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.

1 Introduction

Placing the task of learning under a mathematical formalism has been the aspiration of different groups of researchers for various reasons: philosophers, logicians, information theorists, researchers studying AI alignment, and others.

Historically, mathematicians used objects from statistics to construct a formal framework for learning. Simultaneously, a handful of mathematicians chose to couch learning in the language of group theory, using representation theory of finite groups [4] to realize models on Turing machines [5], [6].

On one hand, the theory of finite groups is of interest to researchers building models on graphs because a graph, which is a set of vertices and a collection of binary relations, is much like a group. On the other hand, for those studying group actions, a group action on a graph can be readily interpreted, motivating those neural network theorists to first try to apply their formalism on various types of graph learning problems.

In an attempt to create a group based formalism for learning, graphs are an attractive object of which to first study because the object suggests that actions on them respect certain constraints so loudly and so naturally, that researchers consider the constraints to be axiomatic. In particular, given a representation $A \in \{0,1\}^{n\times n}$ of a graph $\mathcal{G}$, permuting the labels of each node the same graph, and so functions we wish to construct should be insensitive to this augmentation. This can be formalized by saying that given our graph $\mathcal{G}$, we want $f(\mathcal{G}) = f(\sigma \cdot \mathcal{G})$ for any permutation $\sigma \in \mathbb{S}_n$.

To achieve this invariance, designers

In the most abstract sense, given a graph $\mathcal{G}$, researchers wish to construct a function φ so that $\varphi(\mathcal{G})$ yields something useful. We want to define φ as a composition of a finite number of layers ℓ

$$\varphi = \varphi^{\ell} \circ \varphi^{\ell-1} ... \varphi^{2} \circ \varphi^{1}. \tag{1}$$

To motivate the group theoretic formalism we expound in this paper, let's consider problem of designing a function that takes a graph $\mathcal{G}$ in the form of a star, and consider how we want to design such a function.

Example 1.1 We define a *star graph* to be a graph $\mathcal{G}$ with n vertices, where the first $n-1$ have an edge to the nth vertex, and no other edges exist in the graph.We wish to construct a function φ that indicates whether or not a graph is a star.

Two star graphs are shown in Figure 1.

(a) The Original Star Graph (b) The Star Graph under $\sigma = (06)$

Figure 1: A star graph under different labellings

Say our function φ was first given the graph in Figure 1a and correctly classified it. Now we relabel the nodes in our graph by swapping the label on node 0 with the label on node 6. The relabel graph is shown in Figure 1b.

We want our function to still classy the graph as a star graph. In fact, no matter how we label the graph's nodes, we still want our function to classify the graph as a star graph.

2 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 4.1, 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. 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 [7] 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 subgraph structures, has been recognized recently by other researchers as well, and has been formalized by [8] in a new framework for expressivity measurement, termed *Homomorphism Expressivity*. This initial work established Homomorphism Expressivity for subgraph 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 construction 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.

3 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 3.1 *(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

$$[\sigma \cdot \mathbf{T}]_{i_1,\dots i_k} = [\mathbf{T}]_{\sigma^{-1}(i_1),\dots,\sigma^{-1}(i_k)} \tag{2}$$

[1] Facetious

Remark 3.2 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 3.1. Conceptually, permuting the graph A by Definition 3.1 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 3.3 *(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 $A^{n^{k_1}}$,

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

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

Remark 3.4 Note, I am not sure exactly how you define permutation equivariance if f, represented by a matrix, was not square . For instance if

$$f = L^{m\times n} \tag{4}$$

and $m \neq n$, then $\sigma \cdot L$ cannot be defined using Definition 3.1. $\cdot$.

Definition 3.5 *(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 $A^{n^{k_1}}$,

$$f(\sigma \cdot A) = \sigma \cdot f(A) \tag{5}$$

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

Corollary 3.6 *(An Equivalent Definition of Permutation Equivariance)* Note that the permutation action of $\mathbb{S}_n$ given by Definition 3.1, 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

$$\rho(\sigma) \star f = f \star \rho(\sigma) \tag{6}$$

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.

Definition 3.7 *(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 3.1 leave $\mathcal{G}$ unchanged. That is the set

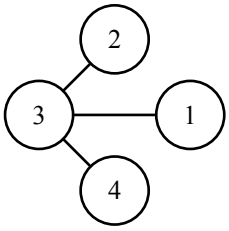
$$\{\sigma \in \mathbb{S}_n \mid \sigma \cdot A = A\}. \tag{7}$$

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

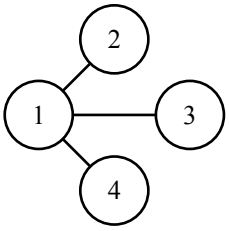
Example 3.8 *(4 node graphs)* It was not initially clear to me why one would be interested in the Automorphism group of a graph at all. Furthermore, For any given graph $A \in \{0,1\}^{n\times n}$, that $\mathbb{S}_n$ was not always essentially e $\mathrm{Aut}(A)$. Clearly though $\mathrm{Aut}(A) \neq \mathbb{S}_n$. 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 $\sigma \notin \mathbb{S}_n$.

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{vs.} \quad \sigma \cdot A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$



(a) The Graph A Given By (8)



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

Figure 2: $A \neq \sigma A$ 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.

