Invariant Theory vs GNNs: An Empirical Comparison

Shiyu Chen
Department of Applied Mathematics and Statistics
Johns Hopkins University
schen355@jh.edu

Abstract

Implementing the intrinsic symmetries of the underlying objects is crucial for the success of machine-learning models on non-Euclidean data such as graphs and point clouds. A recent paper proposes an invariant theory approach to define invariant functions on symmetric matrices and point clouds, yet its empirical value relative to modern neural methods is unclear. We compare this approach with strong graph neural network (GNN) baselines on two benchmarks. For molecular property prediction from Coulomb matrices, invariant features are more accurate but GNNs run an order of magnitude faster. For point-cloud distance prediction, GNN matches the accuracy of invariant features while showing far lower run-to-run variance. Through our experiments, we observe that explicit invariant features excel when the symmetry is hard to encode as a graph, whereas GNNs offer superior efficiency and stability. This motivates hybrid designs combining the two paradigms.

1 Introduction

Many machine learning applications[1, 2] involve data with underlying symmetries, particularly in non-Euclidean domains such as graphs and point clouds. Exploiting these symmetries in model design has been shown to improve both predictive performance and data efficiency [3, 4, 5, 6, 7, 8, 9]. For instance, symmetry–aware architectures appear in both structured and unstructured settings: Graph neural networks (GNNs) are permutation–equivariant to node relabelling by construction [10, 8], whereas set models such as DeepSets [7] and point-cloud networks like PointNet [11] attain the same permutation invariance through symmetric aggregations (e.g., sum, max, or attention pooling) over unordered elements. When inputs also contain spatial coordinates—as in 3-D vision, coordinate-based molecular modeling, or generic point-cloud tasks—models must additionally respect continuous Euclidean symmetries (rotations, reflections, translations), motivating SE(3)-equivariant architectures [12]. The widespread success of such symmetry-aware models has spurred growing interest in principled approaches to incorporate domain symmetries into machine learning algorithms.

Recent work has begun importing tools from invariant theory—the study of functions unchanged by group actions—into machine learning. Rather than asking a network to learn invariances, one can first construct features that are provably invariant under the target symmetry group and then feed them to any learner. Blum-Smith *et al.* [13] show how to do this for two cases relevant here: (i) symmetric matrices that represent weighted graphs (permutation symmetry of rows/columns) and (ii) point clouds in R^d (permutation and Euclidean symmetries). Their method enumerates a compact separating set of fundamental invariants—$O(n^2)$ for an $n \times n$ matrix or $O(n)$ for n points—so that any two non-equivalent inputs map to different feature vectors. Combined with DeepSets, these invariant-function models achieve superior performance on Coulomb-matrix molecular regression and point-cloud distance prediction, suggesting that explicitly constructed invariant features can serve as a universal, symmetry-aware representation.

Preprint. Under review.

On the other hand, modern GNNs provide a flexible and scalable alternative that implicitly learns to respect symmetries through architectural design. They processes graph-structured data by iteratively aggregating information from neighboring nodes, ensuring that any permutation of the input nodes produces an equivalently permuted set of node embeddings. For graph-level outputs, a final symmetric readout function (such as summing or averaging node features) is used to yield a permutation-invariant prediction [10]. These architectures have achieved state-of-the-art results across many applications. For instance, message-passing networks are a standard tool for predicting molecular properties from graphs of atoms [8], and point cloud GNNs can efficiently handle large 3D datasets by learning local geometric features [14]. GNNs thus naturally handle the combinatorial symmetry of permutations while learning complex representations in a data-driven way.

However, since GNNs rely on finite training data and bounded-depth message passing, it is not guaranteed that they will capture every high-order invariant such as long-range, multi-node symmetry patterns [15], nor all subtle structural distinctions arising from small motifs or slight geometric deformations that can decisively influence the target [16]. In fact, there are known theoretical limits on the expressive power of standard GNNs for distinguishing certain graph structures [10]. This gap between guaranteed invariance (through explicit features) and learned invariance (through GNNs) motivates a closer investigation. In this paper, we present an empirical comparison of the invariant-theoretic approach versus GNNs on tasks that demand strict symmetry preservation. We consider two tasks: (i) Molecular property prediction: Each molecule is represented by a Coulomb matrix— a symmetric matrix of inter-atomic potentials without atom labels. So the predictor must be invariant to any permutation of rows and columns; (ii) Point-cloud distance prediction: The target depends only on an unordered 3-D point set and must likewise be invariant to global rotations, translations, and permutations of the points. For each task we compare a strong GNN baseline (Graph Isomorphism Network (GIN) [10] and its variants for Coulomb matrices, Point-GNN [14] for point clouds) with the invariant-function models of Blum-Smith *et al.* [13]. We report accuracy and run-time, tracking how both approaches scale with the number of atoms or points.

