Galois Theory Challenges Weisfeiler Leman: Invariant Features for Symmetric Matrices and Point Clouds

Beatrix Wen[*]
Applied Math and Statistics Department
Johns Hopkins University
Baltimore, MD 21218
`ywen15@jhu.edu`

Caio F. Deberaldini Netto[*]
Applied Math and Statistics Department
Johns Hopkins University
Baltimore, MD 21218
`cnetto1@jhu.edu`

Thabo Samakhoana[*]
Applied Math and Statistics Department
Johns Hopkins University
Baltimore, MD 21218
`tsamakh1@jhu.edu`

Soledad Villar
Applied Math and Statistics Department
Johns Hopkins University
Baltimore, MD 21218
`svillar3@jhu.edu`

Ningyuan (Teresa) Huang
Center for Computational Mathematics
Flatiron Institute
New York, NY 10010
`thuang@flatironinstitute.org`

Abstract

Recent work [Blum-Smith et al., 2024] proposed an invariant machine learning model on point clouds and symmetric matrices based on invariant features. For symmetric matrices, the model is invariant under the group action of permutation by conjugation. Therefore, this model can be used to test whether two graphs are isomorphic (by checking whether the invariant features of two graphs coincide). In this work, we investigate the expressive power of this new method. Our theoretical results show that on undirected graphs, the method is strictly worse than graph neural networks (GNNs) based on message passing. To improve its expressivity, we propose a modified method by adding a new invariant function, which we test empirically against GNNs and other baseline methods. The newly proposed function set outperformed the original invariant features, yielding compatible or exceeding results compared to GNNs.

1 Introduction

Recently, Blum-Smith et al. [2024] proposed a machine learning model that when evaluated on symmetric matrices it is invariant with respect to permutations of rows and columns. By representing graphs with symmetric matrices, this model provides a new way to distinguish nearly all non-isomorphic graphs except for a measure-zero bad set S_1. Graph neural networks (GNNs) based on message-passing operations are also capable of distinguishing nearly all non-isomorphic graphs

[*]These authors contributed equally to this work.

except another measure-zero bad set S_2 [Morris et al., 2023, Joshi et al., 2024]. In this paper, we compare the expressive power of GNNs with that of the new model proposed by Blum-Smith et al. [2024]. Our comparison is done both theoretically and empirically. Theoretically, we find that the new model is no more powerful than GNNs on undirected graphs. Empirically, we demonstrate that the model achieves competitive performance in certain domains, although it performs worse than GNNs in others, despite offering better computational efficiency. To improve the expressive power of the aforementioned method, we propose a modified model that has more expressive power while maintmaintaining same computational complexity as the original one.

This paper is organized as follows: Section 2 presents the basic concepts and definitions, and the problem we address. Section 3 details the proposed method using the theory described in the previous section, whilst in Section 4 the new method is empirically compared with well-known baselines in two different domains, following an analysis of the results and how they relate to the proposed theory. Section 5 closes the paper with a summary of the discussions done throughout the last two sections and pinpoints the future steps of this work.

2 Preliminaries

Let $G = (V, E)$ denote an undirected graph with $V = V(G)$ denoting the set of nodes and $E = E(G) \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ the set of edges[2]. As shorthand, we write (u, v) to represent the edge $\{u, v\}$. The neighborhood of a vertex v is $\mathcal{N}(v) = \{u \in V \mid (u, v) \in E\}$. All graphs we consider have a finite vertex set V with the same number of nodes, i.e. $|V| = n$ for some positive $n \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of nonnegative integers. A weighted graph is a triple $G = (V, E, X)$, where $X(G) = X : V \times V \to \mathbb{N}$ is symmetric, i.e. $X(u, v) = X(v, u)$ for all $u, v \in V$. We will assume all graphs are weighted; if a graph is unweighted, we make it weighted by taking X to be its adjacency matrix, i.e.

$$X(u,v) = \begin{cases} 1 & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } (u,v) \in V \times V.$$