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

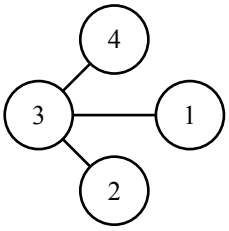


Figure 3: The Automorphism $\sigma' = (4 \quad 1)$ Preserves The Sub-Graph Structure of A

Definition 3.9 *(Homomorphism Expressivity (Q. Zhang et al)[8])*

Corollary 3.10 *(Schur Net Neuron (from Q. Zhang, Xu & Kondor) [2])*

4 How Autobahn And Schur Nets Differ: An Example

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.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

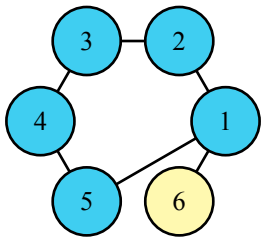


Figure 4: The Graph Given By The Adjacency Matrix In (9)

Remark 4.1 Before we continue, we make a few key observations about our chosen graph (9).

- 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.

4.1 Autobahn's Behavior On (9)

Remark 4.1.1 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 [9]

We now observe how Autobahn works on Figure 4. Our first step is to choose our template graph $\mathcal{T}$ 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

$$A_{\mathcal{T}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{10}$$

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

$$\mathrm{Aut}(A_{\mathcal{T}}) = D_5 \tag{11}$$

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.

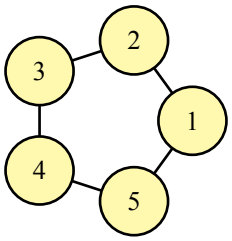


Figure 5: The Template Graph $\mathcal{T}$, A Sub-Graph of A

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

4.1.1 Step 3 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

$$\sigma \cdot A = A_{\mathcal{T}}. \tag{12}$$

4.1.2 Step 4 Construct A Function That Is Equivariant to $\mathbf{Aut}(\boldsymbol{C_5}) = \boldsymbol{D_5}$

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

$$\begin{aligned} f_i^0 &= \deg(v_i) \\ f^0 &= (f_1^0, ..., f_6^0) \\ &= (3,2,2,2,2,1) \end{aligned} \tag{13}$$

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

$$f_i'^0 = (\deg(v_i)) \tag{14}$$

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

We consruct 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

$$f'^1 = \sum_{\sigma \in D_5} \sigma \cdot w * f'^0 \tag{15}$$

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 (15) 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 (15) 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

$$\begin{aligned} &f'^0 = (2,2,2,2,2) \text{ When on considered on the isoloated subgraph} \\ &f'^0 = (3,2,2,2,2) \text{ in actuality. I.e. with respect to the entire graph.} \end{aligned} \tag{16}$$

4.2 Apply Non-Linearity And Add $\boldsymbol{f_6^1}$

Finally, we construct the output layer

$$f'^1 = f'^1 + f_6^1, \tag{17}$$

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

4.3 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 example, 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.

Clearly the major drawback of Autobahn is that it requires users

5 Exemplifying The Expressivity Gap Between Schur Nets and Autobahn

In this first part of the main section, we postpone our use of Homomorphism Expressivity, and analyze the different between Autobahn [1] and Schur Nets [2] by going through particular examples.

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.

6.1 TODOs

- Transcribe C4 example of 5 node (pendant node makes it clear, but I like comparing it to $\mathbb{S}_4$ better) from scratch notes but already 1/3 way.
- Transcribe C3 example (my first attempt earlier this week, include it ?)
- Now, gather parallel trees analysis notes for $d = 2$ and 3. (Too many examples)?
- Now, show exactly how Schur Nets is good at capturing parallel trees up to d, show autobahn captures symmetries more generally. **Schur Nets may miss symmetries in parallel trees after a certain depth.**

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] J. Wood and J. Shawe-Taylor, "Representation Theory and Invariant Neural Networks," *Discrete applied mathematics*, vol. 69, no. 1–2, pp. 33–60, 1996, Accessed: Mar. 07, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0166218X95000753

[6] I. R. Kondor, *Group Theoretical Methods in Machine Learning*. Columbia University, 2008. Accessed: Mar. 07, 2025. [Online]. Available: https://search.proquest.com/openview/d8ae7d9e47c87bb0acb05bf06ae8b6aa/1?pq-origsite=gscholar&cbl=18750

[7] 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

[8] 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

[9] 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

A Appendix

Definition A.1 *(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

$$f \underset{i_1,...i_k}{\downarrow} = (m-k)!^{-1} \sum_{s \in \mathbb{S}_{m-k}} f(\tilde{u}\tilde{s}t) \tag{18}$$

Remark A.2 This was taken from [1]. At a high-level narrowing produces a function $f \downarrow$ that is only dependent on the important nodes $\left(v_{i_1}, ..., v_{i_k}\right)$.

A.1 How Narrowing And Promotion Work

Remark A.1.3 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 [9].

We constructed our example to be particularly simple, and showed such an automorphism based neuron [9]. 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.1 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 $\mathcal{A}_{\mathcal{T}}$. 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 $\mathcal{I}_1 = \{i_1, ..., i_k\}$ and the not important group consisting of the remaining $k - m$ nodes, $\mathcal{I}_2 = \{i_{k+1}, ..., i_m\}$. 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 $\mathcal{I}_1 = \{i_1\}$ and puts the remaining $2, ..., 4$ indices in the not important group $\mathcal{I}_2 = \{i_2, ...i_4\}$. Finally, the narrowing procedure is applied to our sub-graph neuron, which makes our sub-graph neuron only depend on the important nodes in $\mathcal{I}_1$. See the appendix Definition A.1 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 $\mathrm{Aut}(A_{\mathcal{T}})$ 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 $\mathscr{n}$ on this 5-tuple with respect to D_5, by convolving over the element of D_5.

$$D_5 = \sum \tag{19}$$

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.

$$f_1 = (\deg(v_1), 1) = (2, 1) \qquad f_2 = (\deg(v_1)), 0) = (2, 0) = f_2 ... = f_5 \tag{20}$$