
On Graph Neural Networks for Geometric Data

Giorgio Bertone
e12402238@student.tuwien.ac.at
TU Wien

Abstract

This seminar paper analyzes the expressive power of geometric Graph Neural Networks (GNNs), focusing on their ability to distinguish non-isomorphic geometric graphs through the Geometric Weisfeiler-Leman (GWL) test framework. This analysis is relevant because existing theoretical frameworks for GNNs, such as the Weisfeiler-Leman (WL) test, fail when applied to geometric graphs, as they do not account for the inherent physical symmetries of these structures. The GWL test, specifically designed for geometric graphs, addresses these limitations. Geometric GNNs extend the capabilities of standard GNNs when applied to geometric graphs, which are common in scientific and engineering domains, by incorporating physical symmetries. Examples of such domains include biochemistry (molecular structures), material sciences (crystalline structures), physics (physical simulations of complex systems), and robotics. This paper highlights the limitations of invariant GNNs, which are unable to distinguish graphs with identical one-hop neighborhoods and cannot capture global geometric properties. It then explores how equivariant GNNs overcome these limitations by propagating geometric information beyond local neighborhoods. Furthermore, the paper investigates key factors influencing the expressive power of geometric GNNs, including network depth, the body order of scalarization, and the tensor order of features. Finally, synthetic and real data experiments are conducted to support the theoretical findings, demonstrating their practical implications.

1 Introduction

Geometric GNNs are very effective for modeling systems with geometric and relational structures. These systems appear across a wide range of fields, including **biochemistry** (e.g., small molecules, proteins, and DNA/RNA) as highlighted by [Jamasb et al. \[2022\]](#), **materials science** (e.g., inorganic crystals, catalysis systems) as demonstrated by [Chanussot et al. \[2021\]](#), and **physical simulations** as shown by [Sanchez-Gonzalez et al. \[2020\]](#). Beyond these domains, geometric GNNs are also applied in areas such as transportation and logistics, robotic navigation, multiagent robotics [[Li et al., 2020](#)] and 3D computer vision. These types of systems can be modeled through geometric graphs embedded in the Euclidean space. Furthermore, they have also shown promising results as generative models for geometric graphs (e.g., in diffusion models for protein design [[Baek et al., 2021](#)]), as well as molecular dynamics simulators in autoregressive settings [[Batzner et al., 2022](#)].

Given a geometric graph, standard GNNs are ill-suited to handle it because they do not take spatial symmetries into account. However, geometric GNNs have emerged as the state of the art for making predictions about their properties. They still follow the message passing paradigm [[Gilmer et al., 2017](#)], where neighboring nodes exchange information and influence the embeddings of each other. Specifically, features from local neighborhoods are aggregated and used to update node features in a permutation equivariant manner. The most important property is that the updated geometric features of the nodes retain the transformation semantics of the initial attributes.

Despite these successes, important theoretical questions remain unanswered. Crucially, a deeper understanding of the **design space of geometric GNNs** is needed to further advance the field, as well as a more rigorous analysis of their expressivity. In particular, comprehending how key factors such as the **body order of the layer** (which defines the number of nodes required for feature computation), **depth** (the number of layers), and **tensor order of features** (how features transform under geometric operations) determine the expressivity and effectiveness of a geometric GNN.

The Weisfeiler-Leman (WL) test [Weisfeiler and Leman, 1968] is widely used to analyze the expressive power of standard GNNs. However, it does not apply to geometric graphs as it fails to account for geometric transformations (e.g. rotations and translations) that affect node positions and geometric features.

Thus, Joshi et al. [2023] study the expressive power of geometric GNNs from the perspective of discriminating non-isomorphic geometric graphs. To achieve this they propose an extension of the WL test, called Geometric Weisfeiler-Leman (GWL) test, which they prove to be an upper bound on the expressive power of geometric GNNs. Then, they use the GWL framework to understand how design choices influence geometric GNNs expressivity.

Background

A **standard graph** is a set of nodes connected by edges, where each node can also have scalar features. It can be represented as:

$$\mathcal{G} = (A, S)$$

where A is the $n \times n$ adjacency matrix and $S \in \mathbb{R}^{n \times f}$ are the scalar features.

In contrast, in **geometric graphs** each node is embedded in the Euclidean space and associated with both scalar features and geometric attributes (such as position and orientation). A geometric graph can thus be defined as:

$$\mathcal{G} = (A, S, \vec{X}, \vec{V})$$

where $\vec{X} \in \mathbb{R}^{n \times d}$ are the coordinates of the nodes in d -dimensional space, and $\vec{V} \in \mathbb{R}^{n \times d}$ are the geometric features of the nodes (e.g., velocities, accelerations).

Physical symmetries. The key difference between traditional graphs and geometric graphs lies in the transformation behavior under Euclidean transformations (such as rotations, translations, and reflections). While the scalar features S remain unchanged under these transformations, the geometric attributes (i.e., the vectors \vec{V} and the coordinates \vec{X}) transform accordingly. In other words, geometric graphs have additional symmetries, and while geometric GNNs are designed to handle these transformations, ensuring that both scalar and geometric features are updated in a way that preserves the transformation semantics, standard GNNs are not able to provide these guarantees.

Message Passing GNNs. Message Passing Graph Neural Networks (MPGNNs) leverage the structure of graphs to learn meaningful representations by iteratively updating node features using local neighborhood information. The nodes exchange "messages" with their neighbors to aggregate information and refine their own features. The key idea is that each node gathers information from its immediate neighbors at every layer, effectively building a **computation tree**. As the network deepens, the receptive field of each node expands, enabling it to incorporate features from larger neighborhoods. This process can be expressed in two main steps:

1. **Message Computation:** Each edge (i, j) computes a message m_{ij} based on the features of the connected nodes:

$$m_{ij} = f(s_i, s_j)$$

Here, s_i and s_j are the features of nodes i and j , and f is a function that encodes their interaction.

2. **Message Aggregation and Node Update:** Each node aggregates incoming messages from its neighbors (e.g., using sum, mean, or max pooling) and updates its features:

$$s_i^{(t+1)} = \text{UPD} \left(s_i^{(t)}, \text{AGG}(\{m_{ij} : j \in \mathcal{N}(i)\}) \right)$$

where $s_i^{(t)}$ is node i 's features at layer t , $\mathcal{N}(i)$ is the set of neighbors of node i , AGG is a function that combines messages from neighbors (e.g., summation), UPD is a function that refines the node's features using the aggregated messages.