Matrix representation and permutation invariance. Since our graphs have the same number of nodes, we will assume without loss of generality that $V = [n] := \{1, \ldots, n\}$. Thus the weights X of a graph can be identified with a symmetric $n \times n$ matrix, which we also denote by X, whose (i, j)-th entry is $X(i, j)$. Let $\mathcal{G}$ denote the set of weighted graphs and $\mathfrak{S}_n$ denote the group of permutations of $[n]$. Two graphs $G_1, G_2 \in \mathcal{G}$ are isomorphic if there exists $\sigma \in \mathfrak{S}_n$ such that $(\sigma(i), \sigma(j)) \in E(G_2)$ if and only if $(i, j) \in E(G_1)$ and $X(i, j) = X(\sigma(i), \sigma(j))$ for all $i, j \in V(G_1)$. Therefore, if G_1 and G_2 are isomorphic, then the matrices $X(G_1)$ and $X(G_2)$ satisfy

$$X(G_2) = P_\sigma^\top X(G_1) P_\sigma,$$

for the permutation matrix P_σ defined by $P_\sigma x = (x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ for all $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, where σ is the isomorphism between G_1 and G_2. A function $f : \mathcal{G} \to \mathcal{Y}$ is invariant (with respect to $\mathfrak{S}_n$) if $f(G_1) = f(G_2)$ whenever $G_1 = G_2$ are isomorphic. Let $\mathcal{S}(n)$ denote the set of symmetric $n \times n$ matrices. The orbit of $X \in \mathcal{S}(n)$ (under the action of $\mathfrak{S}_n$) is the set $\{P_\sigma^\top X P_\sigma \mid \sigma \in \mathfrak{S}_n\}$. A function $f : \mathcal{S}(n) \to \mathcal{Y}$ is invariant (with respect to $\mathfrak{S}_n$) if it is constant along orbits, i.e. if $f(P_\sigma^\top X P_\sigma) = f(X)$ for all $X \in \mathcal{S}(n)$ and $\sigma \in \mathfrak{S}_n$. Note that an invariant $f : \mathcal{S}(n) \to \mathcal{Y}$ induces an invariant function on $\mathcal{G}$ via $f(G) = f(X(G))$.

Invariant features and Galois theory (IFG) Blum-Smith et al. [2024] proposed a machine learning model on symmetric matrices that is invariant with respect to $\mathfrak{S}_n$. Briefly, their model relies on a set of functions that are invariant with respect to the bigger group $\Gamma = \mathfrak{S}_n \times \mathfrak{S}_{n(n-1)/2}$ that permutes diagonal elements to diagonal elements, and off-diagonal elements to off-diagonal elements independently. The sets of invariant functions for Γ are easy to compute. For example, one could use $\{f^d, f^o\}$ defined under section 3.1. To ensure invariance with respect to the smaller group $\mathfrak{S}_n$, they add an extra feature that is invariant with respect to $\mathfrak{S}_n$ but not Γ. A "Galois Theorem" guarantees that the new set can be used to generate the field of invariant functions with respect to $\mathfrak{S}_n$. We therefore call this model Invariant features and Galois theory (IFG). IFG can separate $\mathfrak{S}_n$ orbits in

[2]Our notation is similar to Morris et al. [2023] and we adopt their convention of excluding self-loops from the set of edges.

$\mathcal{S}(n)$ except on a set of measure zero. This can be seen as a consequence of the Rosenlicht theorem [Rosenlicht, 1956]. More precisely, there exists $B \subset \mathcal{S}(n)$ (the bad set) with Lebesgue measure zero such that $\mathrm{IFG}(X_1) \neq \mathrm{IFG}(X_2)$ for all $X_1, X_2 \in \mathcal{S}(n) \setminus B$ with different orbits. In addition, such a set is $\mathfrak{S}_n$ invariant, i.e. $P_\sigma^\top X P_\sigma \in B$ for all $X \in B$ and $\sigma \in \mathfrak{S}_n$.

Message Passing Graph Neural Networks (MPGNNs). GNNs are machine learning models designed to work with graph data. They create a vectorial representation of each node that incorporates information from the node's neighborhood. More specifically, let $f^{(0)} : V \to \mathbb{R}^{d_0}$ be an initial node representation. The $k+1$ representation (or layer) takes the form (see Xu et al. [2019])

$$f^{(k+1)}(v) = \text{Combine}^{(k)}\left(f^{(k)}(v), \text{Aggregate}^{(k)}\left(\{\!\{ f^{(k)}(u) \mid u \in \mathcal{N}(v) \}\!\}\right)\right),$$

where $\{\!\{\cdot\}\!\}$ denotes a multiset. Thus the k-th layer of GNNs is refinement of the initial representation that incorporates information from k-hop neighbors. For weighted graphs $G = (V, E, X)$, we assume $f^{(0)}$ is consistent with X, i.e. $f^{(0)}(u) = f^{(0)}(v)$ if $X(u,u) = X(v,v)$. The message passing (via the Aggregate functions) in GNNs ensures that they are invariant with respect to $\mathfrak{S}_n$.