In the molecular property prediction task, invariant-function models tend to achieve a lower classification error since the input features already encode the necessary symmetries. GNNs fall slightly behind because a Coulomb matrix induces a fully connected graph, and message-passing networks which are optimized for sparse topologies find it difficult to extract useful structure from such dense connectivity. However, GNNs run markedly faster, whereas the invariant-function models incur a higher cost by employing high-dimensional node embeddings to capture molecular symmetry. In the point cloud task, invariant-function models are noticeably unstable——repeated runs yield widely scattered results, evidenced by a high variance in performance metrics. In contrast, the GNN baseline attains comparable accuracy while exhibiting far lower variance, demonstrating much greater stability.

Explicit invariant-function models guarantee perfect adherence to the target symmetry and often reach higher accuracy when the data provide no reliable graph structure (e.g., fully-connected Coulomb matrices).They are data-efficient and mathematically interpretable, but their reliance on very high-dimensional hidden representations makes them computationally costly and prone to numerical instability. GNNs learn invariance implicitly through message passing. They scale linearly, train and infer faster, readily absorb auxiliary attributes, and show greater run-to-run stability. However, with only a few layers and limited data, they can miss long-range patterns, and they have trouble when the graph is very dense or poorly defined. Therefore, combining the two approaches is a promising direction for future research.

2 Preliminaries

Throughout, a graph is $G = (V, E)$ and a point cloud is $P = \{p_1, \ldots, p_n\} \subset R^3$.

2.1 Graph Isomorphism Network (GIN)

GIN [10] updates each node v by summing neighbour features and applying an MLP:

$$h_v^{(l+1)} = \; MLP^{(l)}\Big((1+\epsilon)\, h_v^{(l)} + \sum_{u \in \mathcal{N}(v)} h_u^{(l)}\Big), \tag{1}$$

where $h_v^{(l)}$ is the feature of node v from the $l-th$ layer, ϵ is a scalar (fixed or learned) and $\mathcal{N}(v)$ denotes the neighbours of v. The summation makes the update permutation-invariant. After L

iterations, a readout function generates a graph-level representation h_G:

$$h_G \;=\; \mathrm{READOUT}\Big(\{\, h_v^{(L)} \mid v \in G \,\}\Big). \tag{2}$$

where $\mathrm{READOUT}$ can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function [17, 18].

2.2 Point-GNN

Point-GNN [14] operates on a radius graph built from a point cloud $P = \{(pos_i, x_i)\}$, where $pos_i \in R^3$ is a coordinate and $x_i \in R^{F_{in}}$ is an initial feature. Each layer applies three learned MLPs $(h_\Theta, f_\Theta, g_\Theta)$:

$$\Delta pos_i = h_\Theta(x_i), \tag{3}$$

$$e_{j,i} = f_\Theta\big(pos_j - pos_i + \Delta pos_i,\; x_j\big), \tag{4}$$

$$x_i' = g_\Theta\Big(\max_{j \in \mathcal{N}(i)} e_{j,i}\Big) + x_i. \tag{5}$$

where Δpos_i is an alignment offset that recentres the local neighbourhood, providing global translation invariance. $e_{j,i}$ is the edge message derived from the relative position $pos_j - pos_i + \Delta pos_i$ together with the neighbour feature x_j and the permutation-invariant aggregation $\max_{j \in \mathcal{N}(i)} e_{j,i}$ yields a neighbourhood summary that g_Θ maps back to the feature space.

2.3 Invariant-function models

We adopt DeepSet for Conjugation Invariance (DS-CI) and its extension DS-CI+ for symmetric matrices and O(d)-Invariant DeepSet (OI-DS) for point clouds [13].

