Universally Invariant Learning in Equivariant GNNs

Jiacheng Cen[1 2 3], **Anyi Li**[1 2 3], **Ning Lin**[1 2 3], **Tingyang Xu**[4 5], **Yu Rong**[4 5]
Deli Zhao[4 5], **Zihe Wang**[1 2 3], **Wenbing Huang**[1 2 3*]
[1] Gaoling School of Artificial Intelligence, Renmin University of China
[2] Beijing Key Laboratory of Research on Large Models and Intelligent Governance
[3] Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE
[4] DAMO Academy, Alibaba Group, Hangzhou, China [5] Hupan Lab, Hangzhou, China
{jiacc.cn, li_anyi, ninglin00}@outlook.com; {xuty_007, yu.rong}@hotmail.com;
zhaodeli@gmail.com; wang.zihe@ruc.edu.cn; hwenbing@126.com

Abstract

Equivariant Graph Neural Networks (GNNs) have demonstrated significant success across various applications. To achieve completeness—that is, the universal approximation property over the space of equivariant functions—the network must effectively capture the intricate multi-body interactions among different nodes. Prior methods attain this via deeper architectures, augmented body orders, or increased degrees of steerable features, often at high computational cost and without polynomial-time solutions. In this work, we present a theoretically grounded framework called Uni-EGNN for constructing complete equivariant GNNs that is both efficient and practical. We prove that a complete equivariant GNN can be achieved through two key components: 1) a complete scalar function, referred to as the canonical form of the geometric graph; and 2) a full-rank steerable basis set. Leveraging this finding, we propose an efficient algorithm for constructing complete equivariant GNNs based on two common models: EGNN and TFN. Empirical results demonstrate that our model demonstrates superior completeness and excellent performance with only a few layers, thereby significantly reducing computational overhead while maintaining strong practical efficacy.

1 Introduction

Various types of scientific data, including chemical molecules, proteins, and other particle-based physical systems, are often represented as *geometric graphs* [1; 2]. This data structure not only captures node characteristics and edge information but also includes a 3D vector (such as position, velocity, etc.) for each node. To effectively process geometric graphs, equivariant Graph Neural Networks (GNNs) have been developed that enable equivariant message passing over nodes while adhering to the $\mathrm{E}(3)$ or $\mathrm{SE}(3)$ symmetries inherent in physical laws. These models have achieved significant success in various scientific tasks, including physical dynamics simulation, molecular property prediction, and protein design [3–30].

In current literature, a key objective in the design of equivariant GNNs is to achieve completeness, which refers to the universal approximation property [31]. This concept was initially explored by [32], who proved the universality of high-degree steerable models, specifically TFN [33]. Subsequent models, such as SEGNN [34] and MACE [35], can be similarly confirmed to be complete by establishing their relations with TFN. Regrettably, the theory proposed by [32] applies only to point clouds (i.e., fully connected geometric graphs) and relies on sufficiently large values of the layer number, body order, and feature degree.

*Wenbing Huang is the corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

More recently, the introduction of the GWL-test [36] and further researches [37–40] have provided a new perspective by connecting the completeness of invariant models to the geometric isomorphism problem through an extension of the Weisfeiler-Lehman (WL) test [41]. Unfortunately, the GWL-test has only quasi-polynomial solutions [42], inheriting the same weakness as the original WL-test.

In this paper, we explore complete equivariant GNNs in a more effective and operable way. To begin with, we reformulate existing equivariant GNNs as an expansion of multi-body high-degree bases coupled with the Clebsch–Gordan (CG) tensor product (Eqs. (2) and (3)). This reformulation allows us to clearly identify the key limitations of current methods: they fail to achieve complete expansion when the degree and body order of CG tensor products are constrained, or incur prohibitively high computational costs otherwise. To overcome these weaknesses, we propose a novel expansion of equivariant GNNs (Eq. (4)), which does not necessarily require CG tensor product: the sum over a finite basis set with dynamic weights dependent on the input. The completeness is achieved if the weights (called the scalar function) are complete and the basis set is full-rank.

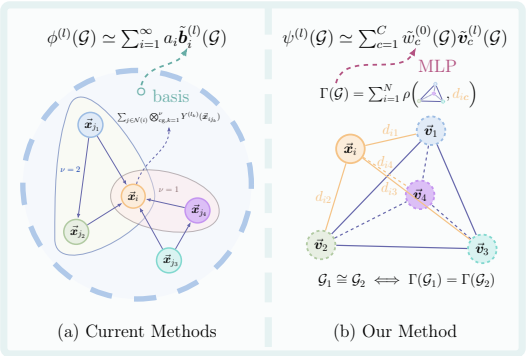


Figure 1: Difference between current methods and our Method. (a) Current methods can be understood as an expansion of multi-body high-degree bases $\tilde{\boldsymbol{b}}_i^{(l)}$ coupled with the Clebsch–Gordan (CG) tensor product (Eq. (3)). The completeness is achieved with sufficiently large values of the body order and feature degree. (b) In contrast, our method proposes a novel expansion which does not necessarily require CG tensor product (Eq. (4)): the sum over a finite basis set with dynamic weights dependent on the input. The completeness is ensured if the weights (the scalar function $\Gamma(\mathcal{G})$) are complete and the basis set $\tilde{\boldsymbol{V}}^{(l)}$ is full-rank.

Interestingly, constructing a complete scalar function is equivalent to solving the geometric isomorphism problem, which, unlike the traditional isomorphism task on topological graphs, can be resolved in polynomial time [42]. Actually, we have presented an algorithm operating with a complexity of $\mathcal{O}(N^6)$ where N is the number of nodes, based on the four-point positioning principle, and further turned it into *canonical form* for comprehensive embeddings of geometric graphs. Additionally, we theoretically prove that a full-rank basis set of any degree can always be constructed, if the input geometric graph is asymmetric. Building on this foundation, we propose a more efficient algorithm with a complexity of $\mathcal{O}(N^2)$ to construct the canonical form specifically for asymmetric graphs.

Thanks to our theoretical findings, we modify existing equivariant GNNs (such as EGNN [43] and TFN [33]) into complete models. Specifically, we introduce two complete implementations of EGNN, termed EGNN/TFN$_{\text{cpl}}$-global and EGNN/TFN$_{\text{cpl}}$-local. We conduct seven sets of experiments, validating our theoretical results and demonstrating the superior performance of our method compared to previous models. Our model achieves great performance with few layers, thereby significantly reducing computational overhead while maintaining strong practical efficacy[2].

2 Related Work

Equivariant GNNs. According to the operators used to build the model, equivariant GNNs can be categorized into two main classes: scalarization-based models and tensor-product-based models. Scalarization-based models (*e.g.*, EGNN [43] and PaiNN [44]) utilize scalars (*e.g.* distance and angles) as coefficients to linearly combine 3D vectors during node updates. In contrast, tensor-product-based models (*e.g.*, TFN [33] and MACE [35]) utilize spherical harmonics to maintain the equivariance of message passing and realize interactions between steerable features of different degrees through CG tensor products. Historically, researchers maintained that tensor-product-based models could provide richer information, despite their significantly higher computational resource demands [45–47]. However, the recent success of spherical-scalarization models such as SO3KRATES [48], HEGNN [49], and GotenNet [50] has prompted a reevaluation of this assumption, suggesting a

[2]Code is available at `https://github.com/GLAD-RUC/Uni-EGNN`.

potential shift towards scalar learning. Our research provides an important conclusion in this direction: a complete lth-degree steerable model can be constructed using two components—a complete scalar model and a full-rank basis set of lth-degree steerable features.

Completeness of Equivariant GNNs. Completeness in equivariant GNNs refers to their ability to approximate any continuous function with arbitrary precision on geometric graphs, thus sparking diverse research avenues. Initially, [32] demonstrated the universality of tensor-product-based models, notably TFN [33], in fully connected geometric graphs. Subsequently, building on the work of [51], later studies began examining the completeness of scalar models. For instance, [52] established the completeness of models such as DimeNet [53], GemNet [54], and SphereNet [55] under $\mathcal{A}$-unsymmetry conditions. Meanwhile, frame-based models [56–59] emerged, innovatively separating geometric structures into group operations and orbits for separate processing. However, this approach can disrupt permutation invariance in symmetric graphs. Furthermore, the GWL test framework, inspired by the WL-test on topological graphs, offers an upper bound on the expressive power of equivariant GNNs over sparse graphs, thereby influencing further researches [37–40]. Nevertheless, it is worth noting that many of these methods rely on message passing and require multiple iterations, akin to multi-layer networks. Our method utilizes the canonical form to derive a complete scalar function, and when integrated with a basis set, facilitates the construction of a fully equivariant neural network that can be effectively achieved with just a single-layer architecture.

3 Method

In this section, we first introduce necessary preliminaries in § 3.1. Then in § 3.2, we review existing equivariant GNNs from the perspective of basis construction. Subsequently, we present our framework in § 3.3, which proposes to construct complete equivariant GNNs from the perspective based on output space. The two components to achieve completeness, including a canonical form of geometric graphs and a full-rank steerable basis set, are presented in § 3.4. Finally in § 3.5, we demonstrate how to modify typical equivariant GNNs to be complete based on our theoretical findings. Comprehensive theoretical analysis and proofs are provided in Appendix A.

3.1 Preliminaries

Geometric graph. A geometric graph of N nodes is defined as $\mathcal{G} \coloneqq (\boldsymbol{H}, \vec{\boldsymbol{X}}; \boldsymbol{A})$, where $\boldsymbol{H} \coloneqq \{\boldsymbol{h}_i \in \mathbb{R}^{C_H}\}_{i=1}^N$ and $\vec{\boldsymbol{X}} \coloneqq \{\vec{\boldsymbol{x}}_i \in \mathbb{R}^3\}_{i=1}^N$ are node features and 3D coordinates, respectively; $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ is the adjacency matrix representing topological connections between different nodes and each edge could be assigned with an edge feature $\boldsymbol{e}_{ij}$ if necessary.

Equivariance. Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output vector spaces, respectively. A function $\phi : \mathbb{X} \to \mathbb{Y}$ is called *equivariant* with respect to group $\mathfrak{G}$ if

$$\forall \mathfrak{g} \in \mathfrak{G}, \phi(\rho_{\mathbb{X}}(\mathfrak{g})\boldsymbol{x}) = \rho_{\mathbb{Y}}(\mathfrak{g})\phi(\boldsymbol{x}), \tag{1}$$

where $\rho_{\mathbb{X}}$ and $\rho_{\mathbb{Y}}$ are the group representations in the input and output spaces, respectively. Since translation invariance can always be achieved by translating the center of all coordinates to the origin, we omit translation and focus solely on equivariance concerning $\mathrm{O}(3)$, the group consisting of rotation and inversion. This implies $\mathrm{O}(3) = \mathrm{SO}(3) \times C_i$, where $\mathrm{SO}(3)$ represents the rotation group, and $C_i = \{\mathfrak{e}, \mathfrak{i}\}$ denotes the inversion group, with $\mathfrak{e}$ as the identity and $\mathfrak{i}$ as the inversion. We define two additional groups: the Euclidean group $\mathrm{E}(3)$ that further involves translation into $\mathrm{O}(3)$, and permutation group S_N acting on a set of N elements.

Irreducible representations and steerable features. For a group element $\mathfrak{r} \in \mathrm{SO}(3)$, its representation is typically achieved through irreducible representations such as the Wigner-D matrix $\boldsymbol{D}^{(l)}(\mathfrak{r}) \in \mathbb{R}^{(2l+1)\times(2l+1)}$, where l denotes the degree. We call features $\tilde{\boldsymbol{v}}^{(l)}$ as lth-degree steerable features if they are transformable using $\boldsymbol{D}^{(l)}$. The most common steerable features can be obtained via spherical harmonics, expressed as $Y^{(l)}(\vec{\boldsymbol{x}}/\|\vec{\boldsymbol{x}}\|) = [Y_m^{(l)}(\vec{\boldsymbol{x}}/\|\vec{\boldsymbol{x}}\|)]_{m=-l}^{l}$, which satisfy the property $Y^{(l)}(\boldsymbol{R}_{\mathfrak{r}}\vec{\boldsymbol{x}}/\|\boldsymbol{R}_{\mathfrak{r}}\vec{\boldsymbol{x}}\|) = \boldsymbol{D}^{(l)}(\mathfrak{r})Y^{(l)}(\vec{\boldsymbol{x}}/\|\vec{\boldsymbol{x}}\|)$. According to [60], spherical harmonics provide a *complete* basis set for $\mathrm{SO}(3)$-equivariant functions on the sphere. Clebsch–Gordan (CG) tensor products are usually leveraged to facilitate interactions between steerable features of differing degrees. Specifically, given features $\tilde{\boldsymbol{v}}^{(l_1)}$ and $\tilde{\boldsymbol{v}}^{(l_2)}$, a new feature $\tilde{\boldsymbol{v}}^{(l)}$ can be generated, for $|l_1 - l_2| \le l \le l_1 + l_2$. This calculation is abbreviated as $\tilde{\boldsymbol{v}}^{(l)} = \tilde{\boldsymbol{v}}^{(l_1)} \otimes_{\mathrm{cg}} \tilde{\boldsymbol{v}}^{(l_2)}$. By including

parity [61], the above concepts can be naturally extended to the orthogonal group $\mathrm{O}(3)$. A more detailed discussion is available in Appendix A.1.

3.2 Rethinking equivariant GNNs: a perspective based on basis construction

By definition, a complete equivariant model is able to express any equivariant function. For instance, spherical harmonics provide complete bases for $\mathrm{SO}(3)$-equivariant functions on the sphere: $\phi(\vec{\boldsymbol{x}}/\|\vec{\boldsymbol{x}}\|)$, enabling the representation of all such functions. In 3D space beyond the sphere, a complete function is achieved by combining spherical harmonics with a learnable radial basis function $\varphi(\|\vec{\boldsymbol{x}}\|)$. However, deriving complete equivariant functions on geometric graphs, namely $\tilde{\boldsymbol{\phi}}^{(l)}(\mathcal{G})$,[3] is more complex due to the need to consider multi-body interactions and permutation symmetry.

To begin with, let us define a function set $\mathbb{F}^{(l)}$, which encompasses all square-integrable lth-degree steerable functions on geometric graphs. Let $\mathbb{B}^{(l)} \subset \mathbb{F}^{(l)}$ denote the basis set that expands the function space of equivariant GNNs. The learning process of equivariant GNNs becomes an approximation:

$$\tilde{\boldsymbol{\phi}}^{(l)}(\mathcal{G}) \approx \textstyle\sum_{\tilde{\boldsymbol{b}}_i^{(l)} \in \mathbb{B}^{(l)}} a_i \tilde{\boldsymbol{b}}_i^{(l)}(\mathcal{G}), \tag{2}$$

where $a_i \in \mathbb{R}$ are weight coefficients independent of the input. Enhancing the expressiveness of equivariant GNNs involves enlarging $\mathbb{B}^{(l)}$ until it coincides with $\mathbb{F}^{(l)}$. When $\mathrm{span}(\mathbb{B}^{(l)}) = \mathbb{F}^{(l)}$, the model is termed a *complete* lth-degree steerable model, or simply a *complete* model, and the basis set is denoted as $\mathbb{B}^{(l)}_{\mathrm{cpl}}$. Specifically, complete 0th-degree steerable models are referred to as *complete scalar models*.

Below, we demonstrate that standard equivariant GNNs share a common basis set construction, differing primarily in their body-order formulation. For simplicity and without loss of generality, we assume uniform node and edge features that can either be treated as identical or safely ignored. This abstraction allows us to focus purely on the geometric and topological aspects of the problem while maintaining the core theoretical insights.

Example 3.1 (Basis Set of Common Equivariant GNNs)**.** The basis set for common equivariant GNNs, such as EGNN [43], HEGNN [49], TFN [33], and MACE [35], can be unified into the following form:

$$\mathbb{B}^{(l)}_{\nu} = \{\textstyle\sum_i \sum_{\boldsymbol{j} \in \mathcal{N}(i)} \bigotimes_{\mathrm{cg},k=1}^{\nu} Y^{(l_k)}(\vec{\boldsymbol{x}}_{ij_k}/\|\vec{\boldsymbol{x}}_{ij_k}\|)\}, \tag{3}$$

where $\nu \geq 1$ denotes the body order, $\boldsymbol{j} = (j_1, \ldots, j_\nu)$ represents all chosen ordered neighbors, and $\vec{\boldsymbol{x}}_{ij_k} \coloneqq \vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_{j_k}$. When only single-body neighbor is considered (*i.e.*, $\nu = 1$), Eq. (3) forms the basis set for EGNN [43] and HEGNN [49], applicable to degrees $l = 1$ and $l \geq 1$, respectively. In contrast, for multi-body interactions, Eq. (3) corresponds to TFN [33] when the body order $\nu = 1$, or MACE [35] with higher body orders $\nu \geq 1$.

Although these models adhere to the form of Eq. (3), their expressive power differs markedly. Multi-body models can create entirely new bases through tensor products and achieve completeness by either utilizing a sufficiently high body order [62] or stacking multiple layers [32], whereas single-body steerable models could not. However, multi-body models suffer from notable limitations: a) They may not encompass steerable features of all degrees, if input degrees are improperly chosen. b) The computational complexity increases sharply with higher degrees l and body orders ν, resulting in significant computational costs.

3.3 Reformulating equivariant GNNs: a perspective based on output space

Expanding the basis set of a complete equivariant GNN is theoretically appealing but practically challenging. In this section, we adopt an alternative approach by deriving the necessary model design from the characteristics of the desired output results. We identify a key insight: a complete lth-degree steerable model can be constructed using two components—a complete scalar model and a full-rank basis set of lth-degree steerable features.

[3] For clarity, our main text focuses exclusively on graph-level functions, where $\tilde{\boldsymbol{\phi}}^{(l)}(\mathcal{G})$ remains invariant to any node permutation. Notably, our discussion is generalizable to both node- and edge-level functions, which will be discussed in Appendix A.2

We denote the target lth-degree steerable function as $\tilde{\boldsymbol{\phi}}^{(l)} \in \mathbb{F}^{(l)}$. Although it might be represented with infinite bases in $\mathbb{B}^{(l)}_{\text{cpl}}$, the output of $\tilde{\boldsymbol{\phi}}^{(l)}(\mathcal{G})$ is still a vector in $\mathbb{R}^{2l+1}$. Let $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G}) = [\tilde{\boldsymbol{v}}^{(l)}_1(\mathcal{G}), \ldots, \tilde{\boldsymbol{v}}^{(l)}_C(\mathcal{G})] \in \mathbb{R}^{(2l+1)\times C}$ define an lth-degree steerable model with C-channel outputs of lth-degree steerable feature ($C \geq 2l+1$). We have the following theorem:

Theorem 3.2 (Dynamic Method)**.** *Given a geometric graph $\mathcal{G}$, suppose there is a matrix $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ with C channels of lth-degree steerable features denoted as $\tilde{\boldsymbol{v}}^{(l)}_c(\mathcal{G})$ satisfying $\operatorname{span}(\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})) = \mathbb{F}^{(l)}(\mathcal{G}) \coloneqq \{\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}) \mid \tilde{\boldsymbol{f}}^{(l)} \in \mathbb{F}^{(l)}\} \subset \mathbb{R}^{2l+1}$. Then for any lth-degree steerable function $\tilde{\boldsymbol{\phi}}^{(l)} \in \mathbb{F}^{(l)}$, there always exists $\tilde{\boldsymbol{w}}^{(0)}(\mathcal{G}) \coloneqq [\tilde{w}^{(0)}_c(\mathcal{G})]^C_{c=1}$ with C-channel output scalars, such that*

$$\tilde{\boldsymbol{\phi}}^{(l)}(\mathcal{G}) = \textstyle\sum_{c=1}^{C} \tilde{w}^{(0)}_c(\mathcal{G}) \tilde{\boldsymbol{v}}^{(l)}_c(\mathcal{G}). \tag{4}$$

A notable distinction from Eq. (2) is that the coefficient in front of the steerable features changes from a_i, which is independent of the input, to a scalar (function) that depends on the entire graph, denoted as $\tilde{w}^{(0)}_c(\mathcal{G})$. A natural idea is that if we can get a complete scalar model (capturing all possible scalars), then we can always satisfy the requirements of Theorem 3.2. Another crucial consideration is that the validity of Theorem 3.2 requires the basis matrix $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ to satisfy $\operatorname{span}(\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})) = \mathbb{F}^{(l)}(\mathcal{G})$. We call such basis matrix that meets this criterion as $\mathbb{F}^{(l)}(\mathcal{G})$-*full-rank*. When $\mathbb{F}^{(l)}(\mathcal{G}) = \mathbb{R}^{2l+1}$, the concept of $\mathbb{F}^{(l)}(\mathcal{G})$-*full-rank* is equivalent to the traditional definition of *full-rank*. For simplicity, in contexts with no ambiguity, we refer to it as *full-rank*.

By comparing Eq. (4) with Eq. (2), our proposed dynamic method offers greater flexibility and simplicity, embodying the concept of complexity transfer. Instead of constructing the difficult complete basis set $\mathbb{B}^{(l)}_{\text{cpl}}$, we transform the process into obtaining a complete scalar function and then design the complete steerable model through Theorem 3.2. In the next section, we will detail how to acquire the complete scalar model and the full-rank basis set of lth-degree steerable features needed for Theorem 3.2.

3.4 Reach completeness: canonical form and full-rank basis set

In this section, we introduce how to obtain the complete scalar function $\tilde{w}^{(0)}_c(\mathcal{G})$ (referred to as the canonical form) and the full-rank basis set $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ required by Theorem 3.2.