The Weisfeiler Leman Test. The Weisfeiler Leman (WL) test of Weisfeiler and Leman [1968] is a method of testing whether two graphs are isomorphic. The WL test begins with an initial labeling of the graph's nodes and iteratively updates these labels by hashing the multiset of a node's label and its neighbors' labels into a new label. The relabeling stops if two graphs have different labels or if the hashing stabilizes – see Huang and Villar [2021] for a dedicated tutorial. This test can distinguish all but a small subset of graphs Babai et al. [1980]. The WL test can be generalized into a k-WL hierarchy, which labels k-node subgraphs of G instead of the nodes. For completeness, the k-WL test is presented in Algorithm 1, where $\vec{v}$ denotes the subgraph formed by the vertices $\vec{v} \in V^k$, h denotes a hash function, $\mathcal{N}(\vec{v}, i) := \left\{ (v_1, \ldots, v_{i-1}, w, v_{i+1}, \ldots, v_k) : w \in V \right\}$, and $c^l_{\vec{v}}$ denotes the l-th coloring of $\vec{v}$.

Algorithm 1 k-WL ($k \geq 2$)

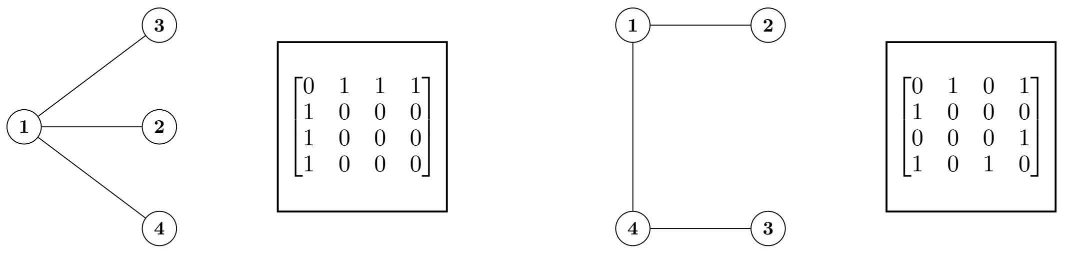
```
Require: $G = (V, E, X), h$
  $c^0_{\vec{v}} \leftarrow h(G[\vec{v}])\ \forall \vec{v} \in V^k$
  repeat
      $c_{\mathcal{N}(\vec{v},i)} \leftarrow \{\!\{ c^{\ell-1}_{\vec{w}} : \vec{w} \in \mathcal{N}(\vec{v}, i) \}\!\}$
                          $\forall i \in [k], \vec{v} \in V^k$
      $c^\ell_{\vec{v}} \leftarrow h\left(c^{\ell-1}_{\vec{v}}, c_{\mathcal{N}(\vec{v},1)}, \ldots, c_{\mathcal{N}(\vec{v},k)}\right)$
                                        $\forall \vec{v} \in V^k$
  until $(c^\ell_{\vec{v}})_{\vec{v} \in V^k} = (c^{\ell-1}_{\vec{v}})_{\vec{v} \in V^k}$
  return $\{\!\{ c^\ell_{\vec{v}} : \vec{v} \in V^k \}\!\}$
```

For more details, see Huang and Villar [2021]. The expressive power of the k-WL test increases with k (Morris et al. [2023]), even though the 1-WL test and the 2-WL test have the same expressive power Grohe [2021].

The connection between WL test and GNNs. The Weisfeiler Leman (WL) test and GNNs are analogous. In both cases, a node's label at the k-th iteration can be interpreted as a hashed representation of a subtree of height k rooted at that node. This connection is fundamental to understanding how the expressive power of GNNs can be analyzed through the WL test. Using this insight, Xu et al. [2019] showed that GNNs are at most as expressive as the WL test and provided sufficient conditions for a GNN to be as expressive as the WL test.

3 Method

In this section, we provide a brief description of the IFG method and its theoretical properties. The main result (Proposition 3.1) is that the expressivity of IFG cannot be better than MPGNN models



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

(a)

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

(b)

Figure 1: Two graphs, with their adjacency matrices, that cannot be distinguished by IFG.

on undirected graphs. To increase the expressive power, we propose an addition invariant, leading to a method we call Augmented IFG (AIFG).