2.3.1 DS-CI

Given a space $\mathcal{X} \subset R^d$, a DeepSet [7] is a parametric function that takes a finite multiset of elements from $\mathcal{X}$ and returns a vector in $R^{d'}$:

$$\mathrm{DeepSet}\big(\{x_i \in \mathcal{X} : i \in I\}\big) \;:=\; \sigma\left(\sum_{i \in I} \phi(x_i)\right). \tag{6}$$

where ϕ and σ are typically multi-layer perceptrons (MLPs), and I indexes are the (finite) input multiset.

Given a symmetric matrix $X \in R^{n \times n}$, define

$$\mathcal{D}(X) = \{X_{ii}\}_{i=1}^n, \quad \mathcal{O}(X) = \big\{X_{ij} \;\mid\; 1 \le i < j \le n\big\}, \quad f^*(X) = \sum_{i \neq j} X_{ii} X_{ij}. \tag{7}$$

Diagonal and off–diagonal scalars are aggregated separately, then concatenated with a global interaction term, we obtain the DS-CI:

$$\text{DS-CI}(X) = \mathrm{MLP}\Big(\mathrm{DeepSet}\big(\mathcal{D}(X),\; \mathrm{DeepSet}\big(\mathcal{O}(X)\big),\; \mathrm{MLP}\big(f^*(X)\big)\Big)\Big). \tag{8}$$

DS-CI+ employs the binary expansion [19] which is a standard step in molecular-property regression. Specifically, the expansion $\phi : R \to [0,1]^d$ lifts a scalar into a higher-dimensional embedding

$$\phi(x) = \big[\ldots,\; \mathrm{sigmoid}\Big(\tfrac{x-\theta}{\theta}\Big),\; \mathrm{sigmoid}\Big(\tfrac{x}{\theta}\Big),\; \mathrm{sigmoid}\Big(\tfrac{x+\theta}{\theta}\Big), \ldots\big]. \tag{9}$$

where $\mathrm{sigmoid}(x) = e^x/(1+e^x)$. We set the width to $d = 100$ and the spacing parameter to $\theta = 1$, following the original recommendation.

2.4 OI-DS

For a point cloud P, select $k = d$ reference vectors $C_P = [c_1, \ldots, c_k] \in R^{d\times k}$, taken here as the k-means centroids ordered by norm. Projecting each point onto these references yields an $SE(3)$-invariant multiset:

$$\mathcal{H}(P) = \{C_P^\top p \mid p \in P\} \subset R^k.$$

A DeepSet on $\mathcal{H}(P)$ pooled together with a global second-moment term forms the model:

$$\text{OI-DS}(P) = \text{MLP}\Big(\text{DeepSet}(\mathcal{H}(P)),\ \text{MLP}_4(C_P^\top C_P)\Big). \tag{10}$$

Compared to DS-CI, the number of invariant features here are reduced from $O(n^2)$ to $O(n)$.

3 Baselines

3.1 GIN and its variants

Node features and edges prepocessing. Each diagonal entry is lifted to a d-dimensional binary–expansion vector as in DS-CI$^+$: $h_i^{(0)} = \phi\big(X_{ii}\big) \in R^d$ and used as the initial node feature. For each edge (i, j), let $w_{ij} = X_{ij}$ and set $e_{ij} = \phi(w_{ij}) \in R^d$ when required.

GIN backbone. The message passing of vanilla GIN can be rewrote as $h_i^{(l+1)} = MLP^{(l)}\Big((1 + \varepsilon)\, h_i^{(l)} + \sum_{j\in\mathcal{N}(i)} m_{ij}^{(l)}\Big)$, where $m_{ij}^{(l)} = h_j^{(l)}$. Based on this, we constructed GIN-weighted and GIN-concat.

GIN–weighted. Edge weights scale the neighbour embeddings before the MLP:

$$m_{ij}^{(l)} = MLP^{(l)}\big(w_{ij}\, h_j^{(l)}\big).$$

GIN–concat. A binary–expanded edge feature is concatenated to the neighbour state:

$$m_{ij}^{(l)} = MLP^{(l)}\big([\, h_j^{(l)} \| e_{ij}\,]\big).$$

3.2 Point-GNN pipeline.

For each point cloud P, we build a radius graph $G = (P, E)$ where $E = \big\{(p_i, p_j) \;\big|\; \|p_i - p_j\|_2 < r\big\}$. Relative coordinates are embedded into node features by

$$h_i^{(0)} = \max_{j\in\mathcal{N}(i)} h_\theta\big(p_j - p_i\big), \tag{2}$$

where $h_\theta : \mathbb{R}^3 \to \mathbb{R}^d$ is a shared MLP and $\max$ is taken element-wise, yielding translation-invariant initial states. The graph is then processed by L standard Point-GNN layers.

