized as follows:

$$\mathcal{F}^{\mathrm{Spec},(d)} = \{F \mid F\ has\ parallel\ trees\ of\ depth\ at\ most\ d\} \tag{24}$$

Specifically, the following properties hold:

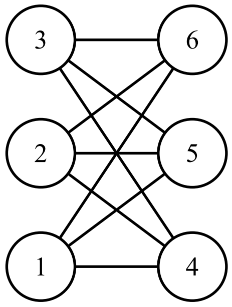
- Given any graphs G and H, $\mathcal{X}_G^{\mathrm{Spec},(d)}(G) = \mathcal{X}_H^{\mathrm{Spec},(d)}(H)$ if and only if, for all connected graphs F with parallel tree depth at most d, $\mathrm{Hom}(F, G) = \mathrm{Hom}(F, H)$.
- $\mathcal{F}^{\mathrm{Spec},(d)}$ is maximal: for any connected graph $F \notin \mathcal{F}^{\mathrm{Spec},(d)}$, there exists graphs G and H so that $\mathcal{X}_G^{\mathrm{Spec},(d)}(G) = \mathcal{H}_H^{\mathrm{Spec},(d)}(H)$ and $\mathrm{Hom}(F, G) \neq \mathrm{Hom}(F, H)$.

4.2 Main Result: Autobahn vs. Schur Nets

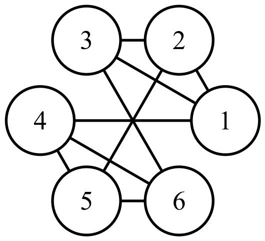
To show how Autobahn and Schur Nets compare under Homomorphism Expressivity. We choose a pair of graphs G and H and a substructure F, and show how Schur Nets is cannot classify such a substructure in $d = 1$ iterations. We show this by computing the homomorphism expressivity given in Definition 4.1.1. Similarly, we show by computing its homomorphism expressivity that Autobahn can recognize our chosen substructure F . We then discuss the implications of this result.

Our graphs will be

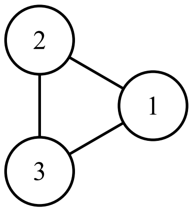
$$G = K_{3,3} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$



$$H = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$



$$F = C_3 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$



4.2.1 One Layer ($d = 1$) Schur Nets

Using Theorem 4.1.6, we compute the parallel tree depth of $F = C_3$. C_3 is just a 3-cycle, it has parallel tree depth of 2 because:

- The first step breaks the cycle[3]
- The second removal turns C_3 into an edge, which is as trivial structure.

Therefore, by Theorem 4.1.6, Schur Nets, which is a spectral invariant GNN, will need at least $d = 2$ iterations to recognize C_3. With only one iteration, Schur nets can only count homomorphisms for graph F with parallel tree depth ≤ 1. Because C_3 has parallel tree depth of 2, a Schur Net with $d = 1$ cannot correctly compute $\text{Hom}(C_3, G)$ or $\text{Hom}(C_3, H)$. **In other words, a one layer ($d = 1$) Schur Nets cannot distinguish graphs G and H using C_3.**

[3]See Definition 3.1 and Definition 3.2 in [3] for more info on how to compute parallel tree depth. In essence, the way it works is by removing one edge one at a time until you hit a trivial structure.

4.2.2 Two Layer ($d = 2$) Schur Nets

So, Schur Nets cannot distinguish G and H based on C_3. What would happen if we let Schur Nets go over the graph twice, $d = 2$? In other words, what if we let Schur Nets neural network have $d = 2$ layers? *The answer is yes.*

Recall the parallel tree depth of C_3 is 2. Theorem 4.1.6 implies that that Schur Nets can handle graphs with parallel tree depth ≤ 2. Therefore, we need to see how Schur Nets counts the homomorphisms $\mathrm{Hom}(C_3, G)$ and $\mathrm{Hom}(C_3, H)$ accurately.