Geometric isomorphism and canonical form. An equivalent problem to constructing a complete scalar function is determining the isomorphism of geometric graphs. To differentiate this from traditional graph isomorphism problem for topological graphs, we specifically refer to our task involving geometric graphs as *geometric isomorphism*. This concept can be defined similarly to the GWL-test [36], which assesses the equivalence of geometric graphs based on their geometric structures and embeddings.

Definition 3.3 (Geometric Isomorphism)**.** *Two geometric graphs $\mathcal{G}(\vec{\boldsymbol{X}}^{(\mathcal{G})}, \boldsymbol{A}^{(\mathcal{G})})$ and $\mathcal{H}(\vec{\boldsymbol{X}}^{(\mathcal{H})}, \boldsymbol{A}^{(\mathcal{H})})$ are called* geometrically isomorphic *if they fulfill both of the following isomorphisms*

1. ***Point Cloud Isomorphism:*** *The two point clouds $\vec{\boldsymbol{X}}^{(\mathcal{G})}$ and $\vec{\boldsymbol{X}}^{(\mathcal{H})}$ are isomorphic,* i.e.*, $\exists\, \sigma \in \mathrm{S}_N, \mathfrak{g} \in \mathrm{E}(3), \forall i, \vec{\boldsymbol{x}}^{(\mathcal{G})}_i = \mathfrak{g} \cdot \vec{\boldsymbol{x}}^{(\mathcal{H})}_{\sigma(i)}$. Here, all $\langle \sigma, \mathfrak{g} \rangle$ make a nonempty set $\mathbb{M}(\mathcal{G}, \mathcal{H})$.*
2. ***Topological Isomorphism:*** *The topological graphs associated with the point clouds are isomorphic,* i.e.*, $\exists \langle \sigma, \mathfrak{g} \rangle \in \mathbb{M}(\mathcal{G}, \mathcal{H}), \forall i, \forall j, [\boldsymbol{A}^{(\mathcal{G})}_{ij}] = [\boldsymbol{A}^{(\mathcal{H})}_{\sigma(i)\sigma(j)}]$.*

Moreover, we denote the geometric isomorphism between $\mathcal{G}$ and $\mathcal{H}$ as $\mathcal{G} \cong \mathcal{H}$.

The most significant difference from the isomorphism problem in topological graphs is that, to date, it remains unclear whether there exists a polynomial-time algorithm capable of determining whether two topological graphs are isomorphic [42; 63]. In contrast, when it comes to distinguishing whether two geometric graphs are geometrically isomorphic in accordance with Definition 3.3, the situation appears to be more favorable. There are indeed algorithms that can resolve this issue in polynomial time, as highlighted by [64]. For example, we present Algos. 1 and 2, which is based on the four-point positioning principle and operates with complexities of $\mathcal{O}(N^6)$ and $\mathcal{O}(N^8)$, respectively.

In most applications, the objective of an equivariant GNN is not to distinguish two geometric graphs, but rather to obtain a comprehensive embedding of a geometric graph. Consequently, we define the *canonical form* similarly to the definition provided in [64] as follows:

Definition 3.4 (Canonical Form of Geometric Graph)**.** *A canonical form of geometric graph is a graph-level scalar function* $\Gamma : (\mathbb{R}^{N\times 3}, \mathbb{R}^{N\times N}) \to \mathbb{R}^H$*, satisfy* $\mathcal{G} \cong \mathcal{H} \iff \Gamma(\mathcal{G}) = \Gamma(\mathcal{H})$.

Built upon Algos. 1 and 2, we have derived a canonical form $\Gamma(\cdot)$ as presented in Algo. 3. In summary, the process involves three steps: a) traversing all ordered four-point sets to serve as reference points for four-point positioning, with a time complexity of $\mathcal{O}(N^4)$; b) transforming the original coordinates into distance vectors measured from the reference points, and converting both point sets and edge sets into scalar sets, with a time complexity of $\mathcal{O}(N)$; c) mapping these scalar sets to the desired canonical form using DeepSet [65].

Theorem 3.5 (General Canonical Form)**.** *Given any geometric graph* $\mathcal{G}$*, Algo. 3 provides a method to create canonical form with time complexity* $\mathcal{O}(N^6)$.

However, Algo. 3 remains impractical because of its high complexity. We now explore the feasibility of attaining a more efficient canonical form by sacrificing some versatility. The crux of Algo. 3 lies in the choice of four non-coplanar points, which contributes to a quartic complexity in traversal. If we reformulate this as a generative problem, by treating the non-coplanar points as learnable graph-level features, we may reduce this quartic complexity factor. Specifically, we consider applying a $\mathrm{E}(3)$-equivariant GNN ζ to generate four non-coplanar nodes (called virtual nodes below). We regard them as reference points in the four-point positioning principle, avoiding the fourth-order complexity cost by the original traversal. Thus, we derive Algo. 4, yielding more efficiency.

Theorem 3.6 (Faster Canonical Form)**.** *Given an* $\mathrm{E}(3)$*-equivariant function* ζ *that generates four non-coplanar points on* $\mathcal{G}$*, Algo. 4 is able to create canonical form with time complexity* $\mathcal{O}(N^2)$.

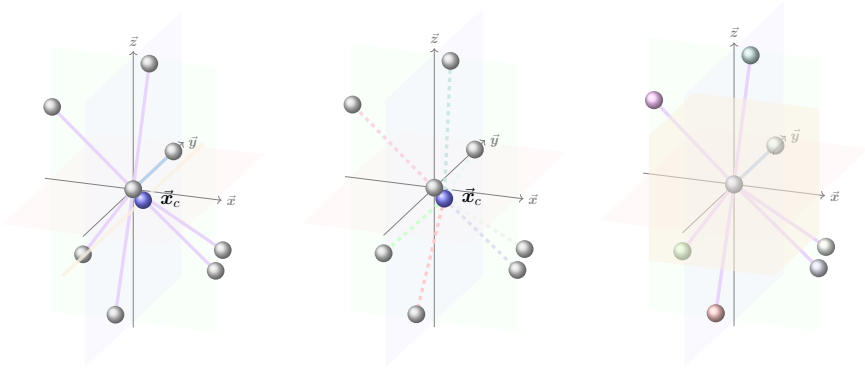
It should be noted that Algo. 3 can encode any geometric graph, while Algo. 4 requires that four non-coplanar reference nodes can be learned through an equivariant GNN. In past studies, such as FastEGNN [66], it is assumed that non-coplanar virtual nodes can always be found, but in the discussion later, we will see that this is not the case.

Full-rank basis set. As previously addressed, the validity of Eq. (4) critically depends on the $\mathbb{F}^{(l)}(\mathcal{G})$-full-rank basis set, a condition not easy to satisfy. The study in [49] has demonstrated that, for symmetric geometric graphs (defined in Definition A.5), $\mathbb{F}^{(l)}(\mathcal{G})$ will degenerate to zero functions for certain values of degree l, implying that it is hard to design a full-rank basis set in this case. In practice, most scenarios study the case of asymmetric graphs, so we give priority to discussing them here. Our key inquiry is whether full-rank basis sets can always be constructed for asymmetric geometric graphs. We first present a pivotal theorem that significantly aids in our analysis.

Theorem 3.7 (Coloring on Asymmetric Graph)**.** *In an asymmetric geometric graph, each point can be assigned a unique color. This implies the existence of an* $\mathrm{E}(3)$ *invariant function that maps the features of the* i*th node to distinct values* $\boldsymbol{h}_i$*. Similarly, for directed edges* ij*, their features are mapped to distinct values* $\boldsymbol{e}_{ij}$.

We propose two node coloring methods: a) Distance to the center, denoted as $\oplus$; b) Tensor product, denoted as $\otimes$. We denote the uncolored model by $\varnothing$ and illustrate the coloring procedure in Fig. 2, taking the $\oplus$ method as an example. A detailed description is provided in Table 9. This theorem is fundamental and forms the core of our subsequent analyses. It decouples the roles of different nodes and edges, simplifying our discussion. This framework enables us to enhance or suppress specific basis functions by selecting appropriate weight functions, or even to focus on a particular basis function by setting the weights of all other basis functions to zero. Given that the node coordinates of the entire geometric graph are non-coplanar, the coordinate differences $\{\vec{\boldsymbol{x}}_{ij}\}$ form a full-rank matrix, which allows us to achieve the desired $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$. Additionally, spherical harmonic functions facilitate the mapping of first-degree steerable features to higher-degree representations. We obtain the following theorem.

Theorem 3.8 (Existence of Full-Rank Basis Set)**.** *For any given asymmetric graph* $\mathcal{G}$*, an* $\mathbb{F}^{(l)}(\mathcal{G})$*-full-rank* $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ *can always be constructed for any degree* l.



(a) Original (b) Center (c) Colored

Figure 2: Different vectors are decoupled by coloring, thereby expanding the space of virtual nodes. The yellow area in the figure represents the out space of 1st-degree functions. Fig. 2(a): All nodes have the same features, and the output can only appear in a straight line in one-dimensional space. Fig. 2(b): Calculate the distance to the center $\|\vec{\boldsymbol{x}}_{ic}\|$ to update each node feature so that each node has a different color (feature). Fig. 2(c): After coloring, all vectors are decoupled, so the output can be generated in the entire 3D space.

For symmetric graphs, the situation is more complex, making the construction of a full-rank basis set considerably difficult. An potentially feasible approach could be using a tensor-product-based model, such as TFN [33] for basis set construction, which will be detailed in the appendix Appendix A.1.3.

3.5 Implementation of Complete Models

The last subsection demonstrated how to achieve theoretical completeness. Here, we present a practical implementation of complete models, using EGNN [43] as an example due to its simplicity and universality. In Example 3.1, we point out that EGNN could be incomplete since its basis set $\mathbb{B}^{(1)}$ is possibly reduced to the sum of all edges $\{\vec{\boldsymbol{x}}_{ij}\}$. However, the edge set itself $\{\vec{\boldsymbol{x}}_{ij}\}$ is full-rank and could be consider as the full-rank basis set $\tilde{\boldsymbol{V}}^{(1)}$ in Theorem 3.2. To make EGNN complete, the remaining issue is how to implement the complete scalar function $\tilde{w}_c^{(0)}$. Inspired by Algo. 4, we derive the $\mathrm{E}(3)$-equivariant function ζ in Theorem 3.6 using FastEGNN [66; 67], along with the pre-coloring process and several other modifications.

Essentially, we first color the geometric graph and then construct the reference virtual nodes through FastEGNN in the following form:

$$\vec{\boldsymbol{Z}} = \mathbf{1}_4\vec{\boldsymbol{x}}_c + \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\varphi_{\vec{\boldsymbol{Z}}}(\boldsymbol{h}_i)\cdot(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_c), \qquad \vec{\boldsymbol{M}}_i = \mathbf{1}_4\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{Z}}, \tag{5}$$

where $\vec{\boldsymbol{Z}} \in \mathbb{R}^{4\times 3}$ denotes the coordinates of four reference virtual nodes, $\vec{\boldsymbol{M}}_i \in \mathbb{R}^{4\times 3}$ computes coordinate difference between node i and all virtual nodes, and $\vec{\boldsymbol{x}}_c = \frac{1}{N}\sum_{i=1}^{N}\vec{\boldsymbol{x}}_i$ signifies the coordinate center. In Algo. 4, the key point is to convert the coordinate information into a distance vector to the reference virtual node. To further refine the information, we employ the inner product form from GMN [68], as an alternative to distance vectors in Algo. 4. The node features are updated as follows:

$$\boldsymbol{h}_i = \boldsymbol{h}_i + \varphi_h(\boldsymbol{m}_i), \qquad \boldsymbol{m}_i = \vec{\boldsymbol{M}}_i^{\top}\vec{\boldsymbol{M}}_i / \|\vec{\boldsymbol{M}}_i^{\top}\vec{\boldsymbol{M}}_i\|_{\mathrm{F}}. \tag{6}$$

Afterward, global operations (*e.g.*, pooling) aggregate the information from all points and edges, mimicking DeepSet [65] calculations to obtain the desired canonical form. Strictly speaking, the virtual nodes forming the tetrahedron should also be included in Algo. 4, but we omit this term as it degrades performance in our experiments.

If we consider a single-layer EGNN, it becomes the following form:

$$\texttt{EGNN}_{\text{cpl}}(\mathcal{G}) = \tfrac{1}{N}\textstyle\sum_{i=1}^{N}\vec{\boldsymbol{x}}_i' = \vec{\boldsymbol{x}}_c + \sum_{\langle i,j\rangle\in\mathcal{E}}\varphi_{\vec{\boldsymbol{x}}}(\boldsymbol{m}_{ij,\text{cpl}})\cdot\vec{\boldsymbol{x}}_{ij}, \tag{7}$$

where the message $\boldsymbol{m}_{ij,\text{cpl}} = \varphi_{\boldsymbol{m}}(\boldsymbol{h}_i, \boldsymbol{h}_j, \|\vec{\boldsymbol{x}}_{ij}\|^2, \boldsymbol{e}_{ij})$ has been a complete scalar function since $\boldsymbol{h}_i$ already contains the information in the canonical form. Through Theorem 3.2, we verify that such single-layer EGNN is a complete model.

The above approach can also be extended to tensor-product-based models, and we have implemented corresponding improvements in TFN [33]. Since virtual nodes (*i.e.* $\vec{\boldsymbol{Z}}$) are generated globally across

Table 1: *The Completeness Test*.

GNN Layer	The Completeness Test 2-body (Table. 3 in GWL)	3-body (Fig. 2(b) in IASR)
$\text{SchNet}_{\text{2-body}}$	50.0 ± 0.0	50.0 ± 0.0
$\text{EGNN}_{\text{2-body}}$	50.0 ± 0.0	50.0 ± 0.0
$\text{GVP-GNN}_{\text{3-body}}$	**100.0 ± 0.0**	50.0 ± 0.0
$\text{TFN}_{\text{2-body}}$	50.0 ± 0.0	50.0 ± 0.0
$\text{MACE}_{\text{3-body}}$	**100.0 ± 0.0**	50.0 ± 0.0
$\text{MACE}_{\text{4-body}}$	**100.0 ± 0.0**	**100.0 ± 0.0**
$\text{Basic}_{\text{cpl}}$	**100.0 ± 0.0**	**100.0 ± 0.0**
$\text{SchNet}_{\text{cpl}}$	**100.0 ± 0.0**	**100.0 ± 0.0**
EGNN_{cpl}	**100.0 ± 0.0**	**100.0 ± 0.0**
$\text{GVP-GNN}_{\text{cpl}}$	**100.0 ± 0.0**	**100.0 ± 0.0**
TFN_{cpl}	**100.0 ± 0.0**	**100.0 ± 0.0**

Table 2: *The Chirality Test*.

	# Color	# TP	The Chirality Test Fig. 4(a)	Fig. 4(b)	Fig. 4(c)
Basic	∅		50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
	⊕		**100.0 ± 0.0**	**100.0 ± 0.0**	50.0 ± 0.0
	⊗		**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	∅	✓	75.0 ±15.0	95.0 ±15.0	**100.0 ± 0.0**
	⊕	✓	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	⊗	✓	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
EGNN	∅		50.0 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
	⊕		**100.0 ± 0.0**	**100.0 ± 0.0**	50.0 ± 0.0
	⊗		**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	∅	✓	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	⊕	✓	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
	⊗	✓	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**

Table 3: MSE loss for predicting the coordinates of the Monge point and twelve-point center and incenter of a tetrahedron. Models with a superscript $*$ indicate that they utilize only half of the hidden dimension of the standard node embeddings. The optimal value is denoted in **bold**, while the suboptimal value is indicated with an underline.

	Monge Point				Twelve-point Center ($\times 10^{-1}$)				Incenter ($\times 10^{-2}$)			
	1-layer	2-layer	3-layer	4-layer	1-layer	2-layer	3-layer	4-layer	1-layer	2-layer	3-layer	4-layer
EGNN	1.188	0.374	0.185	0.131	1.320	0.413	0.265	0.161	32.54	0.527	0.141	0.036
FastEGNN	1.187	0.374	0.149	0.979	1.320	0.417	0.194	0.107	32.40	0.421	0.188	0.055
HEGNN	1.188	0.356	0.157	0.110	1.320	0.317	0.192	0.159	32.54	0.386	0.078	0.058
TFN	1.188	0.467	0.289	0.221	1.320	0.519	0.315	0.246	32.54	2.715	0.151	0.060
MACE	1.188	0.468	0.288	0.223	1.320	0.521	0.316	0.246	32.54	2.707	0.153	0.090
Equiformer	0.416	0.296	0.099	0.116	0.456	0.187	0.122	0.147	0.736	0.077	0.055	0.054
EGNN_{cpl}-global	**0.108**	**0.086**	0.097	0.104	**0.112**	**0.097**	0.099	0.088	**0.025**	**0.018**	**0.015**	**0.012**
$\text{EGNN}^{*}_{\text{cpl}}$-global	0.175	0.116	0.148	0.111	0.155	0.144	0.174	0.152	0.046	0.025	0.059	0.026
EGNN_{cpl}-local	0.460	0.210	0.083	**0.052**	0.510	0.232	**0.095**	**0.067**	0.388	0.258	0.031	0.029
$\text{EGNN}^{*}_{\text{cpl}}$-local	0.460	0.252	0.138	0.091	0.508	0.250	0.182	0.102	0.395	0.320	0.084	0.090
TFN_{cpl}-global	0.468	0.162	0.233	0.084	0.516	0.319	0.187	0.099	8.983	0.398	0.077	0.051
$\text{TFN}^{*}_{\text{cpl}}$-global	0.473	0.291	0.158	0.096	0.525	0.317	0.181	0.105	7.858	0.354	0.085	0.054
TFN_{cpl}-local	0.460	0.210	0.089	0.088	0.518	0.237	0.097	0.078	0.676	0.048	0.029	0.021
$\text{TFN}^{*}_{\text{cpl}}$-local	0.473	0.244	**0.082**	0.073	0.519	0.271	0.122	0.085	0.693	0.056	0.039	0.024

the entire graph, we refer to these models as $\text{EGNN/TFN}_{\text{cpl}}$-global. To better capture local information, virtual nodes can also be generated for individual nodes, resulting in the $\text{EGNN/TFN}_{\text{cpl}}$-local variants. Detailed model architectures are provided in Tables 12 and 13.

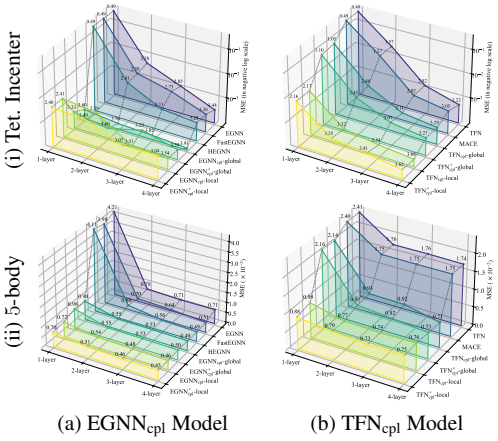
4 Experiment

In this section, we conduct two categories of experiments, totaling seven in number, to validate our theory and methods: a) three toy datasets in § 4.1 to assess the expressivity of our models; b) the remaining four in § 4.2 evaluate the actual performance of our models. Here, we provide a brief overview of the experiments. For further details, please refer to Appendix B.

4.1 Expressivity

Dataset setup. The **completeness test** is derived from the GWL-test [32] and the IASR-test [69], with the objective of distinguishing between two geometrically non-isomorphic graphs. We here focus on the k-*chain* test and 2-body/3-body tasks for testing. The **chirality test** is an experiment designed by ourselves, which includes three tasks as illustrated in Fig. 4. This study aims to address two objectives: a) assess the model's ability to distinguish geometric graphs under reflection transformations; (2) investigate whether distinct coloring strategies generate different virtual node coordinates, as validated by determinant calculations serving as mathematical indicators for reflection determination.

Models. For the completeness test, we selected several baseline models, including SchNet [70], EGNN [43], GVP-GNN [71], TFN [33], and MACE [35]. Furthermore, models utilizing our canonical form are denoted as $\textit{Model}_{\text{cpl}}$, with $\textit{Basic}_{\text{cpl}}$ specifically using the canonical form as the graph embedding directly. For the chirality test, we provide two options to identify reflections: a) Determinant $\det([\vec{z}_2, \vec{z}_3, \vec{z}_4] - \vec{z}_1 \mathbf{1}_{1\times 3})$; b) Tensor product, as described in Example A.15.



(a) EGNN_{cpl} Model (b) TFN_{cpl} Model

Figure 3: Visualization of MSE loss.

Table 4: MSE loss on 5-body system.

	MSE Loss on 5-body system ($\times 10^{-2}$)			
	1-layer	2-layer	3-layer	4-layer
EGNN	4.214	0.780	0.710	0.712
FastEGNN	3.983	0.705	0.640	0.509
HEGNN	4.114	0.801	0.561	0.489
TFN	2.411	1.758	1.758	1.739
MACE	2.403	1.754	1.746	1.746
Equiformer	0.805	0.682	0.465	0.657
EGNN_{cpl}-global	0.943	0.546	0.530	0.492
$\text{EGNN}^{*}_{\text{cpl}}$-global	0.985	0.554	0.533	0.498
EGNN_{cpl}-local	0.768	0.537	0.481	0.458
$\text{EGNN}^{*}_{\text{cpl}}$-local	**0.703**	**0.513**	**0.455**	**0.450**
TFN_{cpl}-global	2.144	0.934	0.916	0.708
$\text{TFN}^{*}_{\text{cpl}}$-global	2.163	0.910	0.825	0.734
TFN_{cpl}-local	0.988	0.766	0.738	0.761
$\text{TFN}^{*}_{\text{cpl}}$-local	0.883	0.792	0.734	0.746

Table 5: 100-body dataset.