Graph Isomorphism. Two attributed graphs \mathcal{G}, \mathcal{H} are **isomorphic** if there exists a bijection

$$b : V(\mathcal{G}) \rightarrow V(\mathcal{H})$$

between their vertex sets that preserves both edge structure and attributes. Specifically, this means that two vertices u, v in \mathcal{G} are adjacent if and only if their mapped counterparts $b(u), b(v)$ in \mathcal{H} are also adjacent. Additionally, the attributes of each vertex remain unchanged under the mapping so that

$$s_i^{(\mathcal{G})} = s_{b(i)}^{(\mathcal{H})}$$

In other words, the problem asks whether two graphs are the same, but drawn differently.

Weisfeiler-Leman Test (WL). The Weisfeiler-Lehman (WL) algorithm is used to determine whether two graphs are isomorphic. At initialization, each node i in the graph is assigned a color $c_i^{(0)} \in C$ from a countable space of colors C , where nodes with identical features receive the same initial color. At each iteration t , the algorithm updates the color $c_i^{(t)}$ of each node i by gathering the multiset of colors from its neighbors, concatenating this multiset with the node's current color $c_i^{(t-1)}$, and applying a hash function that assigns a unique color to each distinct input:

$$c_i^{(t)} = \text{HASH} \left(c_i^{(t-1)}, \{ \{ c_j^{(t-1)} \mid j \in N_i \} \} \right)$$

The process stops when the partitioning does not change anymore across iterations. After this refinement step, the algorithm compares the multiset of final colors between the two graphs. If the multisets differ, the graphs are not isomorphic. If they are identical, the graphs may be isomorphic but require further verification.

Message passing GNNs can be at most as powerful as WL at distinguishing non-isomorphic graphs. Thus, in all the cases where WL will fail, message-passing GNNs will also fail.

Group Theory. A group is a set \mathfrak{G} of elements equipped with a binary operation $\circ : \mathfrak{G} \times \mathfrak{G} \rightarrow \mathfrak{G}$ that satisfies the following fundamental properties:

- **Associativity:** For all elements $a, b, c \in \mathfrak{G}$, the operation satisfies $(a \circ b) \circ c = a \circ (b \circ c)$
- **Identity:** There exists a unique element e in the group satisfying $a \circ e = e \circ a = a$ for all $a \in \mathfrak{G}$
- **Inverse:** For every element a in the group, there exists an inverse element a^{-1} such that $a \circ a^{-1} = a^{-1} \circ a = e$
- **Closure:** $a \circ b \in \mathfrak{G}$ for all $a, b \in \mathfrak{G}$

A group can "act" on a space, meaning it transforms the elements of the space in a structured manner (such as rotations or shifts). The action of a group \mathfrak{G} on a space X is denoted by $\mathfrak{g} \cdot x$, where $\mathfrak{g} \in \mathfrak{G}$ and $x \in X$. Given two spaces X and Y on which the group \mathfrak{G} acts, and a function $f : X \rightarrow Y$, we categorize the function f as follows: (1) \mathfrak{G} -invariant if $f(\mathfrak{g} \cdot x) = f(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in X$, i.e. the output of the function does not change when the input undergoes a transformation by \mathfrak{g} (for example, if X is a square and \mathfrak{G} represents rotations, a \mathfrak{G} -invariant function might return the area of the square, which remains the same regardless of how the square is rotated); (2) \mathfrak{G} -equivariant if $f(\mathfrak{g} \cdot x) = \mathfrak{g} \cdot f(x)$ for all $\mathfrak{g} \in \mathfrak{G}$ and $x \in X$, i.e. applying the group transformation to the input results in an equivalent transformation of the output (for example, if X represents a set of points in a plane and Y represents their locations, with \mathfrak{G} as shifts, then a \mathfrak{G} -equivariant function might compute the center of mass of the points, which shifts in the same way as the points when all points are translated by the same amount).

When a group \mathfrak{G} acts on a space X , each element $x \in X$ can be transformed by all elements of \mathfrak{G} . The \mathfrak{G} -orbit of x , denoted $O_{\mathfrak{G}}(x)$, is the set of all the results of applying the group's transformations to x :

$$O_{\mathfrak{G}}(x) = \{ \mathfrak{g} \cdot x \mid \mathfrak{g} \in \mathfrak{G} \}.$$

We write $x \simeq x'$ if x and x' are in the same orbit (i.e. x' is just a transformed version of x). We say a function $f : X \rightarrow Y$ is \mathfrak{G} -orbit injective if:

$$f(x_1) = f(x_2) \quad \text{if and only if} \quad x_1 \simeq x_2 \quad \forall x_1, x_2 \in X.$$

This means that f assigns the **same output** to all elements in the same orbit, but **different outputs** to elements from *different orbits*. If f is \mathfrak{G} -orbit injective, it is automatically \mathfrak{G} -invariant. This is because transforming an input x by any group element $g \in \mathfrak{G}$ keeps x within its orbit.

Example: Let X be a set of points on a plane, and \mathfrak{G} represent rotations around a fixed point. The orbit $O_{\mathfrak{G}}(x)$ of a point x is the circle it traces under all rotations. A \mathfrak{G} -orbit injective function f could assign the same value (e.g., the radius) to all points in the same orbit (i.e., the same circle). Points on different circles would receive different values.

Actions of Groups on Geometric Graphs. Groups can act on geometric graphs in a way that transforms their structure while maintaining certain properties, such as the relationships between nodes or their relative positions. Below, we examine how three specific groups (the symmetric group S_n , the orthogonal group $O(d)$, and the translation group $T(d)$) act on geometric graphs.

The **symmetric group** S_n consists of all possible permutations of n elements, and it acts on the nodes of a geometric graph by permuting them. This means that the graph's nodes are reshuffled, while their relationships remain intact. Specifically, for a geometric graph \mathcal{G} with adjacency matrix A , node features S , node positions \vec{V} , and node attributes \vec{X} , the action of a permutation $\sigma \in S_n$ is expressed as:

$$P_{\sigma}\mathcal{G} := (P_{\sigma}AP_{\sigma}^T, P_{\sigma}S, P_{\sigma}\vec{V}, P_{\sigma}\vec{X}),$$

where P_{σ} is the permutation matrix corresponding to σ . This transformation alters the labeling of the nodes but preserves the graph's structure and the relationships between the nodes.

The **orthogonal group** $O(d)$, consisting of all $d \times d$ orthogonal matrices, represents rotations and reflections in d -dimensional space. In geometric graphs, $O(d)$ acts on the node positions \vec{V} and node attributes \vec{X} , preserving the Euclidean distances between points. For an orthogonal transformation $Q_g \in O(d)$, the action is given by:

$$\vec{V}Q_g \quad \text{and} \quad \vec{X}Q_g.$$

This action alters the spatial configuration of the nodes, but the geometric structure of the graph, such as the angles and distances between nodes, is preserved.