- $\mathrm{Hom}(C_3, G) = \mathrm{Hom}\big(C_3, K_{3,3}\big) = 0$: This is immediately obvious because $K_{3,3}$ has no triangles in it.
- $\mathrm{Hom}(C_3, H) = \mathrm{Hom}(C_3, \text{Triangular Prism}) = 12$: Imagine H as 3 dimensional solid, whose top and bottoms are a triangles, and each vertex of the one triangle has an edge to its corresponding vertex in the other triangle. Each triangle has 3 homomorphisms (isomorphisms in this case) to C_3, as we can walk C_3 by choosing 3 starting vertices and walk in reverse: 2×3. Given that there are two triangles, that gives us 12.

With the homomorphism counts in hand, observe that since $\mathrm{Hom}(C_3, G) = 0 \neq 12 = \mathrm{Hom}(C_3, H)$, Schur Nets can distinguish G and H. **In summary, a 2 layer Schur Nets network can distinguish G and H by counting triangles (i.e. homomorphism to C_3).**

4.2.3 One Layer Autobahn

Autobahn can distinguish G and H via $F = C_3$ (triangles) in one layer. I provide a brief sketch of a proof. An alternative proof, where we interpret Autobahn as a local 2 GNN, is given in the Appendix.

One way to see this for Autobahn in general is simply to set the template graph $\mathcal{T} = C_3$. Then, because by design Autobahn computes sub-graph automorphisms to the template graph $\mathcal{T}$, for each sub-graph Autobahn visits, it can simply output the number "6" to the pooling layer. Then the pooling layer, which gathers each sub-graph activation together, will sum these $6'$s and output the result, the result will be the $\mathrm{Hom}(\mathcal{G}, C_3)$ (where $\mathcal{G}$ is our input graph).

4.2.4 Implications

At a depth of 1, we saw that Schur Nets lacked the expressivity to detect triangles on $G = K_{3,3}$, and H. Autobahn, in contrast, can distinguish detect G and H via triangles C_3.

However, once we increased Schur Nets to two layers ($d = 2$), Schur Nets can distinguish G and H using C_3. The takeaway is that Autobahn is highly expressive because it allows users to select template graph $\mathcal{T}$ that are salient to the problem domain. But Autobahn has complexity and requires defining $\mathcal{T}$. Schur Nets on the other, sacrifices some of the dedicated expressiveness of Autobahn but is a simpler architecture that generalizes to different groups of graphs.

In essence, given a classification task, if you want your GNN value a sub-graph, whose parallel tree length is quite large, you should probably *avoid* using Schur Nets. In this case, by Theorem 4.1.6 Schur Nets will only be able to recognize your sub-graph after many iterations.

In this case, you want to use Autobahn and construct make its template graph $\mathcal{T}$ your sub-group of interest.

On the other hand, if you do not have a particular sub-structure of interest, or your problem domain has an infeasible number of sub-graphs that are relevant, it may be best to avoid using Autobahn in favor for Schur Nets. **This is especially true, if those many sub-graphs have small parallel tree depths**.

5 Conclusion

6 Acknowledgements

This work is incomplete. I will send the completed version this weekend, but I acknowledged that this submission will be the one that is graded.

References

[1] E. Thiede, W. Zhou, and R. Kondor, "Autobahn: Automorphism-based Graph Neural Nets," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 29922–29934. Accessed: Mar. 07, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/faf02b2358de8933f480a146f4d2d98e-Abstract.html

[2] Q. Zhang, R. Xu, and R. Kondor, "Schur Nets: Exploiting Local Structure for Equivariance in Higher Order Graph Neural Networks," *Advances in Neural Information Processing Systems*, vol. 37, pp. 5528–5551, Jan. 2025, Accessed: Feb. 20, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/hash/0a0e2c6a487314f821346bdc04869e36-Abstract-Conference.html

[3] J. Gai, Y. Du, B. Zhang, H. Maron, and L. Wang, "Homomorphism Expressivity of Spectral Invariant Graph Neural Networks," presented at the The Thirteenth International Conference on Learning Representations, Oct. 2024. Accessed: Feb. 20, 2025. [Online]. Available: https://openreview.net/forum?id=rdv6yeMFpn

[4] W. Fulton and J. Harris, *Representation Theory*, vol. 129. in Graduate Texts in Mathematics, vol. 129. New York, NY: Springer New York, 2004. doi: 10.1007/978-1-4612-0979-9.