	MSE Loss ($\times 10^{-2}$)
EGNN	1.36
FastEGNN	1.10
$\text{TFN}_{l\leq 2}$	3.77
$\text{MACE}_{l\leq 2}$	3.83
$\text{Equiformer}_{l\leq 2}$	0.90
$\text{HEGNN}_{l\leq 1}$	1.13
$\text{HEGNN}_{l\leq 2}$	0.97
$\text{HEGNN}_{l\leq 3}$	0.94
$\text{HEGNN}_{l\leq 6}$	0.86
EGNN_{cpl}-global	0.98
EGNN_{cpl}-local	**0.73**
$\text{TFN}_{\text{cpl}}\text{-global}_{l\leq 2}$	1.78
$\text{TFN}_{\text{cpl}}\text{-local}_{l\leq 2}$	1.73

Table 6: Prediction error ($\times 10^{-2}$) on MD17 dataset (3 runs).

	Aspirin	Benzene	Ethanol	Malonal.	Naph.	Salicylic	Toluene	Uracil
EGNN	14.41±0.15	62.40±0.53	4.64±0.01	13.64±0.01	0.47±0.02	1.02±0.02	11.78±0.07	0.64±0.01
FastEGNN	9.81±0.11	60.84±0.14	4.65±0.00	12.82±0.02	0.38±0.00	1.05±0.08	10.88±0.08	0.56±0.01
$\text{TFN}_{l\leq 2}$	12.37±0.18	58.48±1.98	4.81±0.04	13.62±0.08	0.49±0.01	1.03±0.02	10.89±0.01	0.84±0.02
$\text{MACE}_{l\leq 2}$	10.43±0.44	59.71±2.21	4.83±0.03	13.78±0.04	0.44±0.02	0.94±0.01	10.20±0.11	0.74±0.01
$\text{Equiformer}_{l\leq 2}$	9.84±0.10	**33.28**±0.15	4.69±0.03	13.06±0.04	0.34±0.01	0.86±0.01	**9.50**±0.09	0.57±0.01
$\text{HEGNN}_{l\leq 1}$	10.32±0.58	62.53±7.62	4.63±0.01	12.85±0.01	0.38±0.01	0.90±0.05	10.56±0.10	0.56±0.02
$\text{HEGNN}_{l\leq 2}$	10.04±0.45	61.80±5.92	4.63±0.01	12.85±0.01	0.39±0.01	0.91±0.06	10.56±0.05	0.55±0.01
$\text{HEGNN}_{l\leq 3}$	10.20±0.23	62.82±4.25	4.63±0.01	12.85±0.02	0.37±0.01	0.94±0.10	10.55±0.16	**0.52**±0.01
$\text{HEGNN}_{l\leq 6}$	9.94±0.07	59.93±5.21	4.62±0.01	12.85±0.01	0.37±0.02	0.88±0.02	10.56±0.33	0.54±0.01
EGNN_{cpl}-global	9.60±0.09	58.24±1.40	4.64±0.01	12.85±0.01	0.39±0.01	0.95±0.05	10.37±0.16	0.56±0.02
EGNN_{cpl}-local	9.52±0.42	44.90±1.53	**4.62**±0.00	**12.80**±0.02	0.36±0.02	0.94±0.05	10.21±0.06	0.57±0.00
$\text{TFN}_{\text{cpl}}\text{-global}_{l\leq 2}$	**9.49**±0.04	58.24±0.42	4.63±0.00	12.82±0.00	**0.33**±0.00	**0.80**±0.00	10.24±0.02	0.53±0.00
$\text{TFN}_{\text{cpl}}\text{-local}_{l\leq 2}$	9.52±0.07	48.77±6.51	4.64±0.00	12.83±0.02	0.34±0.00	0.81±0.01	10.95±0.01	0.53±0.00

Results. Results are presented in Table 1 (the 2-body/3-body of completeness test), Table 2 (the chirality test) and Table 10 (the *k-chain* test of completeness test). As shown in Table 1, all models with canonical form, *i.e.* $\text{Model}_{\text{cpl}}$, reach an 100% accuracy, no matter whether the original model could distinguish the geometric graphs. As shown in Table 2, the model utilizing the determinant is highly dependent on the coloring scheme. In the special case illustrated in Fig. 4(c), only the model employing the tensor product can effectively distinguish between the geometric graphs, which demonstrates the importance of choosing an appropriate staining strategy.

4.2 Performance on Physical Systems

Dataset setup. We conducted four experiments to evaluate our model's performance across different scenarios: a) **tetrahedron center prediction**, which involves predicting the Monge point, twelve-point center, and incenter of a tetrahedron based on its four points (with identical node features); b) 5**-body system** [72], c) 100**-body system** [66], d) **MD17 dataset** [68], and e) **Water-3D mini** [66; 67], where predictions are made based on initial coordinates and velocities. Specifically, for the 5-body, 100-body, and Water-3D mini systems, we predict the future positions of charged particles or fluid particles, while for the MD17 dataset, we predict future atomic trajectories within eight different molecules.

These experiments provide a comprehensive evaluation of our model. We assess its performance on graph-level targets (a) and node-level targets (b-e), explore the influence of model architecture layers (a-b), and test its scalability in large systems (c,e) as well as its applicability to real-world datasets (d-e). Detailed information on each dataset can be found in Table 11.

Models. In both experiments, we selected the following baseline models: EGNN [43], FastEGNN [66; 67], HEGNN [49], TFN [33], MACE [35], and Equiformer [73]. We compared these baseline models with our proposed models, specifically $\text{EGNN/TFN}_{\text{cpl}}$-local and $\text{EGNN/TFN}_{\text{cpl}}$-global. More detailed information is provided in Tables 14 and 15.

Results. We evaluated the performance of each model across varying numbers of layers in the first two experiment. For specific values, please refer to Table 3 and Table 4. The experimental results are visualized in Fig. 3 to facilitate comparison. Our model demonstrates superior performance across all

Table 7: Results on Water-3D-mini.

	MSE Loss on Water-3D-mini ($\times 10^{-4}$)			
	1-layer	2-layer	3-layer	4-layer
EGNN	4.904	4.323	3.649	3.338
FastEGNN	4.885	4.332	3.782	3.259
HEGNN	4.885	4.138	3.519	3.287
EGNN_{cpl}-global	4.368	3.711	3.294	3.248
EGNN_{cpl}-local	**3.611**	**3.320**	**2.803**	**2.495**

tasks, often achieving results comparable to those of other models with multiple layers—sometimes even with just a single layer. In addition, our improved model has also achieved significant improvements on large-scale datasets and real-world datasets, see Tables 5 to 7, where our model is in the leading position in the former and in the latter, where we achieve performance leadership in more than half of the tasks.

Additional findings. Furthermore, the baselines in Fig. 3 demonstrate that losses are equal when a single layer is used, confirming the degradation predicted in Example A.16 and highlighting the necessity of node coloring. While TFN and MACE outperform EGNN and HEGNN with a single layer, their advantage decreases with two or more layers. This is likely because EGNN/HEGNN may initially lack necessary basis functions, whereas tensor products used in TFN/MACE in deeper layers produce redundant terms. The introduction of the dynamic method significantly enhances the performance of both EGNN and TFN, showing its ability to construct the missing basis initially and suppress redundant terms in tensor products, thus achieving *adaptive* basis construction. From this perspective, exploring how to encode invariant features with more powerful architectures (*e.g.*, advancing from GNNs to Graph Transformers [74]) represents a promising direction for future research. This perspective also helps explain the performance gains observed in models such as GotenNet [50] compared with HEGNN [49].

5 Conclusion

We propose a novel framework called Uni-EGNN, which is a dynamic method for constructing complete equivariant GNNs. In contrast to previous models, which required stacking multiple layers, increasing body order, and enhancing the degree of steerable features to achieve completeness, our dynamic method simplifies this process by requiring only two essential components: a) A canonical form (a complete scalar function) of a geometric graph; b) A full-rank steerable basis set. Additionally, we have developed an efficient implementation of dynamic method leveraging a polynomial algorithm for geometric isomorphism problems. Experimental results demonstrate that our dynamic method excels in both expressiveness and practical performance.

6 Limitations

For symmetric graphs, constructing a full-rank basis set remains an open problem. To date, it has only been established that tensor-product-based models (*e.g.*, TFN [33]) can construct such a basis, while for scalar methods (*e.g.*, HEGNN [49]) this remains uncertain—and we even conjecture that it may not be feasible. Another limitation of this work lies in the experimental evaluation: due to the lack of tasks with objective functions of degree $l \geq 2$, we are unable to empirically validate the effectiveness of the proposed method in predicting high-degree steerable objective functions.

Acknowledgement

This work was jointly supported by the following projects and institutions: the National Natural Science Foundation of China (No. 62376276, No. 62172422); Beijing Nova Program (No. 20230484278); the Fundamental Research Funds for the Central Universities; the Research Funds of Renmin University of China (23XNKJ19); Public Computing Cloud, Renmin University of China; Damo Academy (Hupan Laboratory) through Damo Academy (Hupan Laboratory) Innovative Research Program and Damo Academy Research Intern Program.

References

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

[2] Wenbing Huang and Jiacheng Cen. Geometric graph learning for drug design. *Deep Learning in Drug Design*, pages 133–151, 2026.

[3] Wenjie Liu, Yifan Zhu, Ying Zha, Qingshan Wu, Lei Jian, and Zhihao Liu. Rotation-and permutation-equivariant quantum graph neural network for 3d graph data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[4] Liming Wu, Zhichao Hou, Jirui Yuan, Yu Rong, and Wenbing Huang. Equivariant spatio-temporal attentive graph networks to simulate physical dynamics. *Advances in Neural Information Processing Systems*, 36, 2023.

[5] Chaohao Yuan, Maoji Wen, Ercan Engin Kuruoğlu, Yang Liu, Jia Li, Tingyang Xu, Deli Zhao, Hong Cheng, and Yu Rong. Non-stationary equivariant graph neural networks for physical dynamics simulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[6] Jiaqi Han, Minkai Xu, Aaron Lou, Haotian Ye, and Stefano Ermon. Geometric trajectory diffusion models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[7] Kai Yang, Yuqi Huang, Junheng Tao, Wanyu Wang, and Qitian Wu. Physics-inspired all-pair interaction learning for 3d dynamics modeling. *arXiv preprint arXiv:2510.04233*, 2025.

[8] Tong Wang, Xinheng He, Mingyu Li, Yatao Li, Ran Bi, Yusong Wang, Chaoran Cheng, Xiangzhen Shen, Jiawei Meng, He Zhang, et al. Ab initio characterization of protein molecular dynamics with ai2bmd. *Nature*, pages 1–9, 2024.

[9] Yusong Wang, Tong Wang, Shaoning Li, Xinheng He, Mingyu Li, Zun Wang, Nanning Zheng, Bin Shao, and Tie-Yan Liu. Enhancing geometric representations for molecules with equivariant vector-scalar interactive message passing. *Nature Communications*, 15(1):313, 2024.

[10] Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

[11] John B Ingraham, Max Baranov, Zak Costello, Karl W Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M Lord, Christopher Ng-Thow-Hing, Erik R Van Vlack, et al. Illuminating protein space with a programmable generative model. *Nature*, pages 1–9, 2023.

[12] J Thorben Frank. Modeling larger length and time scales in machine learning force fields, 2025.

[13] Zian Li, Cai Zhou, Xiyuan Wang, Xingang Peng, and Muhan Zhang. Geometric representation condition improves equivariant molecule generation. In *Forty-second International Conference on Machine Learning*, 2025.

[14] Shaoheng Yan, Zian Li, and Muhan Zhang. Georecon: Graph-level representation learning for 3d molecules via reconstruction-based pretraining. *arXiv preprint arXiv:2506.13174*, 2025.

[15] Zongzhao Li, Jiacheng Cen, Wenbing Huang, Taifeng Wang, and Le Song. Size-generalizable rna structure evaluation by exploring hierarchical geometries. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Fanmeng Wang, Wentao Guo, Qi Ou, Hongshuai Wang, Haitao Lin, Hongteng Xu, and Zhifeng Gao. Polyconf: Unlocking polymer conformation generation through hierarchical generative models. In *Forty-second International Conference on Machine Learning*, 2025.

[17] Fanmeng Wang, Minjie Cheng, and Hongteng Xu. WGFormer: An SE(3)-transformer driven by wasserstein gradient flows for molecular ground-state conformation prediction. In *Forty-second International Conference on Machine Learning*, 2025.

[18] Yurou Liu, Jiahao Chen, Rui Jiao, Jiangmeng Li, Wenbing Huang, and Bing Su. DenoiseVAE: Learning molecule-adaptive noise distributions for denoising-based 3d molecular pre-training. In *The Thirteenth International Conference on Learning Representations*, 2025.

[19] Fanglei Xue, Meihan Zhang, Shuqi Li, Xinyu Gao, James A Wohlschlegel, Wenbing Huang, Yi Yang, and Weixian Deng. Se (3)-equivariant ternary complex prediction towards target protein degradation. *Nature Communications*, 16(1):5514, 2025.

[20] Liming Wu, Wenbing Huang, Rui Jiao, Jianxing Huang, Liwei Liu, Yipeng Zhou, Hao Sun, Yang Liu, Fuchun Sun, Yuxiang Ren, et al. Siamese foundation models for crystal structure prediction. *arXiv preprint arXiv:2503.10471*, 2025.

[21] Qi Li, Rui Jiao, Liming Wu, Tiannian Zhu, Wenbing Huang, Shifeng Jin, Yang Liu, Hongming Weng, and Xiaolong Chen. Powder diffraction crystal structure determination using generative models. *arXiv preprint arXiv:2409.04727*, 2024.

[22] Haitao Lin, Odin Zhang, Jia Xu, Yunfan Liu, Zheng Cheng, Lirong Wu, Yufei Huang, Zhifeng Gao, and Stan Z Li. Tokenizing electron cloud in protein-ligand interaction learning. *arXiv preprint arXiv:2505.19014*, 2025.

[23] Yang Liu, Zinan Zheng, Yu Rong, and Jia Li. Equivariant graph learning for high-density crowd trajectories modeling. *Transactions on Machine Learning Research*, 2024.

[24] Zinan Zheng, Yang Liu, Jia Li, Jianhua Yao, and Yu Rong. Relaxing continuous constraints of equivariant graph neural networks for broad physical dynamics learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4548–4558, 2024.

[25] Yang Liu, Zinan Zheng, Yu Rong, Deli Zhao, Hong Cheng, and Jia Li. Equivariant and invariant message passing for global subseasonal-to-seasonal forecasting. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 1879–1890, 2025.

[26] Yang Liu, Jiashun Cheng, Haihong Zhao, Tingyang Xu, Peilin Zhao, Fugee Tsung, Jia Li, and Yu Rong. SEGNO: Generalizing equivariant graph neural networks with physical inductive biases. In *The Twelfth International Conference on Learning Representations*, 2024.

[27] Anyi Li, Jiacheng Cen, Songyou Li, Mingze Li, Yang Yu, and Wenbing Huang. Geometric mixture models for electrolyte conductivity prediction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[28] Rui Jiao, Hanlin Wu, Wenbing Huang, Yuxuan Song, Yawen Ouyang, Yu Rong, Tingyang Xu, Pengiu Wang, Hao Zhou, Weiying Ma, Jingjing Liu, and Yang Liu. Mof-bfn: Metal-organic frameworks structure prediction via bayesian flow networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[29] Xixian Liu, Rui Jiao, Zhiyuan Liu, Yurou Liu, Yang Liu, Ziheng Lu, Wenbing Huang, Yang Zhang, and Yixin Cao. Learning 3d anisotropic noise distributions improves molecular force fields. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[30] Junjie Xu, Jiahao Zhang, Mangal Prakash, Xiang Zhang, and Suhang Wang. Dualequinet: A dual-space hierarchical equivariant network for large biomolecules. *arXiv preprint arXiv:2506.19862*, 2025.

[31] Jiaqi Han, Jiacheng Cen, Liming Wu, Zongzhao Li, Xiangzhe Kong, Rui Jiao, Ziyang Yu, Tingyang Xu, Fandi Wu, Zihe Wang, et al. A survey of geometric graph neural networks: Data structures, models and applications. *Frontiers of Computer Science*, 19(11):1911375, 2025.

[32] Nadav Dym and Haggai Maron. On the universality of rotation equivariant point cloud networks. In *International Conference on Learning Representations*, 2021.

[33] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[34] Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J Bekkers, and Max Welling. Geometric and physical quantities improve e (3) equivariant message passing. In *International Conference on Learning Representations*, 2021.

[35] Ilyes Batatia, David P Kovacs, Gregor Simm, Christoph Ortner, and Gábor Csányi. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. *Annual Conference on Neural Information Processing Systems*, 35:11423–11436, 2022.

[36] Chaitanya K Joshi, Cristian Bodnar, Simon V Mathis, Taco Cohen, and Pietro Lio. On the expressive power of geometric graph neural networks. In *International conference on machine learning*, pages 15330–15355. PMLR, 2023.

[37] Snir Hordan, Tal Amir, and Nadav Dym. Weisfeiler leman for euclidean equivariant machine learning. In *Proceedings of the 41st International Conference on Machine Learning*, pages 18749–18784, 2024.

[38] Valentino Delle Rose, Alexander Kozachinskiy, Cristóbal Rojas, Mircea Petrache, and Pablo Barceló. Three iterations of (d- 1)-wl test distinguish non isometric clouds of d-dimensional points. *Advances in Neural Information Processing Systems*, 36:9556–9573, 2023.

[39] Yonatan Sverdlov and Nadav Dym. On the expressive power of sparse geometric MPNNs. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Zian Li, Xiyuan Wang, Yinan Huang, and Muhan Zhang. Is distance matrix enough for geometric deep learning? *Advances in Neural Information Processing Systems*, 36:37413–37447, 2023.

[41] Boris Weisfeiler and Andrei Leman. The reduction of a graph to canonical form and the algebra which appears therein. *nti, Series*, 2(9):12–16, 1968.

[42] Brendan D. Mckay and Adolfo Piperno. Practical graph isomorphism, ii. *J. Symb. Comput.*, 60:94–112, January 2014.

[43] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*. PMLR, 2021.

[44] Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.

[45] Junyi An, Xinyu Lu, Chao Qu, Yunfei Shi, Peijia Lin, Qianwei Tang, Licheng Xu, Fenglei Cao, and Yuan Qi. Equivariant spherical transformer for efficient molecular modeling. *arXiv preprint arXiv:2505.23086*, 2025.

[46] Yunyang Li, Lin Huang, Zhihao Ding, Chu Wang, Xinran Wei, Han Yang, Zun Wang, Chang Liu, Yu Shi, Peiran Jin, et al. E2former: An efficient and equivariant transformer with linear-scaling tensor products. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[47] YuQing Xie, Ameya Daigavane, Mit Kotak, and Tess Smidt. The price of freedom: Exploring expressivity and runtime tradeoffs in equivariant tensor products. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[48] J Thorben Frank, Oliver T Unke, Klaus-Robert Müller, and Stefan Chmiela. A euclidean transformer for fast and stable machine learned force fields. *Nature Communications*, 15(1):6539, 2024.

[49] Jiacheng Cen, Anyi Li, Ning Lin, Yuxiang Ren, Zihe Wang, and Wenbing Huang. Are high-degree representations really unnecessary in equivariant graph neural networks? In *Annual Conference on Neural Information Processing Systems*, 2024.

[50] Sarp Aykent and Tian Xia. GotenNet: Rethinking Efficient 3D Equivariant Graph Neural Networks. In *The Thirteenth International Conference on LearningRepresentations*, 2025.

[51] Soledad Villar, David W Hogg, Kate Storey-Fisher, Weichi Yao, and Ben Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *Advances in Neural Information Processing Systems*, 34:28848–28863, 2021.

[52] Zian Li, Xiyuan Wang, Shijia Kang, and Muhan Zhang. On the completeness of invariant geometric deep learning models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[53] Johannes Gasteiger, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[54] Johannes Gasteiger, Florian Becker, and Stephan Gunnemann. Gemnet: Universal directional graph neural networks for molecules. *Advances in Neural Information Processing Systems*, 34:6790–6802, 2021.

[55] Yi Liu, Limei Wang, Meng Liu, Yuchao Lin, Xuan Zhang, Bora Oztekin, and Shuiwang Ji. Spherical message passing for 3d molecular graphs. In *International Conference on Learning Representations*, 2022.

[56] Omri Puny, Matan Atzmon, Edward J. Smith, Ishan Misra, Aditya Grover, Heli Ben-Hamu, and Yaron Lipman. Frame averaging for invariant and equivariant network design. In *International Conference on Learning Representations*, 2022.

[57] Sékou-Oumar Kaba, Arnab Kumar Mondal, Yan Zhang, Yoshua Bengio, and Siamak Ravanbakhsh. Equivariance with learned canonicalization functions. In *International Conference on Machine Learning*, pages 15546–15566. PMLR, 2023.

[58] Justin Baker, Shih-Hsin Wang, Tommaso de Fernex, and Bao Wang. An explicit frame construction for normalizing 3d point clouds. In *Forty-first International Conference on Machine Learning*, 2024.

[59] Shih-Hsin Wang, Yung-Chang Hsu, Justin Baker, Andrea Bertozzi, Jack Xin, and Bao Wang. Rethinking the benefits of steerable features in 3d equivariant graph neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[60] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *Advances in Neural Information Processing Systems*, 31, 2018.

[61] Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks. *arXiv preprint arXiv:2207.09453*, 2022.

[62] Geneviève Dusson, Markus Bachmayr, Gábor Csányi, Ralf Drautz, Simon Etter, Cas van der Oord, and Christoph Ortner. Atomic cluster expansion: Completeness, efficiency and stability. *J. Comput. Phys.*, 454:110946, 2019.

[63] László Babai. Graph isomorphism in quasipolynomial time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 684–697, 2016.