Finally, the **translation group** $T(d)$ consists of all translations in d -dimensional space, where each translation is defined by a vector $\vec{t} \in T(d)$ that shifts all points by the same amount. In the context of geometric graphs, a translation affects the node coordinates \vec{X} , shifting the positions of all nodes uniformly by adding \vec{t} to each node's position: $\vec{x}_i + \vec{t}$ for each node i . Although this transformation changes the absolute positions of the nodes, it preserves the relative distances and angles between them, ensuring that the graph's structure remains unchanged.

2 Related work

In this section, we review the key results in the literature of Geometric Neural Networks, highlighting the two main classes of geometric GNNs — *invariant* and *equivariant* models — and discuss the open questions and challenges that remain in the field.

The design space of Geometric GNNs is typically categorized into two main classes based on how they treat spatial transformations: *invariant* and *equivariant* models.

Invariant GNNs. The first class of models, \mathfrak{G} -invariant GNNs, only update scalar features by scalarization of local geometric information, i.e. converting geometric information within local neighborhoods into an invariant quantity. Thus, they transform local geometric information into invariant quantities that do not change under rotations or translations.

$$s_i^{(t+1)} := \text{UPD}\left(s_i^{(t)}, \text{AGG}\left(\left\{\left(s_i^{(t)}, \vec{v}_i^{(t)}\right), \left(s_j^{(t)}, \vec{v}_j^{(t)}\right) \mid j \in N_i\right\}\right), \vec{x}_{ij} \mid j \in N_i\right)\right)$$

For instance, **SchNet** [Schütt et al., 2018] concatenates the pairwise Euclidean distance between nodes to the message passed between them. This approach is invariant because relative distances are

unchanged under global rotations and translations. SchNet is an example of a GNN with a body order of two, meaning that it requires two nodes to compute a pairwise distance. The body order of a model refers to the number of nodes involved in computing a local invariant scalar. **DimeNet** [Gasteiger et al., 2020], on the other hand, uses both distances and angles among triplets of nodes, making it an invariant model with a body order of three. The inclusion of angles requires the use of three nodes, thus it increases the body order, but it also enables to capture more complex geometric relationships.

Equivariant GNNs. The second class of models, \mathfrak{G} -equivariant GNNs, not only update scalar features, but also propagate geometric vector features.

$$s_i^{(t+1)}, \vec{v}_i^{(t+1)} := \text{UPD}\left((s_i^{(t)}, \vec{v}_i^{(t)}), \text{AGG}\left(\left\{\{(s_i^{(t)}, \vec{v}_i^{(t)}), (s_j^{(t)}, \vec{v}_j^{(t)})\}, \vec{x}_{ij} \mid j \in N_i\}\right\}\right)\right)$$

They aim to preserve the physical symmetries of the system by ensuring that node features transform in a predictable way under Euclidean transformations. The first class of equivariant GNNs operates in Cartesian coordinates (x, y, z). For example, **PaiNN** [Schütt et al., 2021] updates both scalar and vector features by propagating geometric messages. In order to ensure equivariance and respect physical symmetries, however, certain constraints must be forced to the network’s operations, such as limiting the types of non-linearities used (e.g., no ReLU on vector features). The second class of equivariant models employs higher-order spherical tensors as node features. These models use a spherical basis, which involves features based on radius and angle, to describe the geometry. Thus, every node has an associated higher order spherical tensor as feature, and these features are updated via tensor products of neighborhood features with spherical harmonic expansion of the displacement vector. An example of this type of models are **Tensor Field Networks (TFN)** [Thomas et al., 2018].

While invariant models have proven effective in many tasks, equivariant models offer a more flexible framework for capturing complex geometric transformations. However, the trade-offs between these models in terms of expressivity and computational complexity are not sufficiently explored in the literature. Moreover, it is unclear how key design choices influence geometric GNN expressivity.

Furthermore, recent work (Dym and Maron [2020], Villar et al. [2021]) has shown that architectures such as TFN, GemNet, and GVP-GNN can serve as universal approximators of continuous, \mathfrak{G} -equivariant or \mathfrak{G} -invariant multiset functions over fully connected graphs. However, universality is a binary property — a model is either universal or not — which is often of limited practical use. More importantly, there may be varying degrees of discrimination depending on the specific classes of geometric graphs a model can or cannot distinguish, a factor that has yet to be thoroughly explored.

3 Expressivity of Geometric GNNs

3.1 How powerful are Geometric GNNs ?

GNNs struggle to distinguish all local neighborhoods when they only use unordered sets of distances and angles. Some pairs of geometric graphs cannot be differentiated by scalar features alone. To address this issue, geometric information, such as the relative orientation of local neighborhoods, is essential. However, theoretical frameworks for traditional GNNs, such as the WL test, do not apply to geometric GNNs due to physical symmetries inherent in geometric graphs. Thus, it is necessary to develop a framework to evaluate the expressivity of these models.

Geometric Graph Isomorphism. Two geometric graphs \mathcal{G} and \mathcal{H} are *geometrically isomorphic* if there exists an attributed graph isomorphism $b : V(\mathcal{G}) \rightarrow V(\mathcal{H})$, such that the geometric attributes are equivalent for all nodes up to rotations $Q_{\mathfrak{G}} \in \mathfrak{G}$ and translations $\vec{t} \in T(d)$.

$$\left(s_{b(i)}^{(\mathcal{H})}, Q_{\mathfrak{G}} \vec{v}_{b(i)}^{(\mathcal{H})}, Q_{\mathfrak{G}} (\vec{x}_{b(i)}^{(\mathcal{H})} + \vec{t})\right) = \left(s_i^{(\mathcal{G})}, \vec{v}_i^{(\mathcal{G})}, \vec{x}_i^{(\mathcal{G})}\right) \quad \text{for all } i \in V(\mathcal{G})$$

3.2 Geometric Weisfeiler-Leman (GWL) Test

We now need a test to solve the problem of determining whether two graphs are identical up to geometric transformations. We want to retain the node-centric procedure and injective aggregation from local neighbors of the WL test. However, we also need to include in the neighborhood set of invariant and equivariant geometric features, and assign a unique color for each neighborhood

type (i.e., pattern) making use of the geometric information that captures how the neighborhood type is oriented or rotated in space. Thus, the test developed must have the following properties: (1) **Orbit injectivity of colors**, i.e. if two neighborhoods are identical up to a group action \mathfrak{G} , the colors assigned to the central nodes must be the same, which means the coloring function must be orbit injective; (2) **Preservation of local geometry**, since satisfying the first property removes orientation information and makes the aggregation no longer injective, we must introduce auxiliary geometric variables g_i that have to be updated in a way that is both injective and equivariant in order to preserve spatial properties.