[5] N. T. Huang and S. Villar, "A Short Tutorial on the Weisfeiler-Lehman Test and Its Variants," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, pp. 8533–8537. Accessed: Mar. 09, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9413523/?casa_token=_n1rleI2ABMAAAAA:v0F1PK1qx4ZasLCO3a20reQzAboM6HcXkp7S-80tsHLikMFcpJVjHRclbOkCJ21GihswoYhmoA

[6] B. Zhang, G. Feng, Y. Du, D. He, and L. Wang, "A Complete Expressiveness Hierarchy for Subgraph Gnns via Subgraph Weisfeiler-Lehman Tests," in *International Conference on Machine Learning*, PMLR, 2023, pp. 41019–41077. Accessed: Mar. 09, 2025. [Online]. Available: https://proceedings.mlr.press/v202/zhang23k.html

[7] P. de Haan, T. S. Cohen, and M. Welling, "Natural Graph Networks," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, pp. 3636–3646. Accessed: Mar. 09, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/2517756c5a9be6ac007fe9bb7fb92611-Abstract.html

[8] X. Kong, W. Huang, Z. Tan, and Y. Liu, "Molecule Generation by Principal Subgraph Mining and Assembling." Accessed: Mar. 10, 2025. [Online]. Available: http://arxiv.org/abs/2106.15098

[9] L. Trevisan, "Lecture06." Accessed: Mar. 10, 2025. [Online]. Available: https://theory.stanford.edu/~trevisan/cs359g/lecture06.pdf

[10] E. H. Thiede, T. S. Hy, and R. Kondor, "The General Theory of Permutation Equivarant Neural Networks and Higher Order Graph Variational Encoders." Accessed: Mar. 05, 2025. [Online]. Available: http://arxiv.org/abs/2004.03990

[11] B. Y. Weisfeiler and A. A. Leman, "THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN."

[12] B. Zhang, J. Gai, Y. Du, Q. Ye, D. He, and L. Wang, "Beyond Weisfeiler-Lehman: A Quantitative Framework for GNN Expressiveness." Accessed: Mar. 01, 2025. [Online]. Available: http://arxiv.org/abs/2401.08514

[13] D. Eppstein, "Parallel Recognition of Series-Parallel Graphs," *Information and Computation*, vol. 98, no. 1, pp. 41–55, May 1992, doi: 10.1016/0890-5401(92)90041-D.

A Appendix

A.1 Discussion: Autobahn Is a k-Local GNN

Zhang et al [6] provides a definition for GNNs that are based on graph coloring, which is a reformulation of the coloring perspective introduced in [11]. Graph GNNs then fall into one of the sub-categories given by this coloring definition. Their definition is below, and in particular, one could say that Autbahn falls into the k-Local GNN GNN category. In this subsection, we discuss

Autobahn and the general results of Homorphism expressivity for k-Local GNNs. As of writing, a general characterization for k local GNNs has only been done for $k = 2$.

Definition A.1.1 *(Zhang et al [6])* GNNs can be generally described as graph functions that are invariant under isomorphism. This is precisely the characterization we gave at the beginning of the report, where in Section 1 we said the key requirement of graph neural networks is to remain invariant under permutations of the symmetric group, and of course for any $\sigma \in \mathbb{S}_n$, $\mathcal{G} \cong \sigma \cdot \mathcal{G}$.

Most GNN's therefore, can be said to adhere to a *color refinement* approach: they maintain a feature representation, i.e. a color, for each vertex or vertex tuple, and iteratively refine these features by processing them through the equivariant hidden layers, until the last layer pools all of these features together to ensure invariance (isomorphism).