To estimate a symmetry-invariant distance between two clouds P, P' we feed their embeddings through a Siamese head which is the same as DS-CI/DS-CI+:

$$\widehat{\text{GW}}(P, P') = a\, \big\| W\big(F(P) - F(P')\big)\big\|_2^2 + b, \tag{3}$$

where F denotes the Point-GNN encoder, $W \in \mathbb{R}^{k\times d}$ is a learnable projection and a, b are trainable scalars.

4 Experiments

We adopt the same tasks and data from [13]. All experiments were executed locally on a MacBook Air (Apple M2 SoC, 8-core CPU, 10-core GPU, 16 GB unified memory, 15-inch 2023) running macOS 13.4 (Ventura). The software stack comprised Python 3.10 and PyTorch 2.6 with the Metal Performance Shaders (MPS) backend.

4.1 Molecular property prediction (QM7b)

We work on the QM7b benchmark [20, 19], which contains $7\,211$ molecules with 14 regression targets. Each molecule is encoded by an $n \times n$ Coulomb matrix X whose entries depend only on nuclear charges $Z_i \in R$ and 3-D coordinates $R_i \in R^3$:

$$X_{ij} = \begin{cases} 0.5\, Z_i^{2.4}, & i = j, \\ \dfrac{Z_i Z_j}{\|R_i - R_j\|}, & i \neq j, \end{cases} \tag{11}$$

Then we build the graph $G = (V, E)$ with $V = \{1, \ldots, n\}$ and $E = \{(i, j) \mid i \neq j\}$, i.e. every atom pair is connected while self–loops (i, i) are removed. We predict each target separately and report mean-absolute error (MAE).

For a fair comparison with DS-CI/DS-CI+ we keep the training protocol identical wherever possible. All GNN models are trained with a batch size of 128 and the Adam optimizer [21]: the initial learning rate is 5×10^{-3}, weight decay 10^{-5}, and a cosine–plateau scheduler reduces the step size by a factor 0.8 after five epochs without validation improvement (minimum 10^{-5}). Early-stopping is triggered after 20 idle epochs or when the run reaches 1000 epochs.

Evaluation follows stratified ten-fold cross-validation. For every fold we hold out 10% of the data as a test set and split the remaining 90% into 9:1 train/validation. We report the mean $\pm$ standard error of the ten test MAEs and collect them in Table 1. We also log, for target 0, the wall-clock time required by each method to complete one fold and the average time per training step. The results are summarized in Table 2.

The GNN variants outperform DS-CI but still lag behind DS-CI+, indicating that they do not fully capture the molecular structure encoded in a Coulomb matrix. The shortfall is expected: when we treat the Coulomb matrix as an adjacency matrix, every node is connected to every other, whereas message-passing GNNs are designed for and derive their expressive power from sparse neighbourhoods. In contrast, the GNNs run much faster than the invariant-feature baselines because they skip the high-dimensional hidden dimensions and the multiple MLP blocks that DS-CI/DS-CI+ require to learn symmetric representations.

Target	GIN backbone	GIN-weighted	GIN-concat	DS-CI	DS-CI+
0	15.001 ± 0.138	11.597 ± 0.091	13.663 ± 0.150	12.849 ± 0.757	**7.650 ± 0.399**
1	2.255 ± 0.012	1.079 ± 0.011	1.313 ± 0.049	1.776±0.069	**1.045 ± 0.030**
2	0.118 ± 0.001	0.077 ± 0.002	0.072 ± 0.001	0.086±0.003	**0.069 ± 0.005**
3	0.667 ± 0.003	0.335 ± 0.010	0.271 ± 0.018	0.401±0.017	**0.172 ± 0.009**
4	0.662 ± 0.006	0.148 ± 0.008	**0.100 ± 0.004**	0.338±0.048	0.119 ± 0.005
5	0.897 ± 0.010	0.288 ± 0.017	0.220 ± 0.010	0.492±0.058	**0.160 ± 0.011**
6	0.687 ± 0.003	0.392 ± 0.009	0.302 ± 0.020	0.422±0.012	**0.189 ± 0.011**
7	0.725 ± 0.007	0.146 ± 0.012	**0.108 ± 0.004**	0.404±0.047	0.122 ± 0.002
8	0.431 ± 0.003	0.272 ± 0.003	0.232 ± 0.009	0.302±0.009	**0.169 ± 0.007**
9	0.329 ± 0.005	0.182 ± 0.003	0.139 ± 0.004	0.225±0.01	**0.135 ± 0.007**
10	0.487 ± 0.003	0.305 ± 0.003	0.260 ± 0.013	0.329±0.016	**0.183 ± 0.005**
11	0.240 ± 0.002	0.179 ± 0.003	0.143 ± 0.008	0.213±0.008	**0.139 ± 0.004**
12	0.600 ± 0.005	0.211 ± 0.006	0.219 ± 0.008	0.255±0.015	**0.139 ± 0.005**
13	0.526 ± 0.004	0.105 ± 0.002	0.126 ± 0.003	0.114±0.008	**0.088 ± 0.004**