Algorithm. The above properties lead to the following algorithm developed by Joshi et al. [2023] for the GWL test.

First, in the initialization step, each node $i \in V$ is assigned a scalar node color $c_i \in C'$ and an auxiliary object g_i that encodes the geometric information associated with the subgraph around it. Specifically, the initial coloring is computed as:

$$c_i^{(0)} := \text{HASH}(s_i), \quad g_i^{(0)} = (c_i^{(0)}, \vec{v}_i),$$

where HASH is an injective function applied to the scalar attributes s_i of node i , and \vec{v}_i represents the geometric coordinates of node i .

In the next step, local information is aggregated. For each node i , the geometric information of its neighbors is combined into a new nested object $g_i^{(t)}$ at each iteration. This aggregation includes the node's current coloring and geometric information, along with the corresponding information from its neighbors:

$$g_i^{(t)} = \left((c_i^{(t-1)}, g_i^{(t-1)}), \{ \{ (c_j^{(t-1)}, g_j^{(t-1)}, \vec{x}_{ij}) \mid j \in N_i \} \} \right),$$

where N_i represents the neighbors of node i and \vec{x}_{ij} denotes the relative position vector for nodes i and j . Importantly, the group \mathfrak{G} can act on the geometric objects by applying its action to the geometric information contained within them. This nested aggregation is injective and equivariant and with each iteration $g_i^{(t)}$ collects geometric information from larger t -hop neighborhoods $N_i^{(t)}$ around node i .

The next step involves updating the node coloring. The node coloring $c_i^{(t)}$ is computed by applying an orbit-injective and invariant coloring function, denoted $I\text{-HASH}$, to the aggregated geometric information $g_i^{(t)}$:

$$c_i^{(t)} = I\text{-HASH}^{(t)}(g_i^{(t)}).$$

The function $I\text{-HASH}$ ensures that the coloring is invariant under the action of \mathfrak{G} ; that is, for any two geometric objects g and g' , $I\text{-HASH}(g) = I\text{-HASH}(g')$ if and only if there exists a group element $\mathfrak{g} \in \mathfrak{G}$ such that $g = \mathfrak{g} \cdot g'$.

The procedure terminates when the node partitions induced by the colors remain unchanged from the previous iteration. Given two geometric graphs \mathcal{G} and \mathcal{H} , if there exists some iteration t such that

$$\{ \{ c_i^{(t)} \mid i \in V(\mathcal{G}) \} \} \neq \{ \{ c_i^{(t)} \mid i \in V(\mathcal{H}) \} \}$$

then the GWL test distinguishes the two graphs as geometrically non-isomorphic. Otherwise, GWL cannot distinguish them.

Upper Bounding the Expressivity of Geometric GNNs. Equivariant GNNs can be at most as powerful as GWL in distinguishing non-isomorphic geometric graphs.

Theorem 1 (Joshi et al. [2023]). *Any pair of geometric graphs distinguishable by a \mathfrak{G} -equivariant GNN is also distinguishable by GWL. (Proof A.1)*

3.3 Invariant GWL (IGWL)

To better understand the role of equivariance, Joshi et al. [2023] propose a restricted version of GWL that only updates node colours using orbit injective $I\text{-HASH}$ function and does not propagate geometric information. The color update works according to this formula:

$$c_i^{(t)} := I\text{-HASH} \left((c_i^{(t-1)}, \vec{v}_i), \{ \{ (c_j^{(t-1)}, \vec{v}_j, \vec{x}_{ij}) \mid j \in \mathcal{N}_i \} \} \right)$$

Furthermore, the authors also consider $IGWL(k)$ a version of $IGWL$ that is constrained to extract information only from all the possible k -sized tuples of nodes in a neighbourhood:

$$c_i^{(t)} := \text{I-HASH}_{(k)}\left((c_i^{(t-1)}, \vec{v}_i), \{\{(c_j^{(t-1)}, \vec{v}_j, \vec{x}_{ij}) \mid j \in \mathcal{N}_i\}\}\right)$$

and $I\text{-HASH}_{(k+1)}$ is defined as:

$$\text{HASH}\left(\left\{\left\{\text{I-HASH}\left((c_i^{(t-1)}, \vec{v}_i), \{\{(c_{j_1}^{(t-1)}, \vec{v}_{j_1}, \vec{x}_{ij_1}), \dots, (c_{j_k}^{(t-1)}, \vec{v}_{j_k}, \vec{x}_{ij_k})\}\}\right) \mid j \in (\mathcal{N}_i)^k\right\}\right\}\right)$$

where $j = [j_1, \dots, j_k]$ are all the possible k -tuples formed of elements of \mathcal{N}_i .

In the same way that GWL is an upper bound for the expressivity of \mathfrak{G} -equivariant GNNs, $IGWL$ constitutes an **upper bound** on the expressivity of \mathfrak{G} -invariant GNNs.

3.4 Understanding the Design Space

The GWL framework helps analyze how design choices impact GNN expressivity, identifying the limitations of existing models and potential improvements. This is precisely the focus of [Joshi et al., 2023], which examines depth, the comparison between invariant and equivariant layers, and the body order of scalarization.

3.4.1 Role of Depth

Consider two geometric graphs, $\mathcal{G}_1 = (A_1, S_1, \vec{V}_1, \vec{X}_1)$ and $\mathcal{G}_2 = (A_2, S_2, \vec{V}_2, \vec{X}_2)$, where the underlying attributed graphs (A_1, S_1) and (A_2, S_2) are isomorphic. While isomorphism ensures that the graphs share the same connectivity structure and node attributes, their geometric features may still differ. An example arises in molecular graphs: two molecules may have the same atoms connected by identical bond types, yet exhibit different spatial arrangements due to rotations or conformational changes. To capture such differences, we define \mathcal{G}_1 and \mathcal{G}_2 as **k-hop distinct** if, for every graph isomorphism b , there exists a node $i \in V_1$ such that $b(i) \in V_2$, and the corresponding k -hop subgraphs $N_i^{(k)}$ and $N_{b(i)}^{(k)}$ are distinct. Conversely, the graphs are **k-hop identical** if, for every graph isomorphism b , all corresponding k -hop subgraphs are identical up to group actions.

It is now possible to establish the role of depth in distinguishing geometric graphs using the Geometric Weisfeiler-Lehman (GWL) algorithm:

Proposition 1 (Joshi et al. [2023]). *Within k iterations, the GWL algorithm can distinguish any k -hop distinct geometric graphs \mathcal{G}_1 and \mathcal{G}_2 , provided their underlying attributed graphs are isomorphic (Proof A.3)*

Proposition 2 (Joshi et al. [2023]). *Within k iterations, GWL cannot distinguish any k -hop identical geometric graphs \mathcal{G}_1 and \mathcal{G}_2 where the underlying attributed graphs are isomorphic. (Proof A.4)*

Furthermore, the influence of depth is also examined for Invariant GNNs using the $IGWL$ test.