- **MPNN**: Given a graph G, MPNN maintains a color $\mathcal{X}_G^{\mathrm{MP}}(u)$ for each vertex $u \in V_G$. Initially the color only depends on the label of the vertex. For instance $\mathcal{X}_G^{\mathrm{MP},(0)}(u) = \ell_G(u)$. Then, for each layer, the color is refined by the update

$$\mathcal{X}_G^{\mathrm{MP},(t+1)}(u) = \operatorname{hash}\Big(\mathcal{X}_G^{\mathrm{MP},(t)}, \Big\{\Big\{\mathcal{X}_G^{\mathrm{MP},(t)}(v) \mid v \in N_G(u)\Big\}\Big\}\Big). \tag{25}$$

  Here, we used $\{\{\cdot\}\}$ to denote the multi-set. After a certain point, the color processing stabilizes and the graph representation becomes $\mathcal{X}_G^{\mathrm{MP}}(G) = \{\{\mathcal{X}_G^{\mathrm{MP}}(u) \mid u \in V_G\}\}$

- **Local GNN**. We quote their definition

  "Inspired by the k-WL test (Grohe, 2017), Local k-GNN is defined by replacing all global aggregations in k-WL by sparse ones that only aggregate local neighbors, yielding a much more efficient CR algorithm." [6]

I omit the other sub-graph types.

Automorphism based GNN [7] such as Autobahn are considered by homomorphism expressivity to be *local GNN's*, according to Definition A.1.1. Their forumulation for local GNN's is not as precise as for the other 3 GNN classes they characterize, but in general, Autobahn works by being a sparse k-WL. The number of vertices in the given template graph corresponds to k. Moreover, Autobahn avoids the computation overhead of k-WL by identifying sub-sequences of vertices that are isomorphic to the template sub-graph.

Without a more rigorous formulation of the Local GNN variant we cannot prove the Autobahn is a local GNN, which precludes us from using the results for local 2-GNN's in [3]. Moreover, Autobahn is not in a general a local 2-GNN, so Theorem 3.4 [12] cannot be used. However, while we have not classified Autobahn in general under homomorphism expressivity, we classify Autobahn under certain sub-graphs and compare.

Researching a proper definition of Local k-GNN and formulating Autobahn in terms of it is an area for future investigation. However, the only proven general characterization of Local k-GNN is for $k = 2$ [12], and the proof is a major technical contribution. Therefore, it may not be best to consider this route, especially as homomorphism expressivity provides valuable insight alone when using it on particular graphs, as we show now.

Definition A.1.2 *(Narrowing In Autobahn [1])* Let $(i_1, ..., i_k)$ be an ordered subset of $\{1, 2, ...m\}$. The set $\{1, ..., m\}\}$ corresponds to the sub-graph isomorphic to the template graph $\mathcal{T}$, and the ordered subset $(i_1, ..., i_k)$ corresponds to k "ambassador nodes" that overlap with a different sub-graph.

Now, for all $u \in \mathbb{S}_k$, let $\tilde{u} \in \mathbb{S}_k$ be the permutation that u to the first k elements and for all $s \in \mathbb{S}_{m-k}$, let $\tilde{s} \in \mathbb{S}_m$ be the permutation that applies s to the last $m - k$ elements. Then, given our activation function $f : \mathbb{S}_m \to \mathbb{R}^d$, the *narrowing* of f to $(i_1, ..., i_k)$ is

$$f{\downarrow_{i_1,...i_k}} = (m-k)!^{-1} \sum_{s \in \mathbb{S}_{m-k}} f(\tilde{u}\tilde{s}t) \tag{26}$$

Remark A.1.3 This was taken from [1]. At a high-level narrowing produces a function $f\downarrow$ that is only dependent on the important nodes $\big(v_{i_1}, ..., v_{i_k}\big)$.