[64] Sergei Evdokimov and Ilia N. Ponomarenko. On the geometric graph isomorphism problem. *Journal of Pure and Applied Algebra*, pages 253–276, 1997.

[65] 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.

[66] Yuelin Zhang, Jiacheng Cen, Jiaqi Han, Zhiqiang Zhang, JUN ZHOU, and Wenbing Huang. Improving equivariant graph neural networks on large geometric graphs via virtual nodes learning. In *Forty-first International Conference on Machine Learning*, 2024.

[67] Yuelin Zhang, Jiacheng Cen, Jiaqi Han, and Wenbing Huang. Fast and distributed equivariant graph neural networks by virtual node learning. *arXiv preprint arXiv:2506.19482*, 2025.

[68] Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. In *International Conference on Learning Representations*, 2022.

[69] Sergey N Pozdnyakov, Michael J Willatt, Albert P Bartók, Christoph Ortner, Gábor Csányi, and Michele Ceriotti. Incompleteness of atomic structure representations. *Physical Review Letters*, 125(16):166001, 2020.

[70] Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24), 2018.

[71] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021.

[72] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, pages 2688–2697. PMLR, 2018.

[73] Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant Graph Attention Transformer for 3D Atomistic Graphs. In *The Eleventh International Conference on Learning Representations*, 2023.

[74] Chaohao Yuan, Kangfei Zhao, Ercan Engin Kuruoglu, Liang Wang, Tingyang Xu, Wenbing Huang, Deli Zhao, Hong Cheng, and Yu Rong. A survey of graph transformers: Architectures, theories and applications. *arXiv preprint arXiv:2502.16533*, 2025.

[75] Kelin Xia and Guo-Wei Wei. Persistent homology analysis of protein structure, flexibility, and folding. *International journal for numerical methods in biomedical engineering*, 30(8):814–844, 2014.

[76] Floor Eijkelboom, Rob Hesselink, and Erik J Bekkers. E(n) equivariant message passing simplicial networks. In *International Conference on Machine Learning*, pages 9071–9081. PMLR, 2023.

[77] Claudio Battiloro, Ege Karaismailoğlu, Mauricio Tec, George Dasoulas, Michelle Audirac, and Francesca Dominici. E(n) equivariant topological neural networks. In *The Thirteenth International Conference on Learning Representations*, 2025.

[78] Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, 99(1):014104, 2019.

[79] Genevieve Dusson, Markus Bachmayr, Gábor Csányi, Ralf Drautz, Simon Etter, Cas van Der Oord, and Christoph Ortner. Atomic cluster expansion: Completeness, efficiency and stability. *Journal of Computational Physics*, 454:110946, 2022.

[80] Xiangzhe Kong, Wenbing Huang, Zhixing Tan, and Yang Liu. Molecule generation by principal subgraph mining and assembling. *Advances in Neural Information Processing Systems*, 35:2550–2563, 2022.

[81] Xiangzhe Kong, Wenbing Huang, and Yang Liu. Generalist equivariant transformer towards 3d molecular interaction learning. In *Forty-first International Conference on Machine Learning*, 2023.

[82] Xiangzhe Kong, Wenbing Huang, and Yang Liu. Full-atom peptide design with geometric latent diffusion. *Advances in Neural Information Processing Systems*, 2024.

[83] Miltiadis Kofinas, Naveen Nagaraja, and Efstratios Gavves. Roto-translated local coordinate frames for interacting dynamical systems. *Advances in Neural Information Processing Systems*, 34:6417–6429, 2021.

[84] Alexandre Agm Duval, Victor Schmidt, Alex Hernández-Garcıa, Santiago Miret, Fragkiskos D Malliaros, Yoshua Bengio, and David Rolnick. Faenet: Frame averaging equivariant gnn for materials modeling. In *International Conference on Machine Learning*, pages 9013–9033. PMLR, 2023.

[85] Yi-Lun Liao, Brandon M Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations. In *The Twelfth International Conference on Learning Representations*, 2024.

[86] Ziqiao Meng, Liang Zeng, Zixing Song, Tingyang Xu, Peilin Zhao, and Irwin King. Towards geometric normalization techniques in se(3) equivariant graph neural networks for physical dynamics simulations. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 5981–5989, 2024.

[87] Zongzhao Li, Jiacheng Cen, Bing Su, Tingyang Xu, Yu Rong, Deli Zhao, and Wenbing Huang. Large language-geometry model: When LLM meets equivariance. In *Forty-second International Conference on Machine Learning*, 2025.

[88] Gengmo Zhou, Zhifeng Gao, Qiankun Ding, Hang Zheng, Hongteng Xu, Zhewei Wei, Linfeng Zhang, and Guolin Ke. Uni-mol: A universal 3d molecular representation learning framework. In *The Eleventh International Conference on Learning Representations*, 2023.

[89] Weiliang Luo, Gengmo Zhou, Zhengdan Zhu, Yannan Yuan, Guolin Ke, Zhewei Wei, Zhifeng Gao, and Hang Zheng. Bridging machine learning and thermodynamics for accurate p k a prediction. *JACS Au*, 2024.

[90] Gengmo Zhou, Zhen Wang, Feng Yu, Guolin Ke, Zhewei Wei, and Zhifeng Gao. S-molsearch: 3d semi-supervised contrastive learning for bioactive molecule search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[91] Fanmeng Wang, Hongteng Xu, Xi Chen, Shuqi Lu, Yuqing Deng, and Wenbing Huang. Mperformer: An se (3) transformer-based molecular perceptron. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 2512–2522, 2023.

[92] Fanmeng Wang, Wentao Guo, Minjie Cheng, Shen Yuan, Hongteng Xu, and Zhifeng Gao. Mmpolymer: A multimodal multitask pretraining framework for polymer property prediction. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, 2024.

[93] Johannes Klicpera, Janek Groß, and Stephan Günnemann. Directional message passing for molecular graphs. In *International Conference on Learning Representations*, 2020.

[94] Limei Wang, Yi Liu, Yuchao Lin, Haoran Liu, and Shuiwang Ji. Comenet: Towards complete and efficient message passing for 3d molecular graphs. *Advances in Neural Information Processing Systems*, 35:650–664, 2022.

[95] Angxiao Yue, Dixin Luo, and Hongteng Xu. A plug-and-play quaternion message-passing module for molecular conformation representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 16633–16641, 2024.

[96] Angxiao Yue, Zichong Wang, and Hongteng Xu. Reqflow: Rectified quaternion flow for efficient and high-quality protein backbone generation. In *Forty-second International Conference on Machine Learning*, 2025.

[97] Yang Zhang, Wenbing Huang, Zhewei Wei, Ye Yuan, and Zhaohan Ding. Equipocket: an e (3)-equivariant geometric graph neural network for ligand binding site prediction. In *Forty-first International Conference on Machine Learning*, 2024.

[98] Rui Jiao, Jiaqi Han, Wenbing Huang, Yu Rong, and Yang Liu. Energy-motivated equivariant pretraining for 3d molecular graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8096–8104, 2023.

[99] Ziyang Yu, Wenbing Huang, and Yang Liu. Force-guided bridge matching for full-atom time-coarsened dynamics of peptides. *arXiv preprint arXiv:2408.15126*, 2024.

[100] Rui Jiao, Wenbing Huang, Peijia Lin, Jiaqi Han, Pin Chen, Yutong Lu, and Yang Liu. Crystal structure prediction by joint equivariant diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[101] Rui jiao, Xiangzhe Kong, Wenbing Huang, and Yang Liu. 3d structure prediction of atomic systems with flow-based direct preference optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[102] Xiangzhe Kong, Wenbing Huang, and Yang Liu. Conditional antibody design as 3d equivariant graph translation. In *The Eleventh International Conference on Learning Representations*, 2023.

[103] Xiangzhe Kong, Wenbing Huang, and Yang Liu. End-to-end full-atom antibody design. In *Proceedings of the 40th International Conference on Machine Learning*, pages 17409–17429, 2023.

[104] Daniel Widdowson and Vitaliy Kurlin. Recognizing rigid patterns of unlabeled point clouds by complete and continuous isometry invariants with no false negatives and no false positives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1275–1284, 2023.

[105] NA Court. Notes on the orthocentric tetrahedron. *The American Mathematical Monthly*, 41(8):499–502, 1934.

[106] Elias M Stein and Rami Shakarchi. *Fourier analysis: an introduction*, volume 1. Princeton University Press, 2011.

[107] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018.

[108] Sarp Aykent and Tian Xia. Savenet: A scalable vector network for enhanced molecular representation learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Contents of Appendix

A Theoretical Analysis and Proofs

A.1 Detailed Preliminaries

This section introduces the equivariant representation and the CG tensor product with inversion, along with the symmetric graph structure and its related properties.

A.1.1 Equivariance

Let $\mathbb{X}$ and $\mathbb{Y}$ be the input and output vector spaces, respectively. A function $\phi : \mathbb{X} \to \mathbb{Y}$ is called *equivariant* with respect to group $\mathfrak{G}$ if

$$\forall \mathfrak{g} \in \mathfrak{G}, \phi(\rho_{\mathbb{X}}(\mathfrak{g})\boldsymbol{x}) = \rho_{\mathbb{Y}}(\mathfrak{g})\phi(\boldsymbol{x}), \tag{8}$$

where $\rho_{\mathbb{X}}$ and $\rho_{\mathbb{Y}}$ are the group representations in the input and output spaces, respectively.

Since we can permit translation invariance by simply translating the center of all coordinates to the origin, we only discuss equivariance with respect to $\mathrm{O}(3)$ in our paper. Specifically, the group $\mathrm{O}(3)$ consists of rotation and inversion, implying $\mathrm{O}(3) = \mathrm{SO}(3) \times C_i$, where $\mathrm{SO}(3)$ is the rotation group and $C_i = \{\mathfrak{e}, \mathfrak{i}\}$ denotes the inversion group with $\mathfrak{e}$ representing the identity and $\mathfrak{i}$ representing the inversion. In the current literature, researchers typically utilize irreducible representations (*i.e.* Wigner-D matrix $\boldsymbol{D}^{(l)}(\mathfrak{r}) \in \mathbb{R}^{(2l+1)\times(2l+1)}$ with l denoting the degree) to represent $\mathrm{SO}(3)$ and parities $p \in \{\pm 1\}$ to represent the inversion group C_i as follows:

$$\rho^{(l,p)}(\mathfrak{r}\mathfrak{m}) \coloneqq \sigma^{(p)}(\mathfrak{m})\boldsymbol{D}^{(l)}(\mathfrak{r}), \tag{9}$$

where $\sigma^{(p)}(\mathfrak{m}) = p$ if $\mathfrak{m} = \mathfrak{i}$, while $\sigma^{(p)}(\mathfrak{m}) = 1$ if $\mathfrak{m} = \mathfrak{e}$. When considering only rotations, a $(2l+1)$-dimensional variable that can be rotated by the lth-degree Wigner-D matrices is referred to as an lth-degree steerable feature. When further involving the concept of inversion, we adopt the notation presented in e3nn [61]. Specifically, an inversion-invariant lth-degree steerable feature (*i.e.*, $p = 1$) is referred to as an $l\mathtt{e}$-type, while an inversion-equivariant feature (*i.e.*, $p = -1$) is designated as an $l\mathtt{o}$-type. These designations correspond to *even parity* and *odd parity*, respectively.

Example A.1 (Spherical Harmonics)**.** Spherical harmonics $Y^{(l)}(\vec{\boldsymbol{x}}) = [Y_m^{(l)}(\vec{\boldsymbol{x}})]_{m=-l}^{l}$ are lth-degree steerable features, *i.e.* $Y^{(l)}(\boldsymbol{R}_{\mathfrak{r}}\vec{\boldsymbol{x}}) = \boldsymbol{D}^{(l)}(\mathfrak{r})Y^{(l)}(\vec{\boldsymbol{x}})$, and the parity is given as $p = (-1)^l$. It means spherical harmonics are $l\mathtt{o}$-type when l is odd and $l\mathtt{e}$-type when l is even. According to [60], spherical harmonics offer a *complete* set of function bases of $\mathrm{SO}(3)$-equivariant functions.

Another example about common physical quantities is presented in Example A.2.

Example A.2 (Common Physical Quantities)**.** Scalars (*e.g.* mass, charge and energy) are $0\mathtt{e}$-type (invariant to both rotation and inversion). Pseudo-scalar (*e.g.* magnetic charge and magnetic flux) are $0\mathtt{o}$-type (invariant to rotation but change signs under inversion). Vectors (*e.g.* coordinate and velocity) are $1\mathtt{o}$-type, while pseudo-vectors (*e.g.* torque and angular momentum) are $1\mathtt{e}$-type.

Example A.3 (Common Invariant features)**.** Due to the various advantages of scalars, there are a large number of construction methods, which we briefly list here:

1. Purely mathematical representation: tensor product (spherical-scalarization) [48–50], topological characteristics [75–77], cluster expansion basis [78; 79], subgraph blocks [80–82], frames [56; 83; 84], normalization operators [85; 86], LLM-based information [87];
2. Features based on physical and biochemical prior knowledge: distance matrix [40; 88–92], chemical bond length, angle and dihedral angle [93–97], force [98; 99], fractional coordinates [21; 100; 101], canonical ordering [102; 103].

A.1.2 Clebsch–Gordan (CG) tensor product

Tensor product $\otimes(\cdot,\cdot)$ is a common operator to describe interaction between steerable features. Given an l_1th-degree steerable feature $\tilde{\boldsymbol{v}}^{(l_1)}$ and an l_2th-degree steerable feature $\tilde{\boldsymbol{v}}^{(l_2)}$, calculating their tensor product yields a new feature $\tilde{\boldsymbol{v}}^{(l_1)} \otimes \tilde{\boldsymbol{v}}^{(l_2)}$ with its group representation as $\boldsymbol{D}^{(l_1)}(\mathfrak{r}) \otimes \boldsymbol{D}^{(l_2)}(\mathfrak{r})$. The Clebsch-Gordan rule reveals that any tensor product of irreducible representations can be represented as the direct sum of the irreducible representations. Let $\mathbb{V}^{(l_1)}$ and $\mathbb{V}^{(l_2)}$ denote the irreducible

representation space of l_1th-degree and l_2th-degree, respectively. Then, we have the following relation:

$$\mathbb{V}^{(l_1)} \otimes \mathbb{V}^{(l_2)} = \bigoplus_{l=|l_1-l_2|}^{l_1+l_2} \mathbb{V}^{(l)}. \tag{10}$$

This further demonstrates that the group representation matrix $\boldsymbol{D}^{(l_1)}(\mathfrak{r}) \otimes \boldsymbol{D}^{(l_2)}(\mathfrak{r})$ is similar to the matrix $\bigoplus_{l=|l_1-l_2|}^{l_1+l_2} \boldsymbol{D}^{(l)}(\mathfrak{r})$. In general, the lth-degree component from the tensor product $\tilde{\boldsymbol{v}}^{(l_1)} \otimes \tilde{\boldsymbol{v}}^{(l_2)}$ can be expressed as

$$v_m^{(l)} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} Q_{(l_1,m_1),(l_2,m_2)}^{(l,m)} v_{m_1}^{(l_1)} v_{m_2}^{(l_2)}, \tag{11}$$

where $Q_{(l_1,m_1),(l_2,m_2)}^{(l,m)}$ is called the *Clebsch–Gordan coefficients*. To simplify writing, when specifying the degree of the input and output, we directly abbreviate the above formula to $\tilde{\boldsymbol{v}}^{(l)} = \tilde{\boldsymbol{v}}^{(l_1)} \otimes_{\text{cg}} \tilde{\boldsymbol{v}}^{(l_2)}$. We introduce the decomposition of the moment of inertia in Example A.4 to enhance understanding.

It is straightforward to extend the tensor product to incorporate inversion. When performing the tensor product of two steerable features—one exhibiting odd parity and the other exhibiting even parity—the resulting feature will necessarily exhibit odd parity. Conversely, if both features share the same parity, the result will be of even parity. For simplicity, we will omit discussions related to inversion in the subsequent analyses.

Example A.4 (Decomposition of Reducible Representations)**.** For the moment of inertia $\text{vec}(\overleftrightarrow{\boldsymbol{I}}) = \sum_i (\|\vec{\boldsymbol{x}}_i\|^2 \cdot \mathbf{1}_9 - \vec{\boldsymbol{x}}_i \otimes \vec{\boldsymbol{x}}_i) \in \mathbb{R}^9$. The first item is a scalar (`0e`) of 9 channels denoted as `9x0e`, while the corresponding group representation of the second item $\sum_i \vec{\boldsymbol{x}}_i \otimes \vec{\boldsymbol{x}}_i$ is $\boldsymbol{R}_{\mathfrak{r}} \otimes \boldsymbol{R}_{\mathfrak{r}}$ (the inversion are canceled). According to the Clebsch–Gordan rule, this representation is similar to $\bigoplus_{l=0}^{2} \boldsymbol{D}^{(l)}(\mathfrak{r})$, which can be transformed through a specific coefficient matrix, and $\sum_i \vec{\boldsymbol{x}}_i \otimes \vec{\boldsymbol{x}}_i$ can be decomposed into irreducible features of three types: `0e`, `1e`, `2e`.[4]

A.1.3 Discussion on Symmetric Graphs

A symmetric graph is a special kind of geometric graph. [49] points out that in a symmetric graph, steerable features of a particular degree will degenerate, so we list this case separately for analysis.

Definition A.5 (Symmetric Graph)**.** *A geometric graph $\mathcal{G}$ is called a symmetric graph, if there exists a finite and nontrivial subgroup $\mathfrak{H} \leq \mathrm{O}(3), \mathfrak{H} \neq \{\mathfrak{e}\}$, satisfying that $\forall \mathfrak{h} \in \mathfrak{H}, \mathfrak{h} \cdot \mathcal{G} = \mathcal{G}$. All subgroups making $\mathcal{G}$ symmetric yields a set $\mathbb{H}(\mathcal{G})$, and all geometric graphs that are symmetric* w.r.t. $\mathfrak{H}$ *constitute a set denoted as $\mathbb{G}(\mathfrak{H})$. Moreover, all other graphs are referred to as* asymmetric (geometric) graphs.

It is straightforward to conclude that an asymmetric geometric graph must contain non-coplanar nodes; otherwise, if all the nodes lie in the same plane, the graph would coincide with itself after reflection across that plane. Therefore, the presence of non-coplanar nodes is a necessary condition for the asymmetry of the geometric graph.

Lemma A.6 (Image Space of Functions on Symmetric Geometric Graphs)**.** *Given a symmetric graph $\mathcal{G}$, $\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}) \in \bigcap_{\mathfrak{H} \in \mathbb{H}(\mathcal{G})} [\boldsymbol{I}_{2l+1} - \rho^{(l)}(\mathfrak{H})]^{\perp}$, where $\rho^{(l)}(\mathfrak{H}) = \frac{1}{|\mathfrak{H}|} \sum_{\mathfrak{h} \in \mathfrak{H}} \rho^{(l)}(\mathfrak{h})$ denotes the group averaging.*

Here, we strengthen the conclusion of [49]. Below, we give an example that makes the steerable features confined to a subspace instead of degenerating into a zero vector.

Example A.7. Consider a cone with a square base, defined by the set $\mathcal{G} = \{(\pm 1, 0, -1), (0, \pm 1, -1), (0, 0, 4)\}$. For $l = 1$, it is straightforward to find that $\mathbb{F}^{(1)}(\mathcal{G}) = \{(0, 0, z)\} \subset \mathbb{R}^3$, and constructing a $\mathbb{F}^{(1)}(\mathcal{G})$-full-rank function is relatively easy. However, for cases where $l > 1$, while the analysis can still be performed using the orthogonal complement of the group average, constructing a $\mathbb{F}^{(l)}(\mathcal{G})$-full-rank function becomes significantly more challenging. We propose using tensor-product-based models, such as TFN [33], to construct a basis set. These models, despite their computational complexity, have been proven to be complete. Once constructed, this basis set can be applied with our method to achieve the desired result.

[4] Note that since it is a symmetric tensor, the components of `1e` are actually zero.

A.2 Connection between graph-level and subgraph-level functions

In practice, the focus of many problems lies in some local features, such as node-level or edge-level features, which we collectively refer to as subgraph-level. In this section, we will show the connection between graph-level functions and subgraph-level functions.

In fact, the local features also require the message from the whole geometric graph $\mathcal{G}$, thus for a subgraph $\mathcal{G}_{\text{sub}} \subset \mathcal{G}$, the lth-degree steerable feature could be denoted as $\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}_{\text{sub}}, \mathcal{G})$. Moreover, we denote $\mathbb{F}^{(l)}(\mathcal{G}_{\text{sub}}, \mathcal{G}) \coloneqq \{\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}_{\text{sub}}, \mathcal{G}) | \tilde{\boldsymbol{f}}^{(l)} \in \mathbb{F}^{(l)}\} \subset \mathbb{R}^{2l+1}$. Note that the whole graph is also a subgraph, $\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G})$ and $\mathbb{F}^{(l)}(\mathcal{G})$ are actually the simplified version of $\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}, \mathcal{G})$ and $\mathbb{F}^{(l)}(\mathcal{G}, \mathcal{G})$.