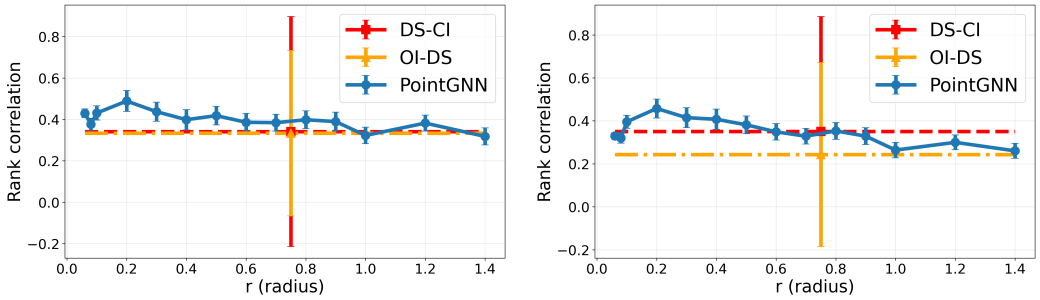
Table 1: Comparison of MAE (Mean ± Standard Error) among GIN backbone, GIN-weighted, GIN-concat, DS-CI and DS-CI+ on QM7b. Best results are highlighted in bold.

Time	GIN backbone	GIN-weighted	GIN-concat	DS-CI	DS-CI+
t_1	0.85s	2.85s	3.45s	11.87s	12.02s
t_2	3.37min	12.02min	12.43min	3.17h	3.33h

Table 2: Comparison of time among GIN backbone, GIN-weighted, GIN-concat, DS-CI and DS-CI+ on target 0 of QM7b. t_1 is the average training time per step and t_2 denotes the total running time of one fold.

r	0.06	0.08	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00	1.20	1.40
deg(training "2")	0.41	0.72	1.15	4.54	9.78	16.46	24.48	33.54	43.27	53.16	62.57	71.04	84.33	92.32
deg(test "2")	0.33	0.62	1.00	4.10	8.97	15.43	23.08	31.56	40.58	49.94	59.31	68.21	82.48	91.37
deg(training "7")	0.21	0.42	0.70	3.62	9.21	16.68	25.23	34.32	43.40	51.96	59.83	66.87	78.73	87.36
deg(test "7")	0.20	0.38	0.64	3.38	8.43	15.24	23.30	31.98	40.84	49.27	57.08	64.22	76.28	85.18

Table 3: Average node degree of graphs constructed with different radius r where the number in "" refers to the class in ModelNet10.



(a) Comparison of Performance on training data

(b) Comparison of Performance on test data

Figure 1: Error bars of MAEs for Point-GNN, DS-CI, OS-DI based on 50 runs for each r ranging from 0.06 to 1.40.

4.2 Point-cloud distance prediction (ModelNet10 subset).

In this task, our goal is to learn a function that is invariant to point permutations and global $SE(3)$ transformations (rotations, reflections, translations) and approximates a ground-truth distance between two point clouds. We sample 80 meshes from ModelNet10 class 2 and 80 from class 7 [22], uniformly resample each surface to 100 points, and split each class into 40 training and 40 test shapes. All $40 \times 40 = 1600$ cross-class pairs form the train (or test) set. The target is the third lower bound of the Gromov–Wasserstein distance $GW(P, P')$ between point cloud P and P' introduced by [23]. Performance is measured by Spearman rank correlation, mean-square error (MSE) between predicted and true distances. We keep all the parameters the same as the settings of OI-DS. We use a hidden and output dimensionality of $d_{\text{hid}} = d_{\text{out}} = 16$, and set the input dimension to 1. Optimization is performed using Adam with a learning rate of 1×10^{-2} for all submodules, except for the top-level regressor which uses 1×10^{-3}.