Proposition 3 (Joshi et al. [2023]). *Since $IGWL$ cannot propagate geometric orientation information beyond 1-hop, no number of iterations of $IGWL$ can distinguish any 1-hop identical geometric graphs \mathcal{G}_1 and \mathcal{G}_2 where the underlying attributed graphs are isomorphic. (Proof A.5)*

3.4.2 Limitations of Invariant Message Passing

Using Propositions 1 and 3 it is possible to compare the expressive power of GWL and $IGWL$.

Theorem 2 (Joshi et al. [2023]). *Since GWL can distinguish a broader class of geometric graphs, GWL is strictly more powerful than $IGWL$. (Proof A.2)*

Indeed, equivariant message passing propagates geometric information across the computational tree, while invariant message passing does not propagate local geometry, and this restricts their expressive power. $IGWL$ and \mathfrak{G} -invariant GNNs fail to understand how various 1-hop neighborhoods in a graph are oriented with respect to each other. As a consequence, they fail to capture global geometry and

thus cannot decide numerous geometric graph properties: area, volume, and perimeter of the bounding box, distance of the center of mass, and dihedral angles. This limitation is particularly significant for applications involving large geometric graphs, such as macromolecules, where structural properties play a crucial role. From a practical perspective, this suggests that \mathcal{G} -equivariant GNNs are better suited for tasks requiring an understanding of global geometry.

3.4.3 Role of Scalarization Body Order

To analyze the impact of body order in scalarization, we can investigate the $\text{IGWL}(k)$ hierarchy, where k represents the number of nodes involved in the scalarization process, thereby defining the body order of the interactions that the algorithm can capture.

Proposition 4 (Joshi et al. [2023]). *IGWL(k) is at least as powerful as IGWL($k - 1$), and for $k \leq 5$, IGWL(k) is strictly more powerful than IGWL($k - 1$) (Proof A.6)*

This proposition implies that increasing the body order k enhances the expressivity of the IGWL test: as k increases, the algorithm becomes capable of capturing progressively more complex geometric patterns. In other words, higher-order scalarization allows the framework to discern richer geometric relationships among multiple nodes, distinguishing between more intricate spatial patterns. However, it is obvious that as k grows, also the computational complexity of the algorithm increases.

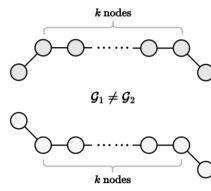
4 Experiments

4.1 Synthetic Data

To complement the theoretical results, Joshi et al. [2023] provide synthetic experiments that empirically evaluate the expressivity of geometric GNNs. More details on the setup of the experiments can be found in Appendix A.7.

4.1.1 Depth and Oversquashing

The Geometric Weisfeiler-Leman (GWL) framework theoretically guarantees perfect propagation of geometric information with each iteration. However, in practice, stacking multiple layers in geometric GNNs can distort distant geometric information, leading to potential limitations in model expressivity. Thus, the first set of experiments is aimed at testing a model’s ability to propagate geometric information non-locally. It involves **k-chains** geometric graphs, which are $(\lfloor \frac{k}{2} \rfloor + 1)$ -hop distinguishable graphs and thus require in theory only $(\lfloor \frac{k}{2} \rfloor + 1)$ GWL iterations. Various geometric GNNs are trained with an increasing number of layers to distinguish $k = 4$ -chains. The results shown in Figure 1 reveal distinct behaviors between invariant and equivariant models. Invariant GNNs fail to distinguish k-chains, underscoring their limited expressive power. In contrast, equivariant GNNs are more expressive, but often require more iterations than predicted by the GWL framework. This discrepancy suggests a potential oversquashing, where geometric information becomes increasingly compressed as it propagates through deeper layers, ultimately limiting the model’s ability to distinguish long-range dependencies. These findings align with theoretical predictions: while invariant models inherently struggle to differentiate geometric graphs, equivariant models have greater expressivity (though they may still suffer from oversquashing of geometric information when depth increases).



(k = 4-chains)		Number of layers				
GNN Layer		$\lfloor \frac{k}{2} \rfloor$	$\lfloor \frac{k}{2} \rfloor + 1 = 3$	$\lfloor \frac{k}{2} \rfloor + 2$	$\lfloor \frac{k}{2} \rfloor + 3$	$\lfloor \frac{k}{2} \rfloor + 4$
Equiv.	GWL	50%	100%	100%	100%	100%
	E-GNN	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	100.0 ± 0.0
	GVP-GNN	50.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	TFN	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	80.0 ± 24.5	85.0 ± 22.9
	MACE	50.0 ± 0.0	90.0 ± 20.0	90.0 ± 20.0	95.0 ± 15.0	95.0 ± 15.0
Inv.	IGWL	50%	50%	50%	50%	50%
	SchNet	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
	DimeNet	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
	SphereNet	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
	SchNet _{full} graph	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	SchNet _{global} feat	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0

Figure 1: [Joshi et al., 2023]. Results of \mathcal{G} -invariant and \mathcal{G} -equivariant GNNs trained to distinguish k-chain geometric graphs

4.1.2 Higher-Order Tensors and Rotational Symmetry

The GWL framework theoretically enables perfect aggregation of equivariant geometric information, capturing complex spatial interactions across layers. However, in practice, geometric GNNs face trade-offs when choosing between Cartesian and spherical coordinate systems, as well as considerations regarding the rank of the tensors used. Thus, Joshi et al. [2023] decided to test the ability of the \mathcal{G} -equivariant layers to identify the orientation of the neighborhood and distinguish the orientation of structures with rotational symmetry. To this aim they trained single layer \mathcal{G} -equivariant GNNs to distinguish two *distinct* rotated versions of each L -fold symmetric structures. L -fold symmetric structures do not change when rotated by an angle $\frac{2\pi}{L}$ a point or an axis.

The results shown in Figure 2 show that a layer using order- L tensors cannot distinguish between different orientations of structures with rotational symmetry greater than L -fold. This limitation is especially pronounced in models based on first-order tensors (Cartesian vectors), which lack the flexibility to handle complex rotational symmetries. This highlights the need for higher-order representations to fully capture geometric information in rotationally symmetric settings.