Lemma A.8. *Given a geometric graph $\mathcal{G}$, where $\mathcal{G}_{sub} \subset \mathcal{G}$ is a subgraph (*e.g. *node, edge), then we have $\mathbb{F}^{(l)}(\mathcal{G}) \subset \mathbb{F}^{(l)}(\mathcal{G}_{sub}, \mathcal{G})$.*

Proof. Note that for the subgraph-level function $\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}_{\text{sub}}, \mathcal{G})$, let the function ignore the previous input $\mathcal{G}_{\text{sub}}$. Therefore, for any graph-level function, there is always a subgraph-level function that is equal to it. □

Lemma A.9. *Given a geometric graph $\mathcal{G}$, where $\mathcal{G}_{sub} \subset \mathcal{G}$ is a subgraph (*e.g. *node, edge). We have $\mathbb{F}(\mathcal{G}_{sub}, \mathcal{G}) = \mathbb{R}^{2l+1}$, if $\mathbb{F}(\mathcal{G}) = \mathbb{R}^{2l+1}$.*

Proof. It is obvious that $\mathbb{F}(\mathcal{G}_{\text{sub}}, \mathcal{G}) \subset \mathbb{R}^{2l+1}$. And we have $\mathbb{R}^{2l+1} = \mathbb{F}(\mathcal{G}) \subset \mathbb{F}(\mathcal{G}_{\text{sub}}, \mathcal{G})$, thus $\mathbb{F}(\mathcal{G}_{\text{sub}}, \mathcal{G}) = \mathbb{F}(\mathcal{G}) = \mathbb{R}^{2l+1}$. □

Proposition A.10. *Given a geometric graph $\mathcal{G}$, where $\mathcal{G}_{sub} \subset \mathcal{G}$ is a subgraph (*e.g. *node, edge), suppose there is a matrix $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G}_{sub}, \mathcal{G})$ with C channels of lth-degree steerable features denoted as $\tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G}_{sub}, \mathcal{G})$ satisfying $\operatorname{span} \tilde{\boldsymbol{V}}^{(l)}(\mathcal{G}_{sub}, \mathcal{G}) = \mathbb{F}^{(l)}(\mathcal{G}_{sub}, \mathcal{G})$. Then for any lth-degree steerable function $\tilde{\boldsymbol{t}}^{(l)} \in \mathbb{F}^{(l)}$, there exists $\tilde{\boldsymbol{w}}^{(0)}(\mathcal{G}_{sub}, \mathcal{G}) \coloneqq [\tilde{w}_c^{(0)}(\mathcal{G}_{sub}, \mathcal{G})]_{c=1}^{C}$ with C-channel output scalars, such that*

$$\tilde{\boldsymbol{t}}^{(l)}(\mathcal{G}_{sub}, \mathcal{G}) = \textstyle\sum_{c=1}^{C} \tilde{w}_c^{(0)}(\mathcal{G}_{sub}, \mathcal{G}) \tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G}_{sub}, \mathcal{G}). \tag{12}$$

Proof. Note that the proof of Theorem 3.2 does not depend on the input form of the function, so the generalization from Theorem 3.2 to this proposition still holds. □

Proposition A.11. *Given a geometric graph $\mathcal{G}$, where $\mathcal{G}_{sub} \subset \mathcal{G}$ is a subgraph (*e.g. *node, edge) and $\mathbb{F}(\mathcal{G}_{sub}, \mathcal{G}) = \mathbb{F}(\mathcal{G})$. Suppose there is a matrix $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ with C channels of lth-degree steerable features denoted as $\tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G})$ satisfying $\operatorname{span} \tilde{\boldsymbol{V}}^{(l)}(\mathcal{G}) = \mathbb{F}^{(l)}(\mathcal{G})$. Then for any lth-degree steerable function $\tilde{\boldsymbol{t}}^{(l)} \in \mathbb{F}^{(l)}$, there exists $\tilde{\boldsymbol{w}}^{(0)}(\mathcal{G}_{sub}, \mathcal{G}) \coloneqq [\tilde{w}_c^{(0)}(\mathcal{G}_{sub}, \mathcal{G})]_{c=1}^{C}$ with C-channel output scalars, such that*

$$\tilde{\boldsymbol{t}}^{(l)}(\mathcal{G}_{sub}, \mathcal{G}) = \textstyle\sum_{c=1}^{C} \tilde{w}_c^{(0)}(\mathcal{G}_{sub}, \mathcal{G}) \tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G}). \tag{13}$$

Proof. Note that $\tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G}) \in \mathbb{F}^{(l)}(\mathcal{G}) \subset \mathbb{F}^{(l)}(\mathcal{G}_{\text{sub}}, \mathcal{G})$, so this proposition still holds. □

Note that for invariant functions, if we can obtain the canonical form at the graph-level, then we can naturally obtain the canonical form at the subgraph-level. Therefore, the discussion of subgraph-level and graph-level is consistent. To simplify the writing, only the graph-level case will be discussed in the following.

A.3 The full form of dynamic method

Before giving a complete theoretical analysis, let us review Theorem 3.2 as follows.

Theorem 3.2 (Dynamic Method)**.** *Given a geometric graph $\mathcal{G}$, suppose there is a matrix $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ with C channels of lth-degree steerable features denoted as $\tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G})$ satisfying $\operatorname{span}(\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})) = \mathbb{F}^{(l)}(\mathcal{G}) \coloneqq \{\tilde{\boldsymbol{f}}^{(l)}(\mathcal{G}) \mid \tilde{\boldsymbol{f}}^{(l)} \in \mathbb{F}^{(l)}\} \subset \mathbb{R}^{2l+1}$. Then for any lth-degree steerable function $\tilde{\boldsymbol{\phi}}^{(l)} \in \mathbb{F}^{(l)}$, there always exists $\tilde{\boldsymbol{w}}^{(0)}(\mathcal{G}) \coloneqq [\tilde{w}_c^{(0)}(\mathcal{G})]_{c=1}^{C}$ with C-channel output scalars, such that*

$$\tilde{\boldsymbol{\phi}}^{(l)}(\mathcal{G}) = \textstyle\sum_{c=1}^{C} \tilde{w}_c^{(0)}(\mathcal{G}) \tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G}). \tag{4}$$

This form decomposes the problem of constructing a complete equivariant function into two components. However, determining the explicit forms of $\tilde{\boldsymbol{w}}^{(0)}(\mathcal{G})$ and $\tilde{\boldsymbol{V}}^{(l)}$ remains a highly nontrivial task. To provide insight into this construction, we present two corollaries of Theorem 3.2, corresponding to two special cases of the equivariant space $\mathbb{F}^{(l)}(\mathcal{G})$: one where it reduces to the trivial set consisting solely of the origin, $\theta := \{\mathbf{0}_{2l+1}\}$, and the other is the opposite, *i.e.* $\dim \mathbb{F}^{(l)}(\mathcal{G}) > 0$.

Proposition A.12. *Given a geometric graph $\mathcal{G}$ with $\mathbb{F}^{(l)}(\mathcal{G}) = \theta$, let $\mathbb{W}^{(0)} \subset \mathbb{F}^{(0)}$ denote all invariant functions that can be expressed by model, any lth-degree steerable function could be written into the form of Theorem 3.2, if $\mathbb{W}^{(0)}$ is not empty.*

Proof. Since $\mathbb{F}^{(l)}(\mathcal{G}) = \theta$, any equivariant output of degree l must be the zero vector. In particular, both the target output $\tilde{\boldsymbol{t}}^{(l)}(\mathcal{G})$ and each basis component $\tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G})$ in the decomposition of Theorem 3.2 are identically zero.

As a result, the decomposition in Theorem 3.2 holds trivially for any choice of invariant scalar function $\tilde{\boldsymbol{w}}^{(0)}(\mathcal{G}) \in \mathbb{W}^{(0)}$, because both sides of the equation reduce to the zero vector. Hence, any lth-degree steerable function admits the desired representation, as long as $\mathbb{W}^{(0)}$ is non-empty. □

Proposition A.13. *Given a geometric graph $\mathcal{G}$ with $\dim \mathbb{F}^{(l)}(\mathcal{G}) > 0$, let $\mathbb{W}^{(0)} \subset \mathbb{F}^{(0)}$ denote all invariant functions that can be expressed by model, any lth-degree steerable function could be written into the form of Theorem 3.2, iff $\mathbb{W}^{(0)} = \mathbb{F}^{(0)}$,* i.e. *the invariant model is complete.*

Proof. The sufficiency is evident. By Theorem 3.2, there exists an invariant function of the required form, and such a function is clearly contained in the space $\mathbb{F}^{(0)}$.

To prove necessity, assume by contradiction that $\mathbb{W}^{(0)} \subsetneq \mathbb{F}^{(0)}$ but that the decomposition in Theorem 3.2 still holds for all steerable functions. Let $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ be a set of linearly independent equivariant vectors that spans $\mathbb{F}^{(l)}(\mathcal{G})$, *i.e.*, a basis of $\mathbb{F}^{(l)}(\mathcal{G})$. Now, consider any $\tilde{w}^{(0)} \in \mathbb{F}^{(0)} \setminus \mathbb{W}^{(0)}$. Then the function $\tilde{\boldsymbol{t}}^{(l)}(\mathcal{G}) = \sum_c \tilde{w}^{(0)}(\mathcal{G})\tilde{\boldsymbol{v}}_c^{(l)}(\mathcal{G})$ is a valid steerable function in $\mathbb{F}^{(l)}(\mathcal{G})$, as it results from a linear combination of basis elements with coefficients in $\mathbb{F}^{(0)}$. However, since $\tilde{w}^{(0)}$ is not expressible by the model, the decomposition cannot be achieved using elements in $\mathbb{W}^{(0)}$, contradicting the assumption that the decomposition is always possible.

Therefore, the assumption must be false, and we conclude that $\mathbb{W}^{(0)} = \mathbb{F}^{(0)}$ is necessary for the representation to hold for all steerable functions. □

A.4 The construction of canonical form

In order to losslessly encode the information of a geometric graph, we define the concept of canonical forms in this section. Then, we give a method to construct canonical forms in polynomial time. Here, we assume that the geometric graph is fully connected and use a double-loop set comparison method to estimate the worst time complexity of the algorithm.

A.4.1 An algorithm for determining point cloud isomorphism

Here, we give Algo. 1 for determining whether two geometric point clouds are isomorphic and prove its correctness. It is noteworthy that this algorithm resembles that of [104], centered on the principle of four-point positioning in three-dimensional space. Our subsequent designs will leverage this principle to adapt it for use in geometric graph scenarios and neural network applications.

For two given point clouds, represented as $\vec{\boldsymbol{X}}^{(\mathcal{G})}$ and $\vec{\boldsymbol{X}}^{(\mathcal{H})}$, we recognize that if these point clouds are isomorphic, there must exist a local substructure (composed of at least four points, *i.e.* $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}}$ and $\vec{\boldsymbol{V}}_{\boldsymbol{\beta}}$) that is also isomorphic. It is worth noting that this is the isomorphism judgment between two point clouds of a certain size, so it is always of constant complexity.

According to the four-point positioning principle, the coordinates of any point $\vec{\boldsymbol{u}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{G})}$ or $\vec{\boldsymbol{v}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{H})}$ can be expressed as distance vectors, *i.e.* $\boldsymbol{d}_i^{(\mathcal{G})}$ or $\boldsymbol{d}_i^{(\mathcal{H})}$, relative to these four foundational points. Consequently, sets of coordinates $\vec{\boldsymbol{X}}^{(\mathcal{G})}, \vec{\boldsymbol{X}}^{(\mathcal{H})}$ can be transformed into sets of distance vectors $\mathbb{D}^{(\mathcal{G})}, \mathbb{D}^{(\mathcal{H})}$. When there is a set of isomorphic substructures with matching distance vector

Algorithm 1: Check *point clouds isomorphic* with time complexity $\mathcal{O}(N^6)$.

Data: Two point clouds $\vec{\boldsymbol{X}}^{(\mathcal{G})}$ and $\vec{\boldsymbol{X}}^{(\mathcal{H})}$, each containing N nodes.
Result: *True* or *False*, indicating whether the two point clouds are isomorphic.
// $\mathcal{O}(N^4)$, the worst case requires traversing all permutations.
Find an ordered tuple containing four non-coplanar points $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}} \leftarrow (\vec{\boldsymbol{u}}_{\alpha_i}^{(\mathcal{G})})_{i=1}^4$ in $\mathcal{G}$;
// $\mathcal{O}(N^4)$, the permutations of 4 elements from a set of size N.
for *all ordered subsets of* $\vec{\boldsymbol{X}}^{(\mathcal{H})}$ *containing four elements donated by* $\vec{\boldsymbol{V}}_{\boldsymbol{\beta}} \leftarrow (\vec{\boldsymbol{v}}_{\beta_i}^{(\mathcal{H})})_{i=1}^4$ **do**
  // $\mathcal{O}(1)$, comparison of two ordered sets of a certain size.
  if $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}}$ *and* $\vec{\boldsymbol{V}}_{\boldsymbol{\beta}}$ *is not point clouds isomorphic according to matching* $(\vec{\boldsymbol{u}}_{\alpha_i}^{(\mathcal{G})}, \vec{\boldsymbol{v}}_{\beta_i}^{(\mathcal{H})})$ **then**
    continue;
  end
  for $\vec{\boldsymbol{u}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{G})}$ **do**
    // $\mathcal{O}(N)$, get a 4-channel scalar vector.
    $\boldsymbol{d}_i^{(\mathcal{G})} \leftarrow (\|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_1}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_2}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_3}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_4}\|)$;
  end
  for $\vec{\boldsymbol{v}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{H})}$ **do**
    // $\mathcal{O}(N)$, get a 4-channel scalar vector.
    $\boldsymbol{d}_i^{(\mathcal{H})} \leftarrow (\|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_1}\|, \|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_2}\|, \|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_3}\|, \|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_4}\|)$;
  end
  // $\mathcal{O}(N)$, change the node set into scalar set.
  $\mathbb{D}^{(\mathcal{G})}, \mathbb{D}^{(\mathcal{H})} \leftarrow \texttt{set}([\boldsymbol{d}_i^{(\mathcal{G})}]_{i=1}^N), \texttt{set}([\boldsymbol{d}_i^{(\mathcal{H})}]_{i=1}^N)$;
  // $\mathcal{O}(N^2)$, the complexity of comparing two sets varies depending on the choice, such as double loop comparison.
  if $\mathbb{D}^{(\mathcal{G})} = \mathbb{D}^{(\mathcal{H})}$ **then**
    return True;
  end
end
return False;

sets, the point clouds are necessarily isomorphic. Conversely, if no isomorphic substructure can be identified or if the distance vector sets cannot be rendered equivalent for any potential isomorphic substructure, then the point clouds must not be isomorphic.

A.4.2 An algorithm for determining geometric graph isomorphism

Furthermore, we discuss how to design an algorithm to distinguish the isomorphism of two geometric graphs. Note that at this point, all node coordinates have been converted to distance vectors, so it is also easy to transform the edge set as follows.

A.4.3 A general canonical form

Now, we further give a method to construct the canonical form of a geometric graph. From Algos. 1 and 2, it is not difficult to see that the core of the algorithm is to transform a set of 3D coordinates into a set of distance vectors. Here we give Algo. 3 as follows.

To encode the set, we employ DeepSet [65] for encoding, which losslessly encodes set data. Here, we ignore the complexity of the neural network (because this is a function of the hidden layer dimension H), and only consider the complexity of the set size. It is not difficult to see from Eq. (14) that for a set of size M, the complexity of DeepSet is $\mathcal{O}(M)$.

Lemma A.14 (DeepSet)**.** *A set function could be always written into such form:*

$$f(\{x_i\}_{i=1}^N) = \rho(\textstyle\sum_{i=1}^N \phi(x_i)). \tag{14}$$

Algorithm 2: Check *geometric graph isomorphism* with time complexity $\mathcal{O}(N^8)$.

Data: Two geometric graphs $\mathcal{G}$ and $\mathcal{H}$, each containing N nodes.
Result: *True* or *False*, indicating whether the two point clouds are isomorphic.
`// O(N^4), the worst case requires traversing all permutations.`
Find an ordered tuple containing four non-coplanar points $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}} \leftarrow (\vec{\boldsymbol{u}}_{\alpha_i}^{(\mathcal{G})})_{i=1}^{4}$ in $\mathcal{G}$;
`// O(N^4), the permutations of 4 elements from a set of size N.`
for *all ordered subsets of* $\vec{\boldsymbol{X}}^{(\mathcal{H})}$ *containing four elements donated by* $\vec{\boldsymbol{V}}_{\boldsymbol{\beta}} \leftarrow (\vec{\boldsymbol{v}}_{\beta_i}^{(\mathcal{H})})_{i=1}^{4}$ **do**
  `// O(1), comparison of two ordered sets of a certain size.`
  if $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}}$ *and* $\vec{\boldsymbol{V}}_{\boldsymbol{\beta}}$ *is not point clouds isomorphic according to matching* $(\vec{\boldsymbol{u}}_{\alpha_i}^{(\mathcal{G})}, \vec{\boldsymbol{v}}_{\beta_i}^{(\mathcal{H})})$ **then**
    continue;
  end
  for $\vec{\boldsymbol{u}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{G})}$ **do**
    `// O(N), get a 4-channel scalar vector.`
    $\boldsymbol{d}_i^{(\mathcal{G})} \leftarrow (\|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_1}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_2}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_3}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_4}\|)$;
  end
  for $\vec{\boldsymbol{v}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{H})}$ **do**
    `// O(N), get a 4-channel scalar vector.`
    $\boldsymbol{d}_i^{(\mathcal{H})} \leftarrow (\|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_1}\|, \|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_2}\|, \|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_3}\|, \|\vec{\boldsymbol{v}}_i - \vec{\boldsymbol{v}}_{\beta_4}\|)$;
  end
  `// O(N + N^2), change the node set and edge set into scalar set.`
  $\mathbb{D}^{(\mathcal{G})}$, $\mathbb{D}^{(\mathcal{H})} \leftarrow$ `set`$([\boldsymbol{d}_i^{(\mathcal{G})}]_{i=1}^{N})$, `set`$([\boldsymbol{d}_i^{(\mathcal{H})}]_{i=1}^{N})$;
  $\mathbb{E}^{(\mathcal{G})}$, $\mathbb{E}^{(\mathcal{H})} \leftarrow$ `set`$([\boldsymbol{d}_i^{(\mathcal{G})}, \boldsymbol{d}_j^{(\mathcal{G})}, \boldsymbol{e}_{ij}^{(\mathcal{G})}]_{\langle i,j\rangle \in \mathcal{E}^{(\mathcal{G})}})$, `set`$([\boldsymbol{d}_i^{(\mathcal{H})}, \boldsymbol{d}_j^{(\mathcal{H})}, \boldsymbol{e}_{ij}^{(\mathcal{H})}]_{\langle i,j\rangle \in \mathcal{E}^{(\mathcal{H})}})$;
  `// O(N^2 + N^4), the complexity of comparing four sets (node-node, edge-edge) varies depending on the choice, such as double loop comparison.`
  if $\mathbb{D}^{(\mathcal{G})} = \mathbb{D}^{(\mathcal{H})}$ *and* $\mathbb{E}^{(\mathcal{G})} = \mathbb{E}^{(\mathcal{H})}$ **then**
    return `True`;
  end
end
return `False`;

Proof. This proof is a simplification of Theorem 7 in Deepset [65]. The necessity, that is, that this form satisfies the permutation invariance of sets is obvious, and its sufficiency is proved below. Consider an Nth degree polynomial $P(t) = \sum_{i=0}^{N} a_i t^i$, which satisfies the set $\{x_i\}_{i=1}^{N}$ of all the N roots of $P(t) = 0$. Thus, the set can be mapped to ordered polynomial coefficients. Vieta's theorem and Newton's identity show that the space of polynomial coefficients is consistent with the space of sum-of-power functions, so the universal approximation theorem of MLP can be used to prove that such a mapping can be found. □

Theorem 3.5 (General Canonical Form)**.** *Given any geometric graph $\mathcal{G}$, Algo. 3 provides a method to create canonical form with time complexity $\mathcal{O}(N^6)$.*

Proof. Due to the completeness of Deepset's encoding, Algo. 3 can naturally distinguish arbitrary geometric graphs. For a detailed time complexity analysis, see the comments in the algorithm block. □

A.4.4 A faster method to construct canonical form

In practical applications, it is impractical to traverse all ordered four-point sets because the complexity is too high. In order to reduce the complexity, a simple and intuitive way is to reduce the number of four-point sets. Here, a very clever change of mind is that we change the ordered four-point set from being selected from a geometric graph to being generated from a geometric graph. We call these generated ordered four-point sets virtual nodes. In order to ensure the correctness of the algorithm, we

Algorithm 3: A canonical form of geometric graphs.

Data: A geometric graph $\mathcal{G}$, and $\psi_{\text{node}}, \psi_{\text{edge}}, \psi_{\text{graph}}$ are DeepSet models.
Result: The canonical form $\Gamma \in \mathbb{R}^H$ of point clouds $\mathcal{G}$.
// $\mathcal{O}(N^4)$, traverse all permutations.
$\mathbb{T} \leftarrow \varnothing$;
for *any ordered set containing four non-coplanar points* $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}} \leftarrow \{\vec{\boldsymbol{u}}_{\alpha_i}\}_{i=1}^4$ *in* $\mathcal{G}$ **do**
  for $\vec{\boldsymbol{u}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{G})}$ **do**
    // $\mathcal{O}(N)$, get a 4-channel scalar vector.
    $\boldsymbol{d}_i \leftarrow (\|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_1}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_2}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_3}\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{u}}_{\alpha_4}\|)$;
  end
  // $\mathcal{O}(N + N^2)$, convert point sets and edge sets into scalar sets.
  $\mathbb{D} \leftarrow \texttt{set}([\boldsymbol{d}_i]_{i=1}^N)$, $\mathbb{E} \leftarrow \texttt{set}([\boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{e}_{ij}]_{\langle i,j\rangle \in \mathcal{E}})$;
  // $\mathcal{O}(1)$, decentralization of the four reference points.
  $\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}} \leftarrow (\boldsymbol{I}_{4\times 4} - \frac{1}{4}\mathbf{1}_{4\times 4})\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}}$;
  // $\mathcal{O}(N + N^2)$, get the embedding based on current four points.
  $\Gamma_{\boldsymbol{\alpha}} \leftarrow \texttt{concat}(\vec{\boldsymbol{U}}_{\boldsymbol{\alpha}}^\top \vec{\boldsymbol{U}}_{\boldsymbol{\alpha}}, \psi_{\text{node}}(\mathbb{D}), \psi_{\text{edge}}(\mathbb{E}))$;
  $\mathbb{T} \leftarrow \mathbb{T} \cup \{\Gamma_{\boldsymbol{\alpha}}\}$;
end
$\Gamma \longleftarrow \psi_{\text{graph}}(\mathbb{T})$;
return Γ;

need to use a $\mathrm{E}(3)$-equivariant method to implement the generation process. We call this generating function ζ, thereby simplifying Algo. 3 to Algo. 4.