3.1 Expressive power of IFG

In order to construct the invariants for $\mathfrak{S}_n$, Blum-Smith et al. [2024] extends the invariants for Γ by adding one extra invariant that is invariant for $\mathfrak{S}_n$ but not for Γ. Specifically, the IFG method relies on a collection of invariant functions $\{f_k^d, f_l^o, f^\star \mid k \in [n], l \in [n(n-1)/2]\}$ defined by

$$\begin{aligned} f_k^d(X) &= \text{the } k\text{-th largest of the numbers } X_{ii},\ 1 \leq i \leq n, \\ f_l^o(X) &= \text{the } l\text{-th largest of the numbers } X_{ij},\ 1 \leq i < j \leq n, \\ f^\star(X) &= \sum_{1 \leq i,j \leq n, i \neq j} X_{ii} X_{ij}. \end{aligned}$$

For each $X \in \mathcal{S}(n)$, $\mathrm{IFG}(X)$ denotes the $\binom{n+1}{2} + 1$ dimensional vector whose components are the above functions, in some fixed order. Theorem 4.4 of Blum-Smith et al. [2024] shows that IFG can separate almost all orbits except on a measure-zero bad set. IFG can be extended to graphs via $\mathrm{IFG}(G) = \mathrm{IFG}(X(G))$.

Extending IFG to graphs in this manner gives a new way to test whether two graphs are isomorphic. Proposition 3.1 shows that this test is strictly less powerful than the 2-WL test on undirected graphs.

Proposition 3.1. *Let $G_1, G_2 \in \mathcal{G}$ have binary weights and suppose $X(G_1)_{ii} = X(G_2)_{ii} = c$ for all $i \in [n]$, where $c \in \{0, 1\}$. Then* $\mathrm{IFG}(G_1) = \mathrm{IFG}(G_2)$ *whenever the 2-WL test fails to distinguish G_1 and G_2. Further, there exists such $G_1, G_2 \in \mathcal{G}$ which can be distinguished by the 2-WL even though* $\mathrm{IFG}(G_1) = \mathrm{IFG}(G_2)$.

The proof of the proposition is deferred to Appendix A. An example of two graphs where IFG fails, but 2-WL succeeds, is illustrated in Figure 1. Since the 2-WL test has the same expressive power as the 1-WL test Grohe [2021] (and, hence, some GNN models Xu et al. [2019], Morris et al. [2019]), it follows that IFG is strictly worse than some GNNs on such graphs.

3.2 The Augmented IFG

To improve the expressive power of IFG, we augment its invariants by adding another function $f^\dagger$ defined by[3]

$$f^\dagger(X) = \sum_{i,j,k} X_{ij} X_{jk}.$$

This polynomial was introduced in Thiéry [2000]. The proposed AIFG method is defined by $\mathrm{AIFG}(X) = (\mathrm{IFG}(X), f^\dagger(X))$, with the extension to graphs done in the obvious manner. Since $f^\dagger$ is invariant, it follows that AIFG remains invariant. Further, [Theorem 3.1, Blum-Smith et al. [2024]] implied that more invariant functions in the set contribute to reducing the size of the bad

[3]At first sight, this function seems to have a computational complexity of order $\mathcal{O}(n^3)$. However, we show in Appendix B that this complexity can be reduced.

Table 1: Mean Absolute Error (MAE) comparison between the proposed method and different baselines across multiple molecular properties. A lower metric means better performance.

MAE ↓	Atomization PBE0	Excitation ZINDO	Absorption ZINDO	HOMO ZINDO	LUMO ZINDO	1st Excitation ZINDO	Ionization ZINDO
KRR Wu et al. [2018]	9.3	1.83	0.098	0.369	0.361	0.479	0.408
DS-CI Blum-Smith et al. [2024]	12.849 ± 0.757	1.776 ± 0.069	0.086 ± 0.003	0.401 ± 0.017	0.338 ± 0.048	0.492 ± 0.058	0.422 ± 0.012
DTNN Wu et al. [2018]	21.5	1.26	0.074	0.192	0.159	0.296	0.214
DS-CI+ Blum-Smith et al. [2024]	*7.650 ± 0.399*	**1.045 ± 0.030**	*0.069 ± 0.005*	*0.172 ± 0.009*	*0.119 ± 0.005*	**0.160 ± 0.011**	**0.189 ± 0.011**
DS-CAI (Ours)	**7.427 ± 0.281**	*1.060 ± 0.011*	**0.066 ± 0.002**	**0.171 ± 0.004**	**0.116 ± 0.002**	*0.166 ± 0.012*	*0.191 ± 0.004*
MAE ↓	Affinity ZINDO	HOMO KS	LUMO KS	HOMO GW	LUMO GW	Polarizability PBE0	Polarizability SCS
KRR Wu et al. [2018]	0.404	0.272	0.239	0.294	0.236	0.225	0.116
DS-CI Blum-Smith et al. [2024]	0.404 ± 0.047	0.302 ± 0.009	0.225 ± 0.010	0.329 ± 0.016	0.213 ± 0.008	0.255 ± 0.015	0.114 ± 0.008
DTNN Wu et al. [2018]	0.174	**0.155**	**0.129**	**0.166**	**0.139**	0.173	0.149
DS-CI+ Blum-Smith et al. [2024]	**0.122 ± 0.002**	*0.169 ± 0.007*	0.135 ± 0.007	0.183 ± 0.005	*0.139 ± 0.004*	*0.139 ± 0.005*	*0.088 ± 0.004*
DS-CAI (Ours)	*0.125 ± 0.003*	0.170 ± 0.008	*0.131 ± 0.005*	*0.178 ± 0.007*	0.143 ± 0.002	**0.137 ± 0.001**	**0.085 ± 0.005**