To identify an appropriate neighbourhood radius r for Point-GNN, we sweep r from 0.06 to 1.40 and construct a radius graph at each value. Table 3 reports the resulting average node degree. As anticipated, the degree rises almost quadratically with r, quickly transitioning from a sparse to a fully-connected regime. For every radius-specific graph we train a Point-GNN Siamese model to predict the Gromov–Wasserstein distance between point clouds. Each setting is repeated 50 times with different random seeds. We report the mean and standard error of the MAE for both training and

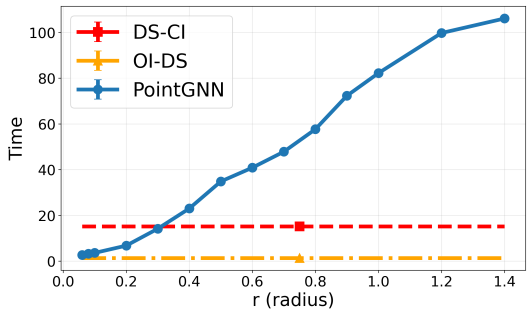


Figure 2: Error bars of times for Point-GNN, DS-CI, OS-DI based on 50 runs for each r ranging from 0.06 to 1.40.

test splits, visualised in Figure1. Training time is also recorded for every run and the average times with their standard errors are plotted in Figure 2. We have the following obervations:

- **Stability**. DS–CI and OI–DS exhibit large error bars, reflecting seed-dependent reference sets and deep, uncoupled MLP stacks; Point-GNN shows much smaller variance because its weights are shared across nodes and do not depend on external reference points.
- **Radius sensitivity**. Point-GNN attains its highest rank-correlation at $r \approx 0.20$. Smaller radii under-connect the graph, while larger ones over-smooth features and add noise.
- **Runtime trade-off**. Point-GNN's cost grows roughly with the mean degree ($\mathcal{O}(r^2)$); at the sweet-spot $r = 0.20$ it needs ~ 3s per fold—about $5\times$ faster than DS-CI—yet remains far more stable. OI-DS is always the quickest but delivers the lowest accuracy.

5 Conclusion

In this work, we systematically compare invariant theory approach with Graph Neural Networks that learn invariances via message passing on two tasks: Coulomb-matrix molecular regression and point-cloud distance prediction. We find that invariant-function models deliver the highest accuracy when the graph is fully connected or poorly defined. They capture all permutation and Euclidean symmetries but incur heavy computational and numerical overhead from multiple MLPs. In contrast, GNNs are faster and exhibit far greater run-to-run stability, though their message passing can fail to recover long-range or high-order invariants within dense graphs. In point cloud task, carefully choosing a radius graph allows Point-GNN to match invariant-function models accuracy while maintaining low variance and linear cost. These complementary strengths point to hybrid architectures, combining explicit invariant features with sparse message passing, as a promising path forward in geometric learning.

6 Acknowledgements

This report was submitted as the final project for the course EN.553.743, Equivariant Machine Learning at Johns Hopkins University in the spring of 2025. I am grateful to Professor Soledad Villar for the engaging lectures and insightful feedback that guided this work. I would also like to thank Ningyuan (Teresa) Huang for her mentorship.

References

[1] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[2] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[3] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

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

[5] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[6] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2282–2290, 2017.

[7] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[8] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[9] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[10] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[11] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[12] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in neural information processing systems*, 33:1970–1981, 2020.

[13] Ben Blum-Smith, Ningyuan Huang, Marco Cuturi, and Soledad Villar. A galois theorem for machine learning: Functions on symmetric matrices and point clouds via lightweight invariant features. *arXiv preprint arXiv:2405.08097*, 2024.

[14] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020.

[15] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[16] Andreas Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

[17] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

[18] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[19] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. *New Journal of Physics*, 15(9):095003, 2013.

[20] Lorenz C Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database gdb-13. *Journal of the American Chemical Society*, 131(25):8732–8733, 2009.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[23] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11:417–487, 2011.