Algorithm 4: A faster method to construct canonical form.

Data: A geometric graph $\mathcal{G}$, and $\psi_{\text{node}}, \psi_{\text{edge}}$ are DeepSet models.
Result: The canonical form $\Gamma \in \mathbb{R}^H$ of point clouds $\mathcal{G}$.
// Get four non-coplanar reference points via generation.
$\vec{\boldsymbol{V}} \leftarrow \zeta(\mathcal{G})$;
for $\vec{\boldsymbol{u}}_i \in \vec{\boldsymbol{X}}^{(\mathcal{G})}$ **do**
  // $\mathcal{O}(N)$, get a 4-channel scalar vector.
  $\boldsymbol{d}_i \leftarrow (\|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{v}}_1\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{v}}_2\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{v}}_3\|, \|\vec{\boldsymbol{u}}_i - \vec{\boldsymbol{v}}_4\|)$;
end
// $\mathcal{O}(N + N^2)$, convert point sets and edge sets into scalar sets.
$\mathbb{D} \leftarrow \texttt{set}([\boldsymbol{d}_i]_{i=1}^N)$, $\mathbb{E} \leftarrow \texttt{set}([\boldsymbol{d}_i, \boldsymbol{d}_j, \boldsymbol{e}_{ij}]_{\langle i,j\rangle \in \mathcal{E}})$;
// $\mathcal{O}(1)$, decentralization of the four reference points.
$\vec{\boldsymbol{V}} \leftarrow (\boldsymbol{I}_{4\times 4} - \frac{1}{4}\mathbf{1}_{4\times 4})\vec{\boldsymbol{V}}$;
// $\mathcal{O}(N + N^2)$, get the embedding based on current four points.
$\Gamma \leftarrow \texttt{concat}(\vec{\boldsymbol{V}}^\top \vec{\boldsymbol{V}}, \psi_{\text{node}}(\mathbb{D}), \psi_{\text{edge}}(\mathbb{E}))$;
return Γ;

It is worth noting that Algo. 4 is applicable only when the four virtual nodes generated by ζ are not coplanar. We summarize it as:

Theorem 3.6 (Faster Canonical Form)**.** *Given an* $\mathrm{E}(3)$*-equivariant function* ζ *that generates four non-coplanar points on* $\mathcal{G}$*, Algo. 4 is able to create canonical form with time complexity* $\mathcal{O}(N^2)$*.*

Proof. Let Γ be the function of Algo. 4, we only need to prove that $\mathcal{G} \cong \mathcal{H} \iff \Gamma(\mathcal{G}) = \Gamma(\mathcal{H})$. We first prove sufficiency, let $\mathcal{G} = \mathfrak{g} \cdot \mathcal{H}$ with $\mathfrak{g} = (\boldsymbol{O}, \vec{\boldsymbol{t}}) \in \mathrm{E}(3)$ be a group element. Since ζ is an $\mathrm{E}(3)$-equivariant function, we have $\vec{\boldsymbol{V}}^{(\mathcal{G})} = \zeta(\mathcal{G}) = \zeta(\mathfrak{g} \cdot \mathcal{H}) = \boldsymbol{O}\vec{\boldsymbol{V}}^{(\mathcal{G})} + \vec{\boldsymbol{t}} = \vec{\boldsymbol{V}}^{(\mathcal{H})}$. After decentralization, we have $\boldsymbol{O}\vec{\boldsymbol{V}}^{(\mathcal{G})} = \vec{\boldsymbol{V}}^{(\mathcal{H})} \implies (\vec{\boldsymbol{V}}^{(\mathcal{G})})^\top \vec{\boldsymbol{V}}^{(\mathcal{G})} = (\vec{\boldsymbol{V}}^{(\mathcal{H})})^\top \vec{\boldsymbol{V}}^{(\mathcal{H})}$. In addition, the encoding of nodes and edges depends on the relative distance to the virtual node and is therefore

invariant, *i.e.* $\mathbb{D}^{(\mathcal{G})} = \mathbb{D}^{(\mathcal{H})}, \mathbb{E}^{(\mathcal{G})} = \mathbb{E}^{(\mathcal{H})}$. Thus, we get $\Gamma(\mathcal{G}) = \Gamma(\mathcal{H})$. Next, we prove the necessity. Note that when we take the reference four points as $\zeta(\mathcal{G})$ and $\zeta(\mathcal{H})$, we can directly get $\mathcal{G} \cong \mathcal{H}$ in Algo. 2, thus completing the proof. □

Table 8: Comparison of time complexity.

	General Method (Algo. 3)		**Faster Method**	
	Algorithm Details	Time Complexity	Algorithm Details	Time Complexity
Possible chosen of VNs	Traverse all possible quadruplets ($\binom{N}{4}$ types).	$\mathcal{O}(N^4)$	Generate directly.	$\mathcal{O}(1)$
Construct VNs	Use real nodes as virtual nodes.	$\mathcal{O}(1)$	Aggregate by all nodes.	$\mathcal{O}(N)$
Get distance vector	Calculate N nodes to possible VNs.	$\mathcal{O}(N^4 \cdot N)$	Calculate N nodes to possible VNs.	$\mathcal{O}(1 \cdot N)$
Set of distance vector	Each chosen of VN prodive a set.	$\mathcal{O}(N^4)$	Each chosen of VN prodive a set.	$\mathcal{O}(1)$
Embed set with Deepset	N for node and N^2 for edge	$\mathcal{O}(N^4 \cdot (N + N^2))$	N for node and N^2 for edge	$\mathcal{O}(1 \cdot (N + N^2))$
Total	$\mathcal{O}(N^6)$		$\mathcal{O}(N^2)$	

A.5 Construction of full-rank basis set

In this section, we discuss the full-rank basis set, including its existence and construction method on asymmetric graphs.

Theorem 3.7 (Coloring on Asymmetric Graph)**.** *In an asymmetric geometric graph, each point can be assigned a unique color. This implies the existence of an* $\mathrm{E}(3)$ *invariant function that maps the features of the* i*th node to distinct values* $\boldsymbol{h}_i$*. Similarly, for directed edges* ij*, their features are mapped to distinct values* $\boldsymbol{e}_{ij}$*.*

Proof. The theorem can be described in the following formal language, given an asymmetric point cloud $\vec{\boldsymbol{X}} \in \mathbb{R}^{3\times N}$, and all function here discussed are node-level scalar function. This theorem means

$$\exists f : \vec{\boldsymbol{X}} \mapsto \boldsymbol{y} \in \mathbb{N}_+^N, \forall i \neq j, \boldsymbol{y}_i \neq \boldsymbol{y}_j. \tag{15}$$

First, we prove that it can be expressed as an equivalent proposition, which is

$$\forall i \neq j, \exists f : \vec{\boldsymbol{X}} \mapsto \boldsymbol{y} \in \mathbb{N}_+^N, \boldsymbol{y}_i \neq \boldsymbol{y}_j. \tag{16}$$

According to the logical relationship, it is obvious that Eq. (15) leads to Eq. (16). We will prove below that Eq. (16) can also lead to Eq. (15). Let $f^{ij} : \vec{\boldsymbol{X}} \mapsto \boldsymbol{y}^{ij}$ denote a function let $\boldsymbol{y}_i^{ij} \neq \boldsymbol{y}_j^{ij}$, then consider map $g : \vec{\boldsymbol{X}} \mapsto \boldsymbol{Y} \coloneqq \oplus_{i,j} \boldsymbol{y}^{ij}$. We can find that $\boldsymbol{Y}_i \neq \boldsymbol{Y}_j$, and notice each f^{ij} only uses finite colors, which means we can further assign a new color to each possible output of g, which is the mapping we want in Eq. (16).

Then we prove it by contradiction Eq. (16), assuming that the following proposition is true:

$$\exists i \neq j, \forall f : \vec{\boldsymbol{X}} \mapsto \boldsymbol{y} \in \mathbb{N}_+^N, \boldsymbol{y}_i = \boldsymbol{y}_j. \tag{17}$$

We now construct a contradiction, hoping to prove that Eq. (17) will lead to point cloud symmetry. Let the permutation matrix $\boldsymbol{P} \in S_N$ represent the exchange between node i and node j, so we can get:

$$f(\boldsymbol{P}\vec{\boldsymbol{X}}) = \boldsymbol{P}\boldsymbol{y} = \boldsymbol{y} = f(\vec{\boldsymbol{X}}). \tag{18}$$

Since f is an $\mathrm{E}(3)$-invarinat function, we guess $\boldsymbol{P}\vec{\boldsymbol{X}}$ will fall in set $\mathrm{E}(3) \cdot \vec{\boldsymbol{X}}$. And now we prove that $\boldsymbol{P}\vec{\boldsymbol{X}}$ must fall in such set. And the canonical form Γ is obvious an $\mathrm{E}(3)$-invarinat function and could distinguish whether two point clouds are isomorphic, which means $\boldsymbol{P}\vec{\boldsymbol{X}}$ must be point cloud isomorphic with $\vec{\boldsymbol{X}}$. And notice that f is a node-level function, which means

$$\vec{\boldsymbol{x}}_i = \mathfrak{g} \cdot \vec{\boldsymbol{x}}_j, \vec{\boldsymbol{x}}_j = \mathfrak{g} \cdot \vec{\boldsymbol{x}}_i.$$

And we have assumed there are no overlapping points, *i.e.* $\vec{\boldsymbol{x}}_i \neq \vec{\boldsymbol{x}}_j$, so $\mathfrak{g} \neq \mathfrak{e}$. And it shows that such a point cloud is symmetric, which constitutes a contradiction and the original proposition is proved. □

Theorem 3.8 (Existence of Full-Rank Basis Set)**.** *For any given asymmetric graph* $\mathcal{G}$*, an* $\mathbb{F}^{(l)}(\mathcal{G})$*-full-rank* $\tilde{\boldsymbol{V}}^{(l)}(\mathcal{G})$ *can always be constructed for any degree* l*.*

Proof. We give the proof via a constructive method. According to Theorem 3.7, there exists a mapping let all node be colored differently, and without loss of generality, we denote the color of node i as c_i. Now we construct function:

$$f_k(\vec{\boldsymbol{X}}) = \sum_{i=1}^{N} \delta_{k,c_i} \cdot \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_k, \tag{19}$$

It is obviously that such a funtion is E(3)-equvariant, and we and further combine f_k of different k, which can construct a full-rank $\mathbb{V}^{(1)}$ since the input point cloud is non-coplanar. □

We just give an existence proof here, which is still affected by the structure of the model itself and the form of message passing. We give an example in Example A.15.

A.6 Comparison of different coloring methods

Node coloring necessitates updating node features. Here, we propose two alternative solutions:

1. **Distance to Center**: A simple preliminary coloring can be achieved by calculating the distance $\|\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_c\|$ from each node to the coordinate center. While this method is straightforward, it lacks practical significance and may not perform well in real-world problems. Additionally, there are counterexamples, such as in Example A.15, where nodes cannot be distinguished.
2. **Tensor Product**: This method extends the previous approach. First, we construct a global feature $\sum_{i=1}^{N} Y^{(l)}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_c)$, which is then used to create the coloring through moments. While this method is more complex and incurs higher computational costs, it enhances expressive power.

Table 9: Different Color Methods. For the tensor product method, the set $\mathbb{L} = \{0, \ldots, L\}$ denotes all degrees and the learnable weights of tensor product are omitted.

Nothing ($\varnothing$)	**Distance to Center** ($\oplus$)	**Tensor Product** ($\otimes$)
	Calculation Formula	
$\varnothing$	$\boldsymbol{h}_i \leftarrow \boldsymbol{h}_i + \sigma(\|\vec{\boldsymbol{x}}_{ic}\|)$	$\tilde{\boldsymbol{h}}^{(\mathbb{L})}_{\text{global}} \leftarrow \sum_{i=1}^{N} \sigma(\|\vec{\boldsymbol{x}}_{ic}\|) \cdot Y^{(\mathbb{L})}(\vec{\boldsymbol{x}}_{ic})$ $\boldsymbol{h}_i \leftarrow \boldsymbol{h}_i + \tilde{\boldsymbol{h}}^{(\mathbb{L})}_{\text{global}} \otimes_{\text{cg}} \cdots \otimes_{\text{cg}} Y^{(\mathbb{L})}(\vec{\boldsymbol{x}}_{ic})$
	Scenarios to Avoid	
Sparse Graphs	Counterexample like Fig. 4(c)	Not Founded

Below, we present two examples to illustrate potential issues without coloring.

Example A.15 (Fig. 4(c))**.** We use the center pooling method to construct such an example. First, we ensure that the center of gravity of the image is at the origin, and secondly, make it asymmetric. The construction here requires a basic point cloud:

$$\vec{\boldsymbol{X}}_0 = \left\{ (-1, 0, 0), \left(\frac{1}{3}, \pm\frac{2\sqrt{2}}{3}, 0 \right), \left(\frac{1}{6}, \pm\frac{\sqrt{35}}{6}, 0 \right) \right\}.$$

All nodes in $\vec{\boldsymbol{X}}_0$ are in the unit circle, which means the distance between each node and the center of mass is the same. And the final point could is

$$\vec{\boldsymbol{X}} = \{\vec{\boldsymbol{X}}_0 \boldsymbol{R}_i\},$$

where we sample some random rotation matrix $\boldsymbol{R}_i$. And then it is obviously only using global pooling will lead all virtual node in the center of mass if we do not use any edge to color this nodes.

However, it could also be solve by high-degree steerable feature. The global steerable features can construct different virtual nodes with tensor product method in Table 9.

Example A.16 (Degeneration in Uncolored Models)**.** There are specific scenarios in which message passing on uncolored geometric graphs can result in degenerate phenomena. A particularly illustrative example is found in geometric graphs where all node features are identical and all edges are bidirectional. Considering a single-layer EGNN as an example, it may degenerate because the

condition $\boldsymbol{h}_i = \boldsymbol{h}_j$ implies $\boldsymbol{m}_{ij} = \boldsymbol{m}_{ji}$. Consequently, we have $\sum_{\langle i,j\rangle\in\mathcal{E}} \varphi_{\vec{\boldsymbol{x}}}(\boldsymbol{m}_{ij}) \cdot \vec{\boldsymbol{x}}_{ij} = \vec{\mathbf{0}}_3$. As a result, Eq. (7) can only yield the coordinate center $\vec{\boldsymbol{x}}_c$. This phenomenon can be observed in the tetrahedron center experiment in § 4, where besides EGNN, advanced models such as HEGNN [49], TFN [33], and MACE [35] also degenerate.

B Experiment Details

Our code is available at https://github.com/GLAD-RUC/Uni-EGNN.

B.1 Expressivity Tests

B.1.1 Completeness Test

In this section, we provide additional details that were omitted in § 4.1, focusing on the following points:

1. The 4-body non-chiral counterexample implemented by the GWL-test (Fig. 2(f) in IASR-test) and the 4-body chiral counterexample (Fig. 2(e) in IASR-test) exhibit a significant issue: the geometric graphs presented by [32] are geometrically isomorphic.
2. Details regarding data construction and model design for the chirality test, which primarily include the coloring strategy and specifics of the construction of `0o`-type features.

Metric. Referring to the settings of the GWL-test [32], we utilize the average accuracy across ten tests as our evaluation metric. The detailed experimental design is based on the implementation of the GWL-test [32], available in the code repository[5]. In this setup, a single GNN layer is employed to encode the geometric graph, followed by a simple classifier that predicts the label. If a GNN fails to distinguish between two geometric graphs, the output embeddings will be identical, resulting in one graph being classified correctly while the other is misclassified, leading to an accuracy of 50%. Conversely, an accuracy rate exceeding 50% indicates that the GNN can generate distinct embeddings and successfully differentiate between the two geometric graphs.

Additionally, the accuracy of the Basic model in Table 2 does not reach 100% due to a limited number of training epochs for the classifier (set to 100). Increasing the number of epochs (e.g., to 200) can achieve 100% classification accuracy. Notably, the EGNN model, with the same settings, also achieves 100% accuracy. We speculate that EGNN improves the embedding of geometric graphs during the message passing process.

Results of k-chain test. This experiment comes from GWL-test. The canonical form introduced by our model can easily obtain global information, so only one layer is needed to identify the geometric graphs differentiation task that other models require multiple layers.

Table 10: k-chain Test.

($k = \mathbf{4}$-chains)	Number of layers				
GNN Layer	$\lfloor \frac{k}{2} \rfloor$	$\lfloor \frac{k}{2} \rfloor + 1 = \mathbf{3}$	$\lfloor \frac{k}{2} \rfloor + 2$	$\lfloor \frac{k}{2} \rfloor + 3$	$\lfloor \frac{k}{2} \rfloor + 4$
EGNN	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	**100.0 ± 0.0**	**100.0 ± 0.0**
MACE	50.0 ± 0.0	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
$\text{Basic}_{\text{cpl}}$	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
$\text{SchNet}_{\text{cpl}}$	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
EGNN_{cpl}	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
$\text{GVP-GNN}_{\text{cpl}}$	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**
TFN_{cpl}	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**	**100.0 ± 0.0**

Geometrical isomorphism of 4-body counterexample. In the open source notebook provided by GWL-test [32], four tasks are presented, the first two of which are the 2-body and 3-body tasks tested

[5] https://github.com/chaitjo/geometric-gnn-dojo.

in Table 1. The other two tasks are termed the *4-body non-chiral counterexample* and the *4-body chiral counterexample*, both originating from IASR-test [69]. According to the experimental results provided in GWL-test, only $\text{MACE}_{\text{5-body}}$ successfully passed the 4-body non-chiral counterexample; moreover, the results for the 4-body chiral counterexample were not presented.

We successfully reproduced and cited the experimental results of the 2-body and 3-body from GWL-test [32]. However, we were unable to reproduce the results reported in the article despite multiple testing attempts, consistently achieving an accuracy of only 50%, indicating that $\text{MACE}_{\text{5-body}}$ failed this task. Furthermore, recent literature [77] also reported similar failures in the test. Upon thorough investigation, we discovered that the geometric graphs used in the two tests were geometrically isomorphic. We suspect that there may be a minor mistake in the GWL-test.

Specifically, the two graphs $\mathcal{G}_1, \mathcal{G}_2$ of the 4-body non-chiral counterexample are constructed as follows:

1. Consider three sub-graphs: $\mathcal{H}_1 = \{(3,2,-4),(0,2,5),(-3,2,-4)\}$, $\mathcal{H}_2 = \{(3,-2,-4),(0,-2,5),(-3,-2,-4)\}$, and $\mathcal{H}_3 = \{(0,5,0)\}$. $\boldsymbol{R}_y$ is a random matrix for rotation around the Oy-axis. Let $\boldsymbol{M}_x, \boldsymbol{M}_y$ denote the reflection about yOz-plane and zOx-plane.
2. $\mathcal{G}_1 = \mathcal{H}_1 \cup (\boldsymbol{R}_y \cdot \mathcal{H}_2) \cup \mathcal{H}_3$ and $\mathcal{G}_2 = \mathcal{H}_1 \cup (\boldsymbol{R}_y \cdot \mathcal{H}_2) \cup (\boldsymbol{M}_y \cdot \mathcal{H}_3)$.

Now we show $\mathcal{G}_1 \cong \mathcal{H}_2$, notice there are serval relation:

- Internal symmetry of geometric graph: $\mathcal{H}_i = \boldsymbol{M}_x \cdot \mathcal{H}_i (i = 1, 2)$, $\mathcal{H}_3 = \boldsymbol{R}_y \mathcal{H}_3$;
- Symmetry between geometric graphs: $\mathcal{H}_1 = \boldsymbol{M}_y \cdot \mathcal{H}_2$;
- Properties between geometric transformations: $\boldsymbol{M}_x^2 = \boldsymbol{M}_y^2 = I$, $\boldsymbol{R}_y \boldsymbol{M}_x = \boldsymbol{M}_x \boldsymbol{R}_y$, $\boldsymbol{R}_y \boldsymbol{M}_y = \boldsymbol{M}_y \boldsymbol{R}_y^\top$.