set, since the bad set is the intersection of the null space of each invariant function. That is, AIFG is no less expressive than IFG because $\mathrm{IFG}(X_1) \neq \mathrm{IFG}(X_2)$ implies $\mathrm{AIFG}(X_1) \neq \mathrm{AIFG}(X_2)$ for all $X_1, X_2 \in \mathcal{S}(n)$. In fact, AIFG is strictly better than IFG as AIFG can distinguish the graphs in Figure 1 while IFG cannot. Indeed, with G_1 as graph (a) and G_2 as graph (b), we have $f^\dagger(X(G_1)) = 12 \neq 10 = f^\dagger(X(G_2))$.

We present the details of computing $f^\dagger$ with $\mathcal{O}(n^2)$ complexity in Appendix B. A further discussion of the expressive power of $f^\dagger$, and, by consequence, AIFG, is given in Appendix B.1.

4 Empirical evidence

To assess the validity of the proposed AIFG method, we perform experiments with datasets in two domains, i.e., molecule property and point cloud distance predictions. We mimic the Deepset formulation of Blum-Smith et al. [2024], with the addition of the newly proposed invariant feature $f^\dagger(X) = \sum_{i,j,k} X_{ij} X_{jk}$.

4.1 Molecular property prediction on QM7b dataset

For the first experiment, we tested our method for predicting molecular properties using the information from the molecule structure. This task is particularly relevant to this work since a molecule's property is invariant to rotations and reflections of the molecule graph structure and atom relabeling. A standard benchmark in this domain is the QM7b molecule property regression dataset Wu et al. [2018], consisting of 7211 molecules and 14 (property) regression tasks. For each molecule, the input feature matrix $X \in \mathbb{R}^{n \times n}$ is the symmetric Coulomb matrix. Mathematically, one can represent this matrix as follows

$$X = \begin{cases} 0.5 Z_i^{2.4}, & i = j \\ \dfrac{Z_i Z_j}{\|R_i - R_j\|}, & i \neq j \end{cases},$$

where $\{Z\}_i^n, Z_i \in \mathbb{R}$ are the nuclear charges and $\{R\}_i^n, R_i \in \mathbb{R}^3$ are the 3D coordinates of the atomic nucleus.

We use the same architecture described in Blum-Smith et al. [2024] with the addition of the new invariant. Specifically, our model is defined as

$$\text{DS-CAI}(X) = \text{MLP}_c\left(\text{DeepSet}_1\left(\{f_k^d(X)\}_{k=1,\ldots,n}\right), \text{DeepSet}_2\left(\{f_\ell^o(X)\}_{\ell=1,\ldots,n(n-1)/2}\right), \text{MLP}_3(f^\star(X)), \text{MLP}_4(f^\dagger(X))\right).$$

We also apply the same binary expansion used in Blum-Smith et al. [2024] to improve the representational power of the scalar input features. Moreover, we kept the same training/validation/test split used in the original paper. Results are averaged over 10 random data splits.

A particular interest of ours was to check whether our method would be better on QM7b regression tasks where Blum-Smith et al. [2024]'s method did not perform well (i.e., HOMO KS, LUMO KS, and HOMO GW.) As illustrated in Table 1, our method produced either competitive or better results than Blum-Smith et al. DS-CI+ on the other 11 regression tasks, while, consequently, outperforming the other baselines.