A.2 Autobahn As a Local 2-GNN Distinguishes

Autobahn is a very flexible framework. Assuming we can indeed think of Autobahn as a k-local GNN, we show that, even as a 2-local GNN, it it is distinguishes the graphs given in Section 4.

Proof A.2.4 *(Trivial)* Using [12], all 2-local GNNs can recognize any two graphs G and H if via F, if F is a *strongly nested ear decomposition* [12], [13]. C_3 is indeed a strongly nested

A.3 How Narrowing And Promotion Work In Autobahn: A Qualitative Explanation

Remark A.3.5 We omitted details of Autobahn's primary contribution, that is, the formalized notions of *narrowing* and *promotion*. This is because we are concerned with Autobahn's sub-graph counting abilities, and narrowing and promotion are techniques to reduce the computation complexity of achieving automorphism based neurons when graphs overlap [7].

We constructed our example to be particularly simple, and showed such an automorphism based neuron [7]. The point of the section was to illustrate, through elementary computation and novice language, the sub-graph counting abilities of automorphism based neuron networks.

See the appendix Section A.3 for an explanation of how Autobahn uses narrowing and promotion to reduce the computation overhead, which is important when the input graph contains more than 1 sub-graph isomorphic to the template graph.

At any given layer i, Autobahn first identifies all the sub-graphs of A that are isomorphic to $\mathcal{A}_{\mathcal{J}}$. In our case, $(v_1, v_2, v_3, v_4, v_5)$ is the only such sub graph of A.

Next, given we are looking at the first layer, we need to define an input domain. Let the input to the first layer be the degree of each node on the feature graph.

Next, for each sub-graph isomorphic to A, Autobahn performs *narrowing*. *Narrowing* will be described plainly here. The formal definition is given in the appendix. *Narrowing* takes the sub-graph $\left(v_{i_1}, ..., v_{i_m}\right)$ and partitions the nodes of the graph into two groups: the important group $\mathcal{J}_1 = \{i_1, ..., i_k\}$ and the not important group consisting of the remaining $k - m$ nodes, $\mathcal{J}_2 = \{i_{k+1}, ..., i_m\}$. Note that I assumed that the first 1 through k indices were the important nodes, but typically narrowing adds the indices to the important group which *overlap* with a another sub-graph. Let us call such nodes the "ambassador" nodes.

In our example, narrowing operates on our sub-graph $\{v_1, ..., v_5\}$. It then adds node v_1 to the important group $\mathcal{J}_1 = \{i_1\}$ and puts the remaining $2, ..., 4$ indices in the not important group $\mathcal{J}_2 = \{i_2, ...i_4\}$. Finally, the narrowing procedure is applied to our sub-graph neuron, which makes our sub-graph neuron only depend on the important nodes in $\mathcal{J}_1$. See the appendix Definition A.1.2 for the formal definition of narrowing.

Similarly to how narrowing is done, then *promoting* is done to the neuron, which takes the narrowing activations and spreads them out to all the m nodes in the sub-graph that this neuron is working on.

We won't go into narrowing or promotion in too much detail, but we can provide a sketch of how to ensure that our neuron $\mathcal{n}$ is equivariant to $\mathrm{Aut}(A_{\mathcal{J}})$ and also equivariant to all of A in general.

For example, in our case let the input feature vectors be $f_1 = \deg(v_1) = 2, f_2 = \deg(v_2) = 2 = ... = f_5$. Notice that $f_1 = 2$, because we are only considering degrees within the sub graph $v_1, ...v_5$.

Now, we can ensure equivariance of our function $\mathcal{n}$ on this 5-tuple with respect to D_5, by convolving over the element of D_5.

$$D_5 = \sum_{\sigma \in D_5} f \tag{27}$$

Now, in order to account for v_1's connection to v_6, we redefine the feature vectors f_i to include a indicator variable saying if the node is adjacent to the pendant node v_6.

$$f_1 = (\deg(v_1), 1) = (2, 1) \qquad f_2 = (\deg(v_1)), 0) = (2, 0) = f_2 ... = f_5 \tag{28}$$