Then we have:

$$
\begin{aligned}
\mathcal{G}_1 =& \mathcal{H}_1 \cup (\boldsymbol{R}_y \cdot \mathcal{H}_2) \cup \mathcal{H}_3 = (\boldsymbol{M}_y \cdot \mathcal{H}_2) \cup (\boldsymbol{R}_y \boldsymbol{M}_y \cdot \mathcal{H}_1) \cup (\boldsymbol{M}_y \cdot \mathcal{H}_3) \\
=& (\boldsymbol{R}_y \boldsymbol{M}_y \cdot \mathcal{H}_1) \cup (\boldsymbol{M}_y \cdot \mathcal{H}_2) \cup (\boldsymbol{M}_y \cdot \mathcal{H}_3) \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top \cdot \mathcal{H}_1) \cup (\boldsymbol{M}_y \cdot \mathcal{H}_2) \cup (\boldsymbol{M}_y \cdot \mathcal{H}_3) \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top) \cdot [\mathcal{H}_1 \cup \boldsymbol{R}_y \mathcal{H}_2 \cup \boldsymbol{R}_y \mathcal{H}_3] \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top) \cdot [\mathcal{H}_1 \cup \boldsymbol{R}_y \mathcal{H}_2 \cup \mathcal{H}_3] \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top \boldsymbol{M}_x) \cdot [\boldsymbol{M}_x \mathcal{H}_1 \cup \boldsymbol{M}_x \boldsymbol{R}_y \mathcal{H}_2 \cup \boldsymbol{M}_x \mathcal{H}_3] \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top \boldsymbol{M}_x) \cdot [\mathcal{H}_1 \cup \boldsymbol{R}_y \boldsymbol{M}_x \mathcal{H}_2 \cup \boldsymbol{M}_x \mathcal{H}_3] \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top \boldsymbol{M}_x) \cdot [\mathcal{H}_1 \cup \boldsymbol{R}_y \mathcal{H}_2 \cup \boldsymbol{M}_x \mathcal{H}_3] \\
=& (\boldsymbol{M}_y \boldsymbol{R}_y^\top \boldsymbol{M}_x) \cdot \mathcal{G}_2 \\
=& (\boldsymbol{M}_x \boldsymbol{R}_y \boldsymbol{M}_y)^\top \cdot \mathcal{G}_2.
\end{aligned}
$$

Moreover, the 4-body chiral counterexample also contains two geometric graphs $\mathcal{G}_3, \mathcal{G}_4$, where $\mathcal{G}_3 \cong \mathcal{G}_4$:

1. Consider two sub-graphs: $\mathcal{H}_4 = \{(3,0,-4),(0,0,5),(-3,0,-4)\}$ and $\mathcal{H}_5 = \{(0,5,0)\}$. $\boldsymbol{M}_x, \boldsymbol{M}_y$ denote the reflection about yOz-plane and zOx-plane.
2. $\mathcal{G}_3 = \mathcal{H}_4 \cup \mathcal{H}_5$ and $\mathcal{G}_4 = \mathcal{H}_4 \cup (\boldsymbol{M}_y \mathcal{H}_5)$.

Now we show $\mathcal{G}_3 \cong \mathcal{H}_3$, notice there are serval relation:

- Internal symmetry of geometric graph: $\mathcal{H}_3 = \boldsymbol{M}_x \cdot \mathcal{H}_3$, $\mathcal{H}_3 = \boldsymbol{M}_y \mathcal{H}_3$ and $\mathcal{H}_4 = \boldsymbol{M}_x \mathcal{H}_4$;
- Properties between geometric transformations: $\boldsymbol{M}_x^2 = \boldsymbol{M}_y^2 = I$.

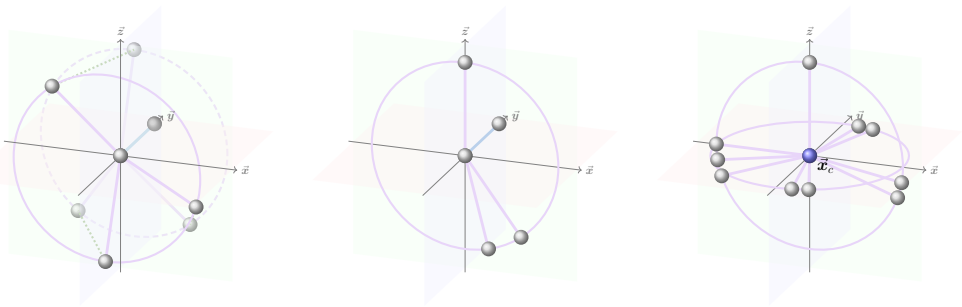
Then we have:

$$
\mathcal{G}_3 = \mathcal{H}_4 \cup \mathcal{H}_5 = (M_x M_y) \cdot [\mathcal{H}_4 \cup \boldsymbol{M}_y \mathcal{H}_5] = (M_x M_y) \cdot \mathcal{G}_4.
$$

B.1.2 Chirality Test

Details of the chirality test.

1. Fig. 4(a): Choose $\mathcal{G}_1$ from the 4-body non-chiral counterexample, and another geometric graph is $-\mathcal{G}_1$.
2. Fig. 4(b): Since $\mathcal{G}_3, \mathcal{G}_4$ are symmetric, we modify it to $\mathcal{G}_5 = \{(3,0,-4), \boldsymbol{R}_y \cdot (0,0,5), (-3,0,-4), (0,5,0)\}$, where $\boldsymbol{R}_y$ is a random matirx for rotation around the Oy-axis.
3. Fig. 4(c): Similar to Example A.15.



(a) Inversion (b) Mirror (c) Counterexample

Figure 4: Examples to test whether the model can recognize chirality. Figs. 4(a) and 4(b) are modified from GWL-test's implementation [36] of IASR-test [69], while Fig. 4(c) is designed by ourselves. Fig. 4(a): Test whether models could distinguish this geometric graph with itself after *inversion*. Fig. 4(b): Test whether models could distinguish this geometric graph with itself after *mirror* about the xOz-plane. Fig. 4(c): The task is same as Fig. 4(a), while all nodes are on the unit sphere, and the center point coincides with the center of the sphere. Such counterexample cannot produce separable virtual nodes by simple center distance coloring.

Details of construction of `0o`-type features. There are two ways to construct `0o`-type features:

1. **Determinant**: Calculate $\det([\vec{\boldsymbol{v}}_2 - \vec{\boldsymbol{v}}_1, \vec{\boldsymbol{v}}_3 - \vec{\boldsymbol{v}}_1, \vec{\boldsymbol{v}}_4 - \vec{\boldsymbol{v}}_1])$;
2. **Tensor Product**: We still begin by constructing $\sum_{i=1}^{N} Y^{(l)}(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_c)$, and employing moments, specifically at least 3rd-order moments, to derive `0o`-type features. This approach clearly serves as an extension of the previously established method.

B.2 Performance Tests

We conducted four experiments:

1. **Tetrahedron center prediction**, which entails predicting graph-level targets, *i.e.* Monge points, twelve-point centers, and incenters, given the coordinates of four points (with the same node features) of a given tetrahedron.
2. **5-body system** [72] involves predicting node-level targets, *i.e.* the coordinates of each charged particle at a specific moment in the future, given its initial coordinates and velocity.
3. **100-body system** [66], a generalization of the 5 body system, is used to test the performance of the model in large-scale systems.
4. **MD17 dataset** [68], containing eight molecules, is used to predict node-level targets, *i.e.* the future coordinates of each atom in the molecular trajectory. This dataset can be used to test the performance of the model on real datasets.
5. **Water-3D mini dataset** [66; 67], is used to test the performance of the model in large-scale systems and real datasets. Due to time constraints, we only evaluate a substantial subset (Water-3D-mini: 3,000/300/300 for train/val/test).

Table 11: Details of each dataset.

Dataset	Target	Type	#sample (train / valid / test)	Node Number	Edge connection
Tetrahedron	graph-level	Synthetic Dataset	500 / 2,000 / 2,000	4	`fully_connected`
5-body	node-level	Synthetic Dataset	5,000 / 2,000 / 2,000	5	`fully_connected`
100-body	node-level	Synthetic Dataset	5,000 / 2,000 / 2,000	100	`fully_connected`
MD17	node-level	Real-world Dataset	500 / 2,000 / 2,000	5-13	`distance_cutoff`
Water-3D mini	node-level	Real-world Dataset	3,000 / 300 / 300	∼8,000	`distance_cutoff`

B.2.1 Tetrahedron Center Prediction and N-body System

We conducted experiments on a single NVIDIA H20 GPU. In both experiments, we configured the batch size to 100, the learning rate to 5×10^{-4}, and the weight decay to 10^{-12}. For the tetrahedron center prediction task, the maximum number of training epochs was set to 300. In the case of the N-body system task, we implemented early stopping with a patience of 100 steps.

Metric. We use Mean Squared Error (MSE) to measure the accuracy of the prediction results in both experiments.

Tetrahedron Center Prediction. We generated our dataset using the calculation formulas for the Monge point $\vec{\boldsymbol{x}}_M$, twelve-point center $\vec{\boldsymbol{x}}_T$ and incenter $\vec{\boldsymbol{x}}_I$ from [105]. Given a tetrahedron defined by four points $\vec{\boldsymbol{x}}_O, \vec{\boldsymbol{x}}_A, \vec{\boldsymbol{x}}_B, \vec{\boldsymbol{x}}_C$, with the vectors $\vec{\boldsymbol{a}} = \vec{\boldsymbol{x}}_A - \vec{\boldsymbol{x}}_O, \vec{\boldsymbol{b}} = \vec{\boldsymbol{x}}_B - \vec{\boldsymbol{x}}_O, \vec{\boldsymbol{c}} = \vec{\boldsymbol{x}}_C - \vec{\boldsymbol{x}}_O$, we can compute the desired points as follows:

$$\begin{aligned}
\vec{\boldsymbol{x}}_M &= \vec{\boldsymbol{x}}_O + \frac{\vec{\boldsymbol{a}} \cdot (\vec{\boldsymbol{b}} + \vec{\boldsymbol{c}})(\vec{\boldsymbol{b}} \times \vec{\boldsymbol{c}}) + \vec{\boldsymbol{b}} \cdot (\vec{\boldsymbol{c}} + \vec{\boldsymbol{a}})(\vec{\boldsymbol{c}} \times \vec{\boldsymbol{a}}) + \vec{\boldsymbol{c}} \cdot (\vec{\boldsymbol{a}} + \vec{\boldsymbol{b}})(\vec{\boldsymbol{a}} \times \vec{\boldsymbol{b}})}{2\vec{\boldsymbol{a}} \cdot (\vec{\boldsymbol{b}} \times \vec{\boldsymbol{c}})}, \\
\vec{\boldsymbol{x}}_T &= \vec{\boldsymbol{x}}_M + \frac{1}{3}(\vec{\boldsymbol{x}}_O - \vec{\boldsymbol{x}}_M), \\
\vec{\boldsymbol{x}}_I &= \frac{\|\vec{\boldsymbol{b}} \times \vec{\boldsymbol{c}}\|\vec{\boldsymbol{a}} + \|\vec{\boldsymbol{c}} \times \vec{\boldsymbol{a}}\|\vec{\boldsymbol{b}} + \|\vec{\boldsymbol{a}} \times \vec{\boldsymbol{b}}\|\vec{\boldsymbol{c}}}{\|\vec{\boldsymbol{b}} \times \vec{\boldsymbol{c}}\| + \|\vec{\boldsymbol{c}} \times \vec{\boldsymbol{a}}\| + \|\vec{\boldsymbol{a}} \times \vec{\boldsymbol{b}}\| + \|\vec{\boldsymbol{b}} \times \vec{\boldsymbol{c}} + \vec{\boldsymbol{c}} \times \vec{\boldsymbol{a}} + \vec{\boldsymbol{a}} \times \vec{\boldsymbol{b}}\|}.
\end{aligned} \tag{20}$$

We utilize 500/2000/2000 samples for training, validation, and testing, respectively. $\epsilon = 10^{-6}$ was added to the denominator to avoid division by zero. In addition, to avoid numerical problems caused by the center point being extremely large, we generated the circumscribed sphere of the tetrahedron when generating the tetrahedron, ensuring that the radius of the sphere is less than or equal to 6.

N-body system. N-body system [72] is a dataset generated from simulations. In our simulations, each system consists of 5 charged particles with random charges $c_i \in \{\pm 1\}$, whose movements are governed by Coulomb forces. The task is to estimate, a node-level target, *i.e.* the coordinates of the N particles after 1000 timesteps. The initial node feature is set as the kinetic energy, while the edge features include the distance and the product of the charges. We use 5000/2000/2000 samples for training, validation, and testing like the setting in HEGNN [49]. Since the particles have different initial velocities, they initially have different characteristics (coloring).

B.2.2 Implementation of Baselines

We undertook further development based on the codes of EGNN [43], TFN [33], and MACE [35] provided in the GWL-test repository. Following our testing, we eliminated the `Gate` modules from both TFN and MACE, as their inclusion resulted in significantly degraded performance. Additionally, for HEGNN [49], we utilized the code[6] from the original paper and opted for the `inner product normalization` option.

Each model utilizes at most 4 layers, and the dimension of the hidden embedding is fixed at 64. The last three models employ at most 2nd-degree steerable features. Both TFN and MACE utilize 8 channels for steerable features, with the correlation parameter for MACE set to 4.

B.2.3 Implementation of Our Models

We provide two complete implementations of EGNN [43] and TFN [33] as follows.

Unlike FastEGNN [66], the virtual nodes in this approach are not updated following their initialization during the message-passing layers. We define the normalization coefficient as $\alpha_i \coloneqq 1/|\mathcal{N}(i)|$. For

[6] https://github.com/GLAD-RUC/HEGNN/.

the ith node, we denote its features as $\boldsymbol{h}_i$ and its coordinates as $\vec{\boldsymbol{x}}_i$. Here, C represents the channels of the virtual nodes, and $\vec{\boldsymbol{Z}} \in \mathbb{R}^{C\times 3}$ denotes all coordinates. Specifically, for TFN$_{\text{cpl}}$-global, we employ the canonical form to build a new feature $\boldsymbol{w}_i$ for each node, thereby rescaling the norm of steerable features in a manner analogous to HEGNN [49], which is consistent with Theorem 3.2.

Table 12: Complete global models.

	EGNN$_{\text{cpl}}$-global	**TFN$_{\text{cpl}}$-global**
Node Coloring	$\boldsymbol{h}_i = \texttt{Color}(\boldsymbol{h}_i, \texttt{distance\_to\_center})$	$\tilde{\boldsymbol{h}}_i^{(0)} = \texttt{Color}(\tilde{\boldsymbol{h}}_i^{(0)}, \texttt{distance\_to\_center})$
Generate VNs	$\vec{\boldsymbol{Z}} = \mathbf{1}_C\vec{\boldsymbol{x}}_c + \alpha_i \sum_{i=1}^N \varphi_{\boldsymbol{Z}}(\boldsymbol{h}_i)\cdot\vec{\boldsymbol{x}}_{ic}$	$\vec{\boldsymbol{Z}} = \mathbf{1}_C\vec{\boldsymbol{x}}_c + \alpha_i \sum_{i=1}^N \varphi_{\boldsymbol{Z}}(\tilde{\boldsymbol{h}}_i^{(0)})\cdot\vec{\boldsymbol{x}}_{ic}$
Create Canonical Form	$\vec{\boldsymbol{M}}_i = \mathbf{1}_C\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{Z}},\ \boldsymbol{m}_{i,\text{cpl}} = \vec{\boldsymbol{M}}_i^\top\vec{\boldsymbol{M}}_i/\|\vec{\boldsymbol{M}}_i^\top\vec{\boldsymbol{M}}_i\|_{\text{F}}$	$\vec{\boldsymbol{M}}_i = \mathbf{1}_C\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{Z}},\ \boldsymbol{m}_{i,\text{cpl}} = \vec{\boldsymbol{M}}_i^\top\vec{\boldsymbol{M}}_i/\|\vec{\boldsymbol{M}}_i^\top\vec{\boldsymbol{M}}_i\|_{\text{F}}$
Update Node Feature	$\boldsymbol{h}_i = \boldsymbol{h}_i + \varphi_{\text{cpl}}(\boldsymbol{m}_{i,\text{cpl}}), \boldsymbol{h}_i = \texttt{Message\_Passing}(\boldsymbol{h}_i, \mathcal{G})$	$\boldsymbol{w}_i = \tilde{\boldsymbol{h}}_i^{(0)} + \varphi_{\text{cpl}}(\boldsymbol{m}_{i,\text{cpl}})$
Message Passing ($\times L$)	$\boldsymbol{m}_{ij} = \varphi_{\boldsymbol{m}}(\boldsymbol{h}_i, \boldsymbol{h}_j, \boldsymbol{e}_{ij}, \|\vec{\boldsymbol{x}}_{ij}\|^2),\ \vec{\boldsymbol{m}}_{ij} = \varphi_{\vec{\boldsymbol{x}}}(\boldsymbol{m}_{ij})\cdot\vec{\boldsymbol{x}}_{ij}$ $\boldsymbol{m}_i = \alpha_i\sum_{j\in\mathcal{N}(i)}\boldsymbol{m}_{ij},\ \vec{\boldsymbol{m}}_i = \alpha_i\sum_{j\in\mathcal{N}(i)}\vec{\boldsymbol{m}}_{ij}$ $\boldsymbol{h}_i = \boldsymbol{h}_i + \varphi_{\boldsymbol{h}}(\boldsymbol{h}_i, \boldsymbol{m}_i),\ \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i + \vec{\boldsymbol{m}}_i$	$\tilde{\boldsymbol{m}}_{ij}^{(\mathbb{L})} = \tilde{\boldsymbol{h}}_i^{(\mathbb{L})} \otimes_{\text{cg}} Y^{(\mathbb{L})}(\vec{\boldsymbol{x}}_{ij})$ $\tilde{\boldsymbol{h}}_i^{(\mathbb{L})} = \tilde{\boldsymbol{h}}_i^{(\mathbb{L})} + \alpha_i\sum_{j\in\mathcal{N}(i)}\tilde{\boldsymbol{m}}_{ij}^{(\mathbb{L})}$ $\tilde{\boldsymbol{h}}_i^{(l)} = \varphi_{\text{cpl}}(\tilde{\boldsymbol{h}}_i^{(0)}, \boldsymbol{w}_i)\cdot\tilde{\boldsymbol{h}}_i^{(l)},\ \boldsymbol{w}_i = \boldsymbol{w}_i + \varphi_{\boldsymbol{w}}(\boldsymbol{w}_i, \tilde{\boldsymbol{h}}_i^{(0)})$
Readout (node-level)	$\boldsymbol{h}_i = \varphi_{\text{out}}(\boldsymbol{h}_i),\ \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i$	$\boldsymbol{h}_i = \varphi_{\text{out}}(\tilde{\boldsymbol{h}}_i^{(0)}),\ \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i + \texttt{e3nn.o3.Linear}(\tilde{\boldsymbol{h}}_i^{(1)})$
Readout (graph-level)	$\boldsymbol{h}_{\mathcal{G}} = \varphi_{\text{pool}}(\frac{1}{N}\sum_{i=1}^N\boldsymbol{h}_i),\ \vec{\boldsymbol{x}}_{\mathcal{G}} = \frac{1}{N}\sum_{i=1}^N\vec{\boldsymbol{x}}_i$	$\boldsymbol{h}_{\mathcal{G}} = \varphi_{\text{pool}}(\frac{1}{N}\sum_{i=1}^N\boldsymbol{h}_i),\ \vec{\boldsymbol{x}}_{\mathcal{G}} = \frac{1}{N}\sum_{i=1}^N\vec{\boldsymbol{x}}_i$

To enhance the model's performance, we permit the virtual nodes to be updated layer by layer, similar to the approach taken in FastEGNN [66]. We denote the normalization coefficient $\alpha_i \coloneqq 1/|\mathcal{N}(i)|$. We define the normalization coefficient as $\alpha_i \coloneqq 1/|\mathcal{N}(i)|$. Notably, we allocate C virtual nodes to each node, denoted as $\vec{\boldsymbol{Z}}_i \in \mathbb{R}^{C\times 3}$, and these C virtual nodes share a scalar feature represented by $\boldsymbol{s}_i$.

Table 13: Complete local models.