Table 2: Performance metrics for DS-CAI, OI-DS, and GIN on training and test sets.

	DS-CAI			**DS-CI**			**OI-DS**			**GIN**		
	Rank Corr. $\uparrow$	MSE $\downarrow$	Rel. Error $\downarrow$	Rank Corr. $\uparrow$	MSE $\downarrow$	Rel. Error $\downarrow$	Rank Corr. $\uparrow$	MSE $\downarrow$	Rel. Error $\downarrow$	Rank Corr. $\uparrow$	MSE $\downarrow$	Rel. Error $\downarrow$
Training set	0.92	0.01	0.40	0.88	0.01	0.53	0.73	0.02	0.60	0.91	0.01	0.35
Test set	0.89	0.02	0.41	0.86	0.02	0.60	0.71	0.03	0.61	0.70	0.05	0.51

Furthermore, the new invariant helped to better predict properties of molecules that DS-CI+ did not do as well, overall. This new function shrank the performance gap between the previous model and well-known baselines without any additional cost on the computational side.

4.2 Point cloud distance prediction on ModelNet10

For the experiments on the point cloud data, we consider the task of predicting the Gromov-Wasserstein (GW) distance between a pair of point clouds of objects from two different classes, taken from Blum-Smith et al. [2024]. We compare the performance of DS-CAI and the $O(d)$-invariant DeepSet (OI-DS) introduced by Blum-Smith et al. [2024] with a shallow Graph Isomorphism Network (GIN) model proposed by Xu et al. [2019] to highlight the relative strengths and limitations of the AIFG method.

We built a shallow 3-layer GIN model whose inputs are symmetric graphs derived from point clouds. Specifically, we construct a k-nearest neighbor (kNN) graph from each point cloud based on the node distances, and use the position vectors as node features. To make it comparable to the k-means method introduced by Blum-Smith et al. [2024], we set k to be 3 when building the kNN graphs. Moreover, a grid search was performed to tune the hyperparameters. It turns out that $k = 3$ yielded performance close to a local optimum. During training, the learning rate was set to be 5×10^{-3} for GIN and DS-CAI and 10^{-2} for OI-DS, with hidden dimension set to 128 for GIN and DS-CAI and 64 for OI-DS.

In terms of performance, as shown in Table 2, DS-CAI demonstrated some advantages over DS-CI[4], OI-DS and GIN on the test dataset, achieving higher rank correlation, lower mean square error, and lower relative error. Visually[5], GIN outputs tended to underestimate the GW distance relative to DS-CAI, but exhibited lower variance on the test set compared to the OI-DS. In terms of efficiency, the GIN model indicated higher computation complexity than OI-DS, as reflected in its longer training time. Therefore, even though OI-DS reported a slightly weaker performance, the computation efficiency can be highly desirable when dealing with large-scale datasets. The DS-CAI had the longest run time among all methods in this experiment. This is primarily because its input is the least processed. The same input (in the form of a fully-connected graph) to a GNN model unsurprisingly requires considerably more computation to achieve the same level of accuracy outputs. Since in practice GNN models would not be applied in this way, we only report the results on the kNN graphs for GIN to ensure a fair comparison for practical use. Overall, depending on the requirements of a given task, users have options to select the model that best suits the needs.

5 Discussions

We have studied the expressive power of IFG and shown that on undirected graphs, it is strictly worse than GNNs. To improve efficiency, we proposed a new method that adds another invariant to IFG. Our new invariant has the same computational complexity as the other invariants in IFG. Therefore, this new invariant does not add to the computational overhead, in big-O sense, while empirically improves the performance of IFG.

A lot of questions remain open. Is IFG strictly worse than GNNs on a larger class of graphs? Can we theoretically quantify the improvement that results from adding $f^{\dagger}$ to IFG? In Blum-Smith et al. [2024], the authors had pointed out that the bad set follows from the non-separable actions by the set of invariant functions. Hence, thoughtfully adding invariant functions can provide tractable way to improve the express power. We hope this work opens the door to potential approaches to answer

[4]Results reported in Blum-Smith et al. [2024]

[5]Due to the lack of space, the qualitative results can be seen in Appendix C

these questions. We are currently exploring ways to address the above questions combining functions that can leverage graph homomorphism illustrated in Thiéry [2000].