GNN Layer		Rotational symmetry			
		2 fold	3 fold	5 fold	10 fold
Cart.	E-GNN $_{L=1}$	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	GVP-GNN $_{L=1}$	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
Spherical	TFN/MACE $_{L=1}$	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	TFN/MACE $_{L=2}$	100.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	TFN/MACE $_{L=3}$	100.0 \pm 0.0	100.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	TFN/MACE $_{L=5}$	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	50.0 \pm 0.0
	TFN/MACE $_{L=10}$	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0

Figure 2: [Joshi et al., 2023]. Results of single layer \mathcal{G} -equivariant GNNs in distinguishing two distinct rotated versions of L -fold symmetric structures

4.1.3 Scalarisation Body Order

Finally, the authors try to assess the ability of geometric GNNs to generate distinguishing fingerprints for local neighborhoods. Their test involves assessing whether a single-layer geometric GNN can differentiate between counterexample structures that remain indistinguishable under k -body scalarization. The results shown in Figure 3 highlight a fundamental limitation: geometric GNN layers with body order k do not distinguish the corresponding counterexample structures, indicating that higher-order interactions are necessary for capturing certain geometric patterns. This further suggests that increasing body order enhances expressivity.

GNN Layer		Counterexample from Pozdnyakov et al. (2020)		
		2-body	3-body (Fig.1(b))	4-body (Fig.2(f))
Inv.	SchNet $_{2\text{-body}}$	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	DimeNet $_{3\text{-body}}$	100.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	SphereNet $_{4\text{-body}}$	100.0 \pm 0.0	100.0 \pm 0.0	50.0 \pm 0.0
$O(3)$ -Equiv.	E-GNN $_{2\text{-body}}$	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	GVP-GNN $_{3\text{-body}}$	100.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	TFN $_{2\text{-body}}$	50.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	MACE $_{3\text{-body}}$	100.0 \pm 0.0	50.0 \pm 0.0	50.0 \pm 0.0
	MACE $_{4\text{-body}}$	100.0 \pm 0.0	100.0 \pm 0.0	50.0 \pm 0.0
	MACE $_{5\text{-body}}$	100.0 \pm 0.0	100.0 \pm 0.0	100.0 \pm 0.0

Figure 3: [Joshi et al., 2023]. Results of single layer geometric GNNs in distinguishing counterexample structures that are indistinguishable using k -body scalarisation

4.2 Real Data

This experiment evaluates the performance of Geometric Graph Neural Networks (GNNs) in predicting the electric dipole moment of drug-like molecules, using the QM9 dataset. Several GNN architectures are tested, ranging from a basic MPNN that ignores 3D coordinates to more advanced

models incorporating geometric symmetries such as rotations and translations. For more details on experimental setup see Appendix A.8.

The results are shown in Figure 4. The FinalMPNNModel, which ensures equivariance to 3D symmetries, consistently outperforms the other models, achieving the lowest Test MAE. The InvariantMPNNModel also performs well, demonstrating the importance of incorporating geometric information in a principled manner. The CoordMPNNModel performs poorly, highlighting the limitations of naively incorporating 3D coordinates without ensuring invariance or equivariance to rotations and translations. The Vanilla MPNNModel, GCNModel and GATModel also perform worse, underlining the importance of geometric information as well as the need of a correct way to incorporate them.

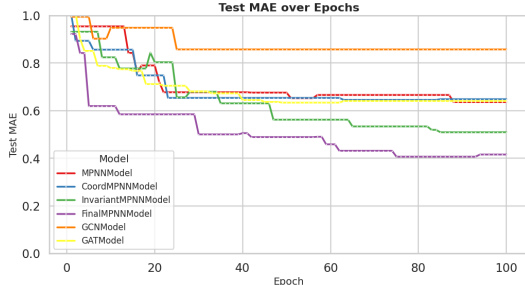


Figure 4: Test MAE for Different GNNs on QM9 Dataset

5 Conclusion

The Geometric Weisfeiler-Leman (GWL) framework was introduced to assess the expressive power of geometric GNNs, filling a crucial gap as traditional GNNs and their theoretical tools are inadequate for geometric graphs. In addition to this new framework, the contributions of Joshi et al. [2023] are twofold: (1) They highlight how key design choices influence the expressivity of geometric GNNs using the GWL framework; (2) They present synthetic experiments that uncover challenges in developing expressive geometric models.

A key takeaway from this work is a deeper understanding of the **geometric GNN design space**. Specifically: (1) **Invariant layers** exhibit limited expressivity, as they cannot distinguish 1-hop identical geometric graphs; (2) **Equivariant layers**, on the other hand, can distinguish larger classes of graphs and propagate geometric information beyond local neighborhoods, thus enhancing model performance; (3) The utility of **higher-order tensors** and **scalarization** is emphasized as crucial for building more powerful GNNs capable of capturing complex geometric interactions.

Future Directions. Given the Geometric Weisfeiler-Leman test, it is now necessary to develop models that have injective and \mathfrak{G} -equivariant aggregation functions and update scalar quantities with \mathfrak{G} -invariant functions and vector quantities with \mathfrak{G} -equivariant functions, which remains challenging in practice. Future work should aim to develop practical and powerful geometric GNNs based on these insights. Some steps have already been made in this direction, with several papers (Amir et al. [2024], Du et al. [2024], Wang et al. [2024], Li et al. [2024]) leveraging these insights and results to present new geometric deep learning models.

References

- Tal Amir et al. Neural injective functions for multisets, measures and graphs via a finite witness theorem. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.
- Minjoon Baek, Frank DiMaio, Igor Anishchenko, Jurgis Dauparas, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 2021.
- Sebastian Batzner, Armen Musaelian, Lijun Sun, Michael Geiger, Jason P. Mailoa, Michael Kornbluth, Nicholas Molinari, T. Edward Smidt, and Boris E. Kozinsky. (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 2022.

- Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, et al. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 2021.
- Yuanqi Du et al. A new perspective on building efficient and expressive 3d equivariant graph neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.
- N. Dym and H. Maron. On the universality of rotation equivariant point cloud networks. In *International Conference on Learning Representations (ICLR)*, 2020.
- J. Gastegger, J. Groß, and S. Günnemann. Directional message passing for molecular graphs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. URL <https://openreview.net/forum?id=Hylw8aHtPr>.
- Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
- Amir R. Jamasb, Roberto Vicente Torné, Ethan J. Ma, Yang Du, Christopher Harris, Kangning Huang, David Hall, Pietro Liò, and Tom L. Blundell. Graphein: A python library for geometric deep learning and network analysis on biomolecular structures and interaction networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- Chaitanya K. Joshi, Cristian Bodnar, Simon V. Mathis, Taco Cohen, and Pietro Liò. On the expressive power of geometric graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning, ICML’23*. JMLR.org, 2023.
- Qi Li, Fabio Gama, Alberto Ribeiro, and Achim Prorok. Graph neural networks for decentralized multi-robot path planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- Zian Li et al. Is distance matrix enough for geometric deep learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2024.
- Sergei N. Pozdnyakov, Matthew J. Willatt, Albert P. Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Incompleteness of atomic structure representations. *Physical Review Letters*, 125(16):166001, 2020. doi: 10.1103/PhysRevLett.125.166001.
- Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.
- K. Schütt, O. Unke, and M. Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021. URL <https://proceedings.mlr.press/v139/schutt21a.html>.
- K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. Schnet—a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018. doi: 10.1063/1.5019779.
- N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018. URL <https://arxiv.org/abs/1802.08219>.
- S. Villar, D. W. Hogg, K. Storey-Fisher, W. Yao, and B. Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. In *NeurIPS*, 2021.
- Yusong Wang et al. Enhancing geometric representations for molecules with equivariant vector-scalar interactive message passing. *Nature*, 2024.
- Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 1968.