	EGNN$_{\text{cpl}}$-local	**TFN$_{\text{cpl}}$-local**
Generate VNs	$\boldsymbol{m}_{ij} = \varphi_{\boldsymbol{m}}(\boldsymbol{h}_i, \boldsymbol{h}_j, \boldsymbol{e}_{ij}, \|\vec{\boldsymbol{x}}_{ij}\|^2)$ $\boldsymbol{s}_i = \varphi_{\boldsymbol{s}}(\varphi_{\text{init}}(\boldsymbol{h}_i), \boldsymbol{m}_{ij})$ $\vec{\boldsymbol{Z}}_i = \mathbf{1}_C\vec{\boldsymbol{x}}_i + \alpha_i\sum_{j\in\mathcal{N}(i)}^N\varphi_{\boldsymbol{Z}}(\boldsymbol{h}_i)\cdot\vec{\boldsymbol{x}}_{ij}$	$\boldsymbol{m}_{ij} = \varphi_{\boldsymbol{m}}(\tilde{\boldsymbol{h}}_i^{(0)}, \tilde{\boldsymbol{h}}_j^{(0)}, \boldsymbol{e}_{ij}, \|\vec{\boldsymbol{x}}_{ij}\|^2)$ $\boldsymbol{s}_i = \varphi_{\boldsymbol{s}}(\varphi_{\text{init}}(\tilde{\boldsymbol{h}}_i^{(0)}), \boldsymbol{m}_{ij})$ $\vec{\boldsymbol{Z}}_i = \mathbf{1}_C\vec{\boldsymbol{x}}_i + \alpha_i\sum_{j\in\mathcal{N}(i)}^N\varphi_{\boldsymbol{Z}}(\boldsymbol{h}_i)\cdot\vec{\boldsymbol{x}}_{ij}$
Message Passing ($\times L$) (real & virtual)	$\boldsymbol{m}_{ij} = \varphi_{\boldsymbol{m}}(\boldsymbol{h}_i, \boldsymbol{h}_j, \boldsymbol{e}_{ij}, \|\vec{\boldsymbol{x}}_{ij}\|^2),\ \vec{\boldsymbol{m}}_{ij} = \varphi_{\vec{\boldsymbol{x}}}(\boldsymbol{m}_{ij})\cdot\vec{\boldsymbol{x}}_{ij}$ $\boldsymbol{m}_i = \alpha_i\sum_{j\in\mathcal{N}(i)}\boldsymbol{m}_{ij},\ \vec{\boldsymbol{m}}_i = \alpha_i\sum_{j\in\mathcal{N}(i)}\vec{\boldsymbol{m}}_{ij}$ $\boldsymbol{h}_i = \boldsymbol{h}_i + \varphi_{\boldsymbol{h}}(\boldsymbol{h}_i, \boldsymbol{m}_i),\ \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i + \vec{\boldsymbol{m}}_i$ $\vec{\boldsymbol{M}}_{ij} = \mathbf{1}_C\vec{\boldsymbol{x}}_j - \vec{\boldsymbol{Z}}_i, \boldsymbol{m}_{ij,\text{cpl}} = \vec{\boldsymbol{M}}_{ij}^\top\vec{\boldsymbol{M}}_{ij}/\|\vec{\boldsymbol{M}}_{ij}^\top\vec{\boldsymbol{M}}_{ij}\|_{\text{F}}$ $\boldsymbol{m}_{ij,\text{cpl}} = \varphi_{\boldsymbol{m},\text{cpl}}(\boldsymbol{s}_i, \boldsymbol{h}_j, \boldsymbol{m}_{ij,\text{cpl}})$ $\boldsymbol{h}_i = \boldsymbol{h}_i + \varphi_{\boldsymbol{h},\text{cpl}}(\boldsymbol{m}_{ij,\text{cpl}}),\ \boldsymbol{s}_i = \boldsymbol{s}_i + \varphi_{\boldsymbol{s},\text{cpl}}(\boldsymbol{m}_{ij,\text{cpl}})$ $\vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i + \alpha_i\sum_{j\in\mathcal{N}(i)}\varphi_{\vec{\boldsymbol{x}}}^{(v)}(\boldsymbol{m}_{ij,\text{cpl}})\cdot\vec{\boldsymbol{M}}_{ij}$ $\vec{\boldsymbol{Z}}_i = \vec{\boldsymbol{Z}}_i + \alpha_i\sum_{j\in\mathcal{N}(i)}\varphi_{\vec{\boldsymbol{Z}}}^{(v)}(\boldsymbol{m}_{ij,\text{cpl}})\cdot(-\vec{\boldsymbol{M}}_{ij})$	$\tilde{\boldsymbol{m}}_{ij}^{(\mathbb{L})} = \tilde{\boldsymbol{h}}_i^{(\mathbb{L})} \otimes_{\text{cg}} Y^{(\mathbb{L})}(\vec{\boldsymbol{x}}_{ij})$ $\tilde{\boldsymbol{h}}_i^{(\mathbb{L})} = \tilde{\boldsymbol{h}}_i^{(\mathbb{L})} + \alpha_i\sum_{j\in\mathcal{N}(i)}\tilde{\boldsymbol{m}}_{ij}^{(\mathbb{L})}$ $\tilde{\boldsymbol{h}}_i^{(l)} = \varphi_{\text{cpl}}(\tilde{\boldsymbol{m}}_i^{(0)})\cdot\tilde{\boldsymbol{h}}_i^{(l)}$ $\vec{\boldsymbol{M}}_{ij} = \mathbf{1}_C\vec{\boldsymbol{x}}_j - \vec{\boldsymbol{Z}}_i, \boldsymbol{m}_{ij,\text{cpl}} = \vec{\boldsymbol{M}}_{ij}^\top\vec{\boldsymbol{M}}_{ij}/\|\vec{\boldsymbol{M}}_{ij}^\top\vec{\boldsymbol{M}}_{ij}\|_{\text{F}}$ $\boldsymbol{m}_{ij,\text{cpl}} = \varphi_{\boldsymbol{m},\text{cpl}}(\boldsymbol{s}_i, \tilde{\boldsymbol{h}}_j^{(0)}, \boldsymbol{m}_{ij,\text{cpl}})$ $\tilde{\boldsymbol{h}}_i^{(0)} = \tilde{\boldsymbol{h}}_i^{(0)} + \varphi_{\tilde{\boldsymbol{h}}^{(0)},\text{cpl}}(\boldsymbol{m}_{ij,\text{cpl}}),\ \boldsymbol{s}_i = \boldsymbol{s}_i + \varphi_{\boldsymbol{s},\text{cpl}}(\boldsymbol{m}_{ij,\text{cpl}})$ $\vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i + \alpha_i\sum_{j\in\mathcal{N}(i)}\varphi_{\vec{\boldsymbol{x}}}^{(v)}(\boldsymbol{m}_{ij,\text{cpl}})\cdot\vec{\boldsymbol{M}}_{ij}$ $\vec{\boldsymbol{Z}}_i = \vec{\boldsymbol{Z}}_i + \alpha_i\sum_{j\in\mathcal{N}(i)}\varphi_{\vec{\boldsymbol{Z}}}^{(v)}(\boldsymbol{m}_{ij,\text{cpl}})\cdot(-\vec{\boldsymbol{M}}_{ij})$
Readout (node-level)	$\boldsymbol{h}_i = \varphi_{\text{out}}(\boldsymbol{h}_i),\ \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i$	$\boldsymbol{h}_i = \varphi_{\text{out}}(\tilde{\boldsymbol{h}}_i^{(0)}),\ \vec{\boldsymbol{x}}_i = \vec{\boldsymbol{x}}_i + \texttt{e3nn.o3.Linear}(\tilde{\boldsymbol{h}}_i^{(1)})$
Readout (graph-level)	$\boldsymbol{h}_{\mathcal{G}} = \varphi_{\text{pool}}(\frac{1}{N}\sum_{i=1}^N\boldsymbol{h}_i),\ \vec{\boldsymbol{x}}_{\mathcal{G}} = \frac{1}{N}\sum_{i=1}^N\vec{\boldsymbol{x}}_i$	$\boldsymbol{h}_{\mathcal{G}} = \varphi_{\text{pool}}(\frac{1}{N}\sum_{i=1}^N\boldsymbol{h}_i),\ \vec{\boldsymbol{x}}_{\mathcal{G}} = \frac{1}{N}\sum_{i=1}^N\vec{\boldsymbol{x}}_i$

Parameter Quantities. Here, we present the number of parameters for each model in the tasks shown in § 4.2. Since the inputs for the two tasks differ, the number of parameters for models with the same number of layers also varies.

Table 14: Tetrahedron dataset.

	Total Parameters			
	1-layer	2-layer	3-layer	4-layer
EGNN	33.7k	67.1k	100.6k	134.1k
FastEGNN	67.4k	134.4k	201.7k	268.8k
HEGNN	46.9k	84.9k	122.9k	160.9k
TFN	46.5k	93.0k	139.4k	185.9k
MACE	48.9k	97.8k	126.6k	195.5k
Equiformer	313.5k	340.5k	367.5k	349.4k
EGNN$_{\text{cpl}}$-global	181.7k	215.2k	248.7k	282.1k
EGNN$^*_{\text{cpl}}$-global	46.8k	55.4k	63.9k	72.5k
EGNN$_{\text{cpl}}$-local	107.5k	180.7k	253.8k	327.0k
EGNN$^*_{\text{cpl}}$-local	28.2k	47.3k	66.5k	85.7k
TFN$_{\text{cpl}}$-global	215.2k	291.1k	267.1k	443.1k
TFN$^*_{\text{cpl}}$-global	70.3k	104.6k	138.9k	173.3k
TFN$_{\text{cpl}}$-local	138.0k	241.7k	345.5k	449.2k
TFN$^*_{\text{cpl}}$-local	51.0k	93.1k	135.2k	177.3k

Table 15: N-body dataset.

	Total Parameters			
	1-layer	2-layer	3-layer	4-layer
EGNN	33.5k	66.9k	100.4k	133.8k
FastEGNN	67.5k	134.6k	201.4k	268.4k
HEGNN	46.7k	84.6k	122.6k	160.5k
TFN	46.4k	92.8k	139.2k	185.6k
MACE	47.2k	94.3k	141.5k	188.6k
Equiformer	313.6k	340.6k	367.5k	394.5k
EGNN$_{\text{cpl}}$-global	181.3k	214.7k	248.1k	281.5k
EGNN$^*_{\text{cpl}}$-global	46.6k	55.1k	63.6k	72.1k
EGNN$_{\text{cpl}}$-local	107.3k	180.4k	253.5k	326.6k
EGNN$^*_{\text{cpl}}$-local	28k	47.2k	66.4k	85.5k
TFN$_{\text{cpl}}$-global	215.6k	291.6k	367.7k	443.7k
TFN$^*_{\text{cpl}}$-global	70.5k	104.8k	139.2k	173.6k
TFN$_{\text{cpl}}$-local	138.2k	242.0k	345.8k	449.7k
TFN$^*_{\text{cpl}}$-local	51.1k	93.3k	135.4k	177.5k

C Rethinking the Equivariance and General Functions

In this section, we give a Fourier series-like approach to understanding equivariant GNNs.

C.1 Rethinking equivariance: A perspective from Fourier expansion.

According to [106], the Fourier expansion of a multivariate function necessitates the simultaneous consideration of the basis functions associated with all variables, as demonstrated below:

$$f(t)=\sum_{n\in\mathbb{Z}} f_n \exp(-2\pi jnt) \longrightarrow f(\boldsymbol{t})=\sum_{\boldsymbol{n}\in\mathbb{Z}^d} f_{\boldsymbol{n}} \exp(-2\pi j\boldsymbol{n}^\top \boldsymbol{t}) = f(\boldsymbol{t})=\sum_{\boldsymbol{n}\in\mathbb{Z}^d} f_{\boldsymbol{n}} \prod_{k=1}^{d} \exp(-2\pi j n_k t_k), \tag{21}$$

represents a multi-dimensional variable. In fact, any function defined on the unit sphere can be expressed as a summation of spherical harmonics. Let $\hat{\boldsymbol{x}}$ denote a point on the unit sphere.

$$f(\hat{\boldsymbol{x}})=\sum_{l=0}^{\infty}\sum_{m=-l}^{l} f_m^{(l)} Y_m^{(l)}(\hat{\boldsymbol{x}}) = \sum_{l=0}^{\infty}\langle \boldsymbol{a}^{(l)}, Y^{(l)}(\hat{\boldsymbol{x}})\rangle, \tag{22}$$

This formulation offers two representations. The fully expanded form aligns more closely with the Fourier series, while the inner product formulation facilitates a connection to equivariance. At this juncture, we introduce a second point $\hat{\boldsymbol{y}}$ on the unit sphere and consider the expansion of the function with two inputs, $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$.

$$f(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})=\sum_{l_1=0}^{\infty}\sum_{l_2=0}^{\infty}\sum_{m_1=-l_1}^{l_1}\sum_{m_2=-l_2}^{l_2} f_{m_1,m_2}^{(l_1,l_2)} Y_{m_1}^{(l_1)}(\hat{\boldsymbol{x}}) Y_{m_2}^{(l_2)}(\hat{\boldsymbol{y}}). \tag{23}$$

We observe that the product $Y_{m_1}^{(l_1)}(\hat{\boldsymbol{x}})Y_{m_2}^{(l_2)}(\hat{\boldsymbol{y}})$ constructs $(2l_1+1)\cdot(2l_2+1)$ basis functions, *i.e.* elements in $Y^{(l_1)}(\hat{\boldsymbol{x}})\otimes Y^{(l_2)}(\hat{\boldsymbol{y}}) \in \mathbb{R}^{(2l_1+1)\cdot(2l_2+1)}$. Let $\boldsymbol{f}^{(l_1\otimes l_2)} \in \mathbb{R}^{(2l_1+1)\cdot(2l_2+1)}$ be the coefficients. Consequently, the expansion can also be expressed in an inner product form as follows:

$$f(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})=\sum_{l_1=0}^{\infty}\sum_{l_2=0}^{\infty}\langle \boldsymbol{f}^{(l_1\otimes l_2)}, Y^{(l_1)}(\hat{\boldsymbol{x}})\otimes Y^{(l_2)}(\hat{\boldsymbol{y}})\rangle. \tag{24}$$

According to the notation of tensor product, we denote the lth-degree steerable feature derived from the tensor product $Y^{(l_1)}(\hat{\boldsymbol{x}})\otimes Y^{(l_2)}(\hat{\boldsymbol{y}})$ as $\tilde{\boldsymbol{b}}^{(l_1\otimes l_2\to l)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$, which can be expressed as follows:

$$\bigoplus_{l=|l_1-l_2|}^{l_1+l_2} \tilde{\boldsymbol{b}}^{(l_1\otimes l_2\to l)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}) = \boldsymbol{Q}^{(l_1\otimes l_2)}\cdot\left(Y^{(l_1)}(\hat{\boldsymbol{x}})\otimes Y^{(l_2)}(\hat{\boldsymbol{y}})\right). \tag{25}$$

Let us define $\bigoplus_{l=|l_1-l_2|}^{l_1+l_2} \boldsymbol{w}^{(l_1\otimes l_2\to l)} = (\boldsymbol{Q}^{(l_1\otimes l_2)})^\top \boldsymbol{f}^{(l_1\otimes l_2)}$. This leads us to the following expression:

$$f(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})=\sum_{l=0}^{\infty}\sum_{|l_1-l_2|\le l\le l_1+l_2}\langle \boldsymbol{w}^{(l_1\otimes l_2\to l)}, \tilde{\boldsymbol{b}}^{(l_1\otimes l_2\to l)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})\rangle, \tag{26}$$

where a lth-degree steerable function $\tilde{\boldsymbol{t}}^{(l)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})$ incorporates only the lth-degree components in Eq. (26), and can be expressed as follows:

$$\tilde{\boldsymbol{t}}^{(l)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})=\sum_{|l_1-l_2|\le l\le l_1+l_2} w^{(l_1\otimes l_2\to l)}\cdot \tilde{\boldsymbol{b}}^{(l_1\otimes l_2\to l)}(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}}), \tag{27}$$

Eq. (27) offers a more concrete understanding of the framework. It is important to note, however, the formula in Eq. (3) is invariant to permutations of the inputs, while the inputs in Eq. (27) are presented with a specific order. Nevertheless, this specificity does not preclude us from conducting an analysis based on this formulation:

1. Since l_1 and l_2 can take any values, there will be infinite basis functions in $\mathbb{B}^{(l)}$ when the geometric graph accommodates interactions involving more than two bodies.

2. Not all bases in Eq. (27) will be meaningful; that is, some bases may not emerge in the objective function, although they satisfy the equivariance constraints.

From this perspective, the advantages and disadvantages of the equivariant model become evident. By incorporating an equivariance prior into the model design, it effectively eliminates components that cannot exist within the framework. However, this constraint necessitates a limited selection of operators in the model design. In the current setup, only the tensor product exhibits strict equivariance (while others can be transformed and expressed using tensor products, *e.g.* tensor decomposition in frame averaging [56; 84]), whereas other operators, such as group convolution, require approximations under current floating-point operation systems and can achieve only partial equivariance. This limitation, in turn, constrains the model's capacity for learning.

C.2 Rethinking General Functions: the Equivariant Decomposition

Generally, the input of a geometric graph could be denoted as $\{\vec{\boldsymbol{x}}_i\}_{i=1}^N$, without loss of generality, we assume that these coordinates are decentralized. Although in practice, people generally use edges as input (*i.e.* $\{\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j\}_{i,j=1}^N$), for the sake of convenience we still discuss the vectors of points here, and it is easy to verify that they are equivalent. Then a normal function $\boldsymbol{f} : \mathbb{R}^{3\times N} \to \mathbb{R}^{2l+1}$ we want in a l-degree problem could be rewritten as:

$$\boldsymbol{f}^{(l)}(\mathcal{G}) = \sum_{\tilde{\boldsymbol{b}}_\alpha^{(l)}(\mathcal{G})\in\mathbb{B}^{(l)}} w_\alpha \tilde{\boldsymbol{b}}_\alpha^{(l)}(\mathcal{G}) + \boldsymbol{f}_{\text{else}}^{(l)}. \tag{28}$$

To completely give the expansion formula of such $\boldsymbol{f}^{(l)}$, we need to analyze each dimension separately. And we denote the mth-dimension ($-l \le m \le l$) of $\boldsymbol{f}^{(l)}$ as $f^{(l,m)}$ (similarly, $b^{(l,m)}$ represents the mth dimension in $\tilde{\boldsymbol{b}}^{(l)}$), which is:

$$f^{(l,m)}(\mathcal{G}) = \sum_{\tilde{\boldsymbol{b}}_\alpha^{(l)}(\mathcal{G})\in\mathbb{B}^{(l)}} w_\alpha^{(l)} \tilde{\boldsymbol{b}}_\alpha^{(l,m)}(\mathcal{G}) + \sum_{s=0}^{\infty} \sum_{\tilde{\boldsymbol{b}}_\beta^{(s)}(\mathcal{G})\in\mathbb{B}^{(s)}} \sum_{t=-s}^{s} w_\beta^{(s,t)} \tilde{\boldsymbol{b}}_\beta^{(s,t)}(\mathcal{G}). \tag{29}$$

And it could find that he mth-dimension $f_{\text{else}}^{(l,m)}$ of $\boldsymbol{f}_{\text{else}}^{(l)}$ could be donated as:

$$f_{\text{else}}^{(l,m)}(\mathcal{G}) = \sum_{s=0}^{\infty} \sum_{\tilde{\boldsymbol{b}}_\beta^{(s)}(\mathcal{G})\in\mathbb{B}^{(s)}} \sum_{t=-s}^{s} w_\beta^{(s,t)} b_\beta^{(s,t)}(\mathcal{G}). \tag{30}$$

Here α and β are dummy index for enumeration and both $w_\alpha^{(l)}$ and $w_\beta^{(l,m)}$ are *constant* (or only the function about all radials $\{\|\vec{\boldsymbol{x}}_i\|\}_{i=1}^N$). Notice that the same basis may appear in both two items in the equation, to ensure uniqueness, we set

$$w_\alpha^{(l)} = \underset{w_\alpha^{(l)}}{\operatorname{argmin}} \|\boldsymbol{f}_{\text{else}}^{(l)}(\mathcal{G})\|^2 = \underset{w_\alpha^{(l)}}{\operatorname{argmin}} \sum_{\tilde{\boldsymbol{b}}_\beta^{(t)}(\mathcal{G})\in\mathbb{B}^{(t)}} \sum_{s=-t}^{t} (w_\beta^{(s,t)})^2. \tag{31}$$

And with definition of equivariance error in Spherical CNNs [107], we could find that

$$\begin{aligned}
\mathcal{L}_{\text{equ}} :=& \mathbb{E}\left[\left\|\rho^{(l)}(\mathfrak{g})\cdot\boldsymbol{f}^{(l)}(\mathcal{G}) - \boldsymbol{f}^{(l)}(\mathfrak{g}\cdot\mathcal{G})\right\|\right] \\
=& \mathbb{E}\left[\left\|\rho^{(l)}(\mathfrak{g})\cdot\boldsymbol{f}_{\text{else}}^{(l)}(\mathcal{G}) - \boldsymbol{f}_{\text{else}}^{(l)}(\mathfrak{g}\cdot\mathcal{G})\right\|\right] \\
\le& \mathbb{E}\left[\left\|\rho^{(l)}(\mathfrak{g})\cdot\boldsymbol{f}_{\text{else}}^{(l)}(\mathcal{G})\| + \|\boldsymbol{f}_{\text{else}}^{(l)}(\mathfrak{g}\cdot\mathcal{G})\right\|\right] \\
\le& 2\cdot\|\boldsymbol{f}_{\text{else}}^{(l)}(\mathcal{G})\|.
\end{aligned} \tag{32}$$

Traditionally, practitioners have opted for data-driven methods, such as data augmentation, to achieve equivariance. This approach aims to constrain the norm $\|\boldsymbol{f}_{\text{else}}^{(l)}\|$ by adjusting the parameters $w_\beta^{(s,t)}$ throughout training, with the goal of minimizing their values. In contrast, models derived from geometric learning directly incorporate the equivariance constraint into the model architecture, effectively achieving $\|\boldsymbol{f}_{\text{else}}^{(l)}\| = 0$. A fundamental aspect of these models is the construction of the

basis set $\mathbb{B}^{(l)}$, allowing them to systematically remove irrelevant or non-existent components from the learning process.

Moreover, for a $\boldsymbol{f}^{(l)}(\mathcal{G})$, obtaining $\boldsymbol{f}_{\text{test}}^{(l)}(\mathcal{G})$ or calculating $\|\boldsymbol{f}_{\text{test}}^{(l)}(\mathcal{G})\|$can be intractable. However, a plausible approach is to utilize another general model whose output is bounded by a constant M to learn the residue. This leads us to the relationship $\|\boldsymbol{f}_{\text{test}}^{(l)}(\mathcal{G})\| \leq \|\boldsymbol{f}^{(l)}(\mathcal{G})\| \leq M$. We introduce this concept here, hoping it will inspire future research.

Additionally, it is important to emphasize that we only require the inference model to be equivariant. In practice, we allow the use of non-equivariant methods for training our model. For instance, noise injection in SaVeNet [108] and the Huber loss in MEAN [102] are examples of such methods. These approaches modify the coefficients w_α, but they do not violate the equivariance constraint.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We prove that a complete equivariant GNN can be achieved through two key components: 1) a complete scalar function, referred to as the canonical form of the geometric graph; 2) a full-rank steerable basis set.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide all these in § 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide all these in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all these in Appendix B.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is available at https://github.com/GLAD-RUC/Uni-EGNN.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all these in Appendices B.2.2 and B.2.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification:We performed multiple runs and reported the variance in all expressivity tests and experiments with MD17.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We conducted experiments on a single NVIDIA H20 GPU.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We used the code from `https://github.com/GLAD-RUC/HEGNN` and `https://github.com/chaitjo/geometric-gnn-dojo`, which are available under the MIT license.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code is available at `https://github.com/GLAD-RUC/Uni-EGNN`.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer:[NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.