References

László Babai, Paul Erdős, and Stanley M. Selkow. Random graph isomorphism. *SIAM Journal on Computing*, 9(3):628–635, 1980. 3

Ben Blum-Smith, Ningyuan Huang, Marco Cuturi, and Soledad Villar. Learning functions on symmetric matrices and point clouds via lightweight invariant features. *arXiv preprint arXiv:2405.08097*, 2024. 1, 2, 4, 5, 6, 8

Martin Grohe. The logic of graph neural networks. In *Symposium on Logic in Computer Science (LICS)*, pages 1–17. IEEE, 2021. 3, 4

Ningyuan Teresa Huang and Soledad 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)*, pages 8533–8537. IEEE, 2021. 3

Chaitanya K. Joshi, Cristian Bodnar, Simon V. Mathis, Taco Cohen, and Pietro Liò. On the expressive power of geometric graph neural networks, 2024. URL `https://arxiv.org/abs/2301.09308`. 2

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 *AAAI Conference on Artificial Intelligence*, pages 4602–4609. AAAI Press, 2019. 4

Christopher Morris, Yaron Lipman, Haggai Maron, Bastian Rieck, Nils M Kriege, Martin Grohe, Matthias Fey, and Karsten Borgwardt. Weisfeiler and leman go machine learning: The story so far. *The Journal of Machine Learning Research*, 24(1):15865–15923, 2023. 2, 3

Maxwell Rosenlicht. Some basic theorems on algebraic groups. *American Journal of Mathematics*, 78(2):401–443, 1956. 3

Nicolas M Thiéry. Algebraic invariants of graphs; a study based on computer exploration. *ACM SIGSAM Bulletin*, 34(3):9–20, 2000. 4, 7

Boris Weisfeiler and Andrey A. Leman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968. 3

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018. 5

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ryGs6iA5Km`. 3, 4, 6

A Proof of Proposition 3.1

Proof. First suppose IFG$(G_1) \neq$ IFG(G_2). We will show that the 2-WL test can distinguish G_1 and G_2. Let

$$f^o(X) = (f^o_1(X), \ldots, f^o_{n(n-1)/2}(X)) \quad \text{and} \quad f^d(X) = (f^d_1(X), \ldots, f^d_n(X)).$$

Since IFG$(G_1) \neq$ IFG(G_2) and $f^d(X(G_1)) = f^d(X(G_2))$ by assumption, it holds that $f^o(X(G_1)) \neq f^o(X(G_2))$ or $f^\star(X(G_1)) \neq f^\star(X(G_2))$. In fact, since the weights are binary, $f^\star(X(G_1)) \neq f^\star(X(G_2))$ holds only if $f^o(X(G_1)) \neq f^o(X(G_2))$, hence we can assume without loss of generality that $f^o(X(G_1)) \neq f^o(X(G_2))$. Since $f^o(X(G_1)) \neq f^o(X(G_2))$ implies G_1 and G_2 have different edge counts, 2-WL distinguishes G_1 and G_2 at initialization.

Now let G_1 and G_2 have adjacency matrices

$$X(G_1) = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad X(G_2) = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

Clearly $f^\star(X(G_1)) = f^\star(X(G_2))$, $f^d(X(G_1)) = f^d(X(G_2))$, and $f^o(X(G_1)) = f^o(X(G_2)) = (1,1,1,0,0,0)$. Therefore IFG$(G_1) =$ IFG(G_2). On the other hand, after 1 iteration, 2-WL distinguishes the two graphs, for example, due to the hash value of the $(1,1)$ ☐

B Computational complexity of $f^\dagger$ function

Claim B.1. *Let $\mathbf{1}$ be the all ones vector in $\mathbb{R}^n$ and $\mathcal{S}(n)$ denote the set of symmetric $n \times n$ matrices. Then for any $X \in \mathcal{S}(n)$, it holds that*

$$f^\dagger(X) = \|X\mathbf{1}\|_2^2.$$

Proof. We have

$$f^\dagger(X) = \sum_{i=1}^n \sum_{k=1}^n \sum_{j=1}^n X_{ij} X_{jk} \overset{(i)}{=} \sum_{i=1}^n \mathbf{1}_i \sum_{k=1}^n (X^2)_{ik} \mathbf{1}_k = \mathbf{1}^\top X^2 \mathbf{1} \overset{(ii)}{=} (X\mathbf{1})^\top X\mathbf{1} = \|X\mathbf{1}\|_2^2,$$