A Appendix

We acknowledge the use of AI tools for assistance in improving syntax, grammar and clarity of the writing of this report. Prompts such as "Improve the following text for clarity and grammar" were used to refine the text.

A.1 Proof of Theorem 1

Proof. Proof by contradiction. Consider two geometric graphs \mathcal{G} and \mathcal{H} . Then, **Theorem 1** implies that if a GNN output is $f(\mathcal{G}) \neq f(\mathcal{H})$, then also the GWL test will always determine that \mathcal{G} and \mathcal{H} are non-isomorphic, i.e. $\mathcal{G} \neq \mathcal{H}$. Assume now the opposite and suppose that after T iterations the GWL cannot distinguish \mathcal{G} and \mathcal{H} (which means they must have the same colours for all the iterations from zero until T), yet the GNN output is $f(\mathcal{G}) \neq f(\mathcal{H})$. The proof works by now showing that on the same graph for nodes i and k , if $(c_i^{(t)}, g_i^{(t)}) = (c_k^{(t)}, g_k^{(t)})$, we will always have GNN features such that $(s_i^{(t)}, \vec{v}_i^{(t)}) = (s_k^{(t)}, Q_{\mathfrak{g}} \vec{v}_k^{(t)})$ for any iteration t . This holds for $t = 0$ because both GWL and the GNN start with the same initial node features and geometric embeddings.

Inductive Step. Suppose now this holds also for iteration t :

$$(c_i^{(t)}, g_i^{(t)}) = (c_k^{(t)}, g_k^{(t)}) \implies (s_i^{(t)}, \vec{v}_i^{(t)}) = (s_k^{(t)}, Q_{\mathfrak{g}} \vec{v}_k^{(t)}).$$

Then at iteration $t + 1$ we will have:

$$(c_i^{(t+1)}, g_i^{(t+1)}) = (c_k^{(t+1)}, g_k^{(t+1)}),$$

then:

$$\left\{ (c_i^{(t)}, g_i^{(t)}), \{ (c_j^{(t)}, g_j^{(t)}) \mid j \in \mathcal{N}_i \} \right\} = \left\{ (c_k^{(t)}, g_k^{(t)}), \{ (c_j^{(t)}, g_j^{(t)}) \mid j \in \mathcal{N}_k \} \right\}$$

Since the GNN applies the same aggregate and update operations at every node, the same inputs are mapped to the same outputs. Thus, by our assumption on iteration t :

$$(s_i^{(t+1)}, \vec{v}_i^{(t+1)}) = (s_k^{(t+1)}, Q_{\mathfrak{g}} \vec{v}_k^{(t+1)}).$$

By induction, this holds for all t .

If \mathcal{G} and \mathcal{H} have the same collection of node colors and geometric multisets for all iterations t , then the GNN produces the same collection of node features:

$$\left\{ (s_i^{(t)}, \vec{v}_i^{(t)}), \{ (s_j^{(t)}, \vec{v}_j^{(t)}) \mid j \in \mathcal{N}_i \} \right\}$$

Thus, the graph-level readout of the GNN outputs $f(\mathcal{G}) = f(\mathcal{H})$, contradicting the assumption. \square

A.2 Proof of Theorem 2

Proof. First, we show that GWL is at least as powerful as IGWL. To do so it suffices to see that IGWL does not update g_i over iterations for all $i \in V$. Thus, GWL can behave the same way as IGWL if it simply does not update g_i , i.e. $g_i^{(t)} = g_i^{(t-1)} = g_i^{(0)} = (s_i, \vec{v}_i)$. This means the GWL class contains IGWL. Next, we show that GWL is strictly more powerful than IGWL. To do this we must show that GWL can distinguish some graphs that IGWL cannot. Let's consider k -hop distinct graphs (with $k > 1$). Then, Proposition 1 states that GWL can distinguish them, while Proposition 3 states that IGWL cannot. \square

A.3 Proof of Proposition 1

Proof. In the k -th iteration, the GWL test identifies the \mathfrak{G} -orbit of the k -hop subgraph $\mathcal{N}_i^{(k)}$ for each node i , which corresponds to the geometric multiset $g_i^{(k)}$.

If \mathcal{G}_1 and \mathcal{G}_2 are k -hop distinct, there exists a bijection $b : V_1 \rightarrow V_2$ and some node $i \in V_1, b(i) \in V_2$ such that:

$$\mathcal{N}_i^{(k)} \neq \mathcal{N}_{b(i)}^{(k)}$$

Thus, it must be that the \mathfrak{G} -orbits of the geometric multisets $g_i^{(k)}$ and $g_{b(i)}^{(k)}$ are mutually exclusive:

$$O_{\mathfrak{G}}(g_i^{(k)}) \cap O_{\mathfrak{G}}(g_{b(i)}^{(k)}) = \emptyset \implies c_i^{(k)} \neq c_{b(i)}^{(k)}$$

This indicates that their geometric multisets remain distinct under all transformations in \mathfrak{G} . Thus, GWL can successfully differentiate \mathcal{G}_1 and \mathcal{G}_2 within k iterations. \square

A.4 Proof of Proposition 2

Proof. Let \mathcal{G}_1 and \mathcal{G}_2 be two k -hop identical graphs, where the underlying attributed graphs are isomorphic. This means that for every bijection $b : V_1 \rightarrow V_2$, and for each node $i \in V_1$, the corresponding k -hop subgraphs in both graphs are identical, i.e.,

$$\mathcal{N}_i^{(k)} = \mathcal{N}_{b(i)}^{(k)}.$$

In the k -th iteration of the GWL test, the algorithm identifies the \mathfrak{G} -orbit of the k -hop subgraph $\mathcal{N}_i^{(k)}$ for each node i , which corresponds to the geometric multiset $g_i^{(k)}$.

The action of the group \mathfrak{G} on these geometric multisets results in the same orbits for the corresponding subgraphs. Therefore, the orbits of the geometric multisets $g_i^{(k)}$ and $g_{b(i)}^{(k)}$ are identical:

$$O_{\mathfrak{G}}(g_i^{(k)}) = O_{\mathfrak{G}}(g_{b(i)}^{(k)}) \implies c_i^{(k)} = c_{b(i)}^{(k)}$$

Since the geometric multisets of the corresponding k -hop subgraphs are identical under the action of \mathfrak{G} , the GWL test cannot distinguish between \mathcal{G}_1 and \mathcal{G}_2 within k iterations.

Thus, the GWL test cannot distinguish any k -hop identical geometric graphs \mathcal{G}_1 and \mathcal{G}_2 within k iterations. \square

A.5 Proof of Proposition 3

Proof. The IGWL algorithm, at each iteration, identifies the \mathfrak{G} -orbit of the 1-hop local neighborhood $\mathcal{N}_i^{(k=1)}$ for every node i . However, IGWL is inherently limited to 1-hop neighborhoods because it does not propagate geometric information beyond this local scope.

Now, consider two graphs \mathcal{G}_1 and \mathcal{G}_2 that are 1-hop identical. This means that for any bijection b between the nodes of \mathcal{G}_1 and \mathcal{G}_2 , and for every node $i \in V_1$ with $b(i) \in V_2$, the corresponding 1-hop neighborhoods $\mathcal{N}_i^{(k)}$ and $\mathcal{N}_{b(i)}^{(k)}$ are identical up to group actions. As a result, the \mathfrak{G} -orbits of the geometric multisets $g_i^{(1)}$ and $g_{b(i)}^{(1)}$ coincide, i.e.,

$$O_{\mathfrak{G}}(g_i^{(1)}) = O_{\mathfrak{G}}(g_{b(i)}^{(1)}).$$

This implies that the colors assigned by IGWL, $c_i^{(k)}$ and $c_{b(i)}^{(k)}$, will also be identical. Consequently, no matter how many iterations of IGWL are performed, the algorithm will never be able to distinguish between \mathcal{G}_1 and \mathcal{G}_2 . \square

A.6 Proof of Proposition 4

Proof. By definition, **I-HASH**(k) computes \mathfrak{G} -invariant scalar values for all possible tuples of up to k nodes, which are selected from the neighborhood along with the central node. Consequently, the function class **I-HASH**(k) contains **I-HASH**($k - 1$). This directly implies that the associated test **IGWL**(k) is at least as powerful as **IGWL**($k - 1$).

Next, to establish that **IGWL**(k) is strictly more expressive than **IGWL**($k - 1$) for $k \leq 5$, it is sufficient to demonstrate the existence of geometric neighborhoods that **IGWL**(k) can differentiate but **IGWL**($k - 1$) cannot:

- **For $k = 3$ and $\mathcal{G} = O(3)$ or $SO(3)$** , the local neighborhood depicted in Figure 1 of [Schütt et al., 2021] contains two distinct configurations differing in the angles between neighboring nodes. These can be distinguished by **IGWL(3)** but not by **IGWL(2)**.
- **For $k = 4$ and $G = O(3)$ or $SO(3)$** , the two geometric neighborhoods illustrated in Figure 1 of [Pozdnyakov et al., 2020] can be differentiated by **IGWL(4)** but not by **IGWL(3)**.
- **For $k = 5$ and $G = O(3)$** , the distinct local neighborhoods shown in Figure 2(e) of [Pozdnyakov et al., 2020] are distinguishable using **IGWL(5)** but not by **IGWL(4)**.
- **For $k = 5$ and $G = SO(3)$** , the geometric neighborhoods from Figure 2(f) in [Pozdnyakov et al., 2020] can be separated by **IGWL(5)** but not by **IGWL(4)**.

□

A.7 Synthetic Experiments Details

Architectures

- **G-invariant GNNs**: SchNet, DimeNet, SphereNet
- **G-equivariant GNNs (Cartesian vectors)**: E-GNN, GVP-GNN
- **G-equivariant GNNs (Spherical tensors)**: TFN, MACE

Tasks

Synthetic experiments test key challenges in geometric GNNs:

- **Distinguishing k-chains**: Evaluates non-local geometric propagation and oversquashing.
- **Rotationally symmetric structures**: Assesses the ability to identify neighborhood orientation.
- **Pozdnyakov et al. [2020] counterexamples**: Tests local fingerprinting and the role of higher-body order scalarization.

Hyperparameters

- **Implementations**: SchNet and DimeNet from PyTorch Geometric (PyG), others adapted from original authors.
- **Feature Channels**: 128 for SchNet, DimeNet, SphereNet, E-GNN; 64 for GVP-GNN, TFN, MACE.
- **Tensor Orders**: TFN and MACE use $L = 2$ tensors; MACE defaults to body order 4.
- **Training**: 100 epochs with Adam optimizer, initial learning rate 10^{-4} , reduced by a factor of 0.9 after 25 epochs of plateauing. Results averaged over 10 random seeds.

A.8 Real Data Experiments Details

We also decided to apply Geometric GNNs to real data to test if in practical cases they outperform standard GNNs. In particular, the experiment focuses on the application of Geometric Graph Neural Networks to the task of molecular property prediction, specifically predicting the electric dipole moment of drug-like molecules. The experiment is designed to explore the impact of incorporating 3D geometric information (such as atom positions) into GNNs and how different approaches to handling geometric symmetries affect model performance.

Dataset The QM9 dataset is a widely used benchmark dataset in molecular machine learning. It consists of approximately 130,000 small organic molecules, each with 19 regression targets representing various quantum mechanical properties. For this experiment, the focus is on predicting the electric dipole moment, a measure of the molecule’s polarity. The dataset is split into training, validation, and test sets, with 1,000 molecules each for training, validation, and testing in this experiment.

Models The experiment explores several GNN architectures: (1) Vanilla Message Passing Neural Network (MPNN) that ignores 3D coordinates and only uses node and edge features; (2) CoordMPNNModel, that extends the vanilla MPNN by incorporating 3D atom coordinates into the node features without ensuring invariance to rotations and translations; (3) Graph Convolutional Network (GCN) that takes into account also the geometric information on atom position; (4) Graph Attention Network (GAT) that takes into account also the geometric information on atom position; (5) InvariantMPNNModel, that leverages geometric information while maintaining invariance to 3D symmetries; (6) FinalMPNNModel, that ensures equivariance to 3D rotations and translations.

Experimental Setup Each model is trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. The Mean Squared Error (MSE) is used to measure the difference between predicted and actual dipole moments. Models are evaluated on both the validation and test sets using Mean Absolute Error (MAE).