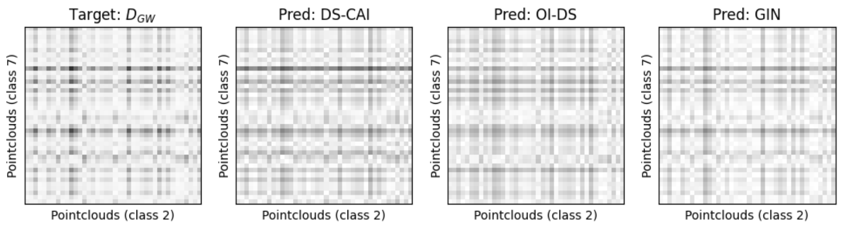
where step (i) follows by definition of matrix multiplication and the fact that $\mathbf{1}_i = \mathbf{1}_k = 1$ for all $i, k \in \{1, \ldots, n\}$, while step (ii) follows because X is symmetric. ☐

The above shows that the new invariant function can be reduced to a matrix-vector multiplication, which for symmetric matrices has complexity at most $\mathcal{O}(n^2)$. More important than the reduction in computational complexity ($\mathcal{O}(n^2)$ *vs.* $\mathcal{O}(n^3)$), in practice the new invariant does not add any significant overhead to the method proposed in Blum-Smith et al. [2024], since its computation can be separated in two steps: (i) taking the row-sum vector and (ii) computing its L_2 norm. The former is exactly an intermediate procedure in the implemented algorithm from the aforementioned work to compute the previous invariant features. Thus, the only contribution comes from the cheap norm calculation.

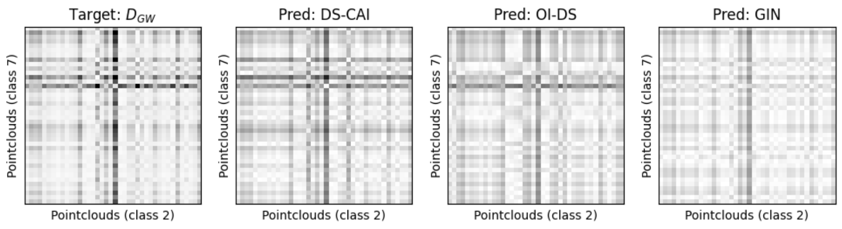
B.1 Analysis of the expressive power of $f^\dagger$ function

An immediate consequence of the above claim B.1 is that the invariant $f^\dagger(X) = \sum_{i,j,k} X_{ij} X_{jk}$ cannot distinguish any $X, Y \in \mathcal{S}(n)$ satisfying $\|X\mathbf{1}\|_2^2 = \|Y\mathbf{1}\|_2^2$. In particular, if X and Y are adjacency matrices of two *regular graphs*, then $f^\dagger$ cannot distinguish the two graphs.

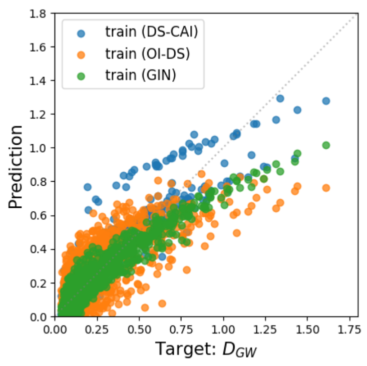
C Results for point cloud distance prediction on ModelNet10



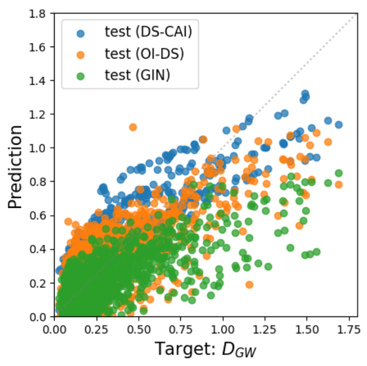
(a) Training set distances



(b) Test set distances



(c) Training set correlation



(d) Test set correlation

	DS-CAI	OI-DS	GIN
Training time	335s	6s	63s

(e) Training time (in seconds) for DS-CAI, OI-DS, and GIN after data processing

Figure 2: Qualitative results for predicting the point cloud distances on ModelNet10 benchmark. We compare the discussed methods, DS-CAI and OI-DS, with a shallow GIN. All three methods performed reasonably well. Panels (a) and (b) represent the target distance and predicted distance by the models. The leftmost grid depicts the ground truth while the rest three displays predicted distances by the models. Panels (c) and (d) provide visual comparison of the target vs. predicted distance, with the diagonal line indicating the ground truth. OI-DS produces wider-spread outputs than the other two methods. Lastly, panel (e) demonstrates the training times corresponding to each method.