A Survey of Geometric Graph Neural Networks: Data Structures, Models and Applications

Jiaqi Han*[4], Jiacheng Cen*[1], Liming Wu*[1], Zongzhao Li[1], Xiangzhe Kong[2], Rui Jiao[2], Ziyang Yu[2], Tingyang Xu[3], Fandi Wu[3], Zihe Wang[1], Hongteng Xu[1], Zhewei Wei[1], Yang Liu◦ [2], Yu Rong◦ [3], Wenbing Huang◦ [1]

Abstract—Geometric graph is a special kind of graph with geometric features, which is vital to model many scientific problems. Unlike generic graphs, geometric graphs often exhibit physical symmetries of translations, rotations, and reflections, making them ineffectively processed by current Graph Neural Networks (GNNs). To tackle this issue, researchers proposed a variety of Geometric Graph Neural Networks equipped with invariant/equivariant properties to better characterize the geometry and topology of geometric graphs. Given the current progress in this field, it is imperative to conduct a comprehensive survey of data structures, models, and applications related to geometric GNNs. In this paper, based on the necessary but concise mathematical preliminaries, we provide a unified view of existing models from the geometric message passing perspective. Additionally, we summarize the applications as well as the related datasets to facilitate later research for methodology development and experimental evaluation. We also discuss the challenges and future potential directions of Geometric GNNs at the end of this survey.

Index Terms—Scientific Systems, Geometric Graphs, Graph Neural Networks, Equivariance, Invariance

1 Introduction

Many scientific problems particularly in physics and biochemistry require to process data in the form of geometric graphs [24]. Distinct from typical graph data, geometric graphs additionally assign each node a special type of node feature in the form of geometric vectors. For example, a molecule/protein can be regarded as a geometric graph, where the 3D position coordinates of atoms are the geometric vectors; in a general multi-body physical system, the 3D states (positions, velocities or spins) are the geometric vectors of the particles. Notably, geometric graphs exhibit symmetries of translations, rotations and/or reflections. This is because the physical law controlling the dynamics of the atoms (or particles) is the same no matter how we translate or rotate the physical system from one place to another. When tackling this type of data,it is essential to incorporate the inductive bias of symmetry into the design of the model, which motivates the study of geometric Graph Neural Networks (GNNs).

Constructing GNNs that permit such symmetry constraints has long been challenging to methodological design. Pioneer approaches such as DTNN [222], DimeNet [135] and GemNet [136], transform the input geometric graph into distance/angle/dihedral-based scalars that are invariant to rotations or translations, constituting the family of invariant GNNs. Noticing the limit on the expressivity of invariant

- * denotes equal contributions; ◦ denotes corresponding authors.
- [1] Gaoling School of Artificial Intelligence, Renmin University of China.
- [2] Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, Tsinghua University.
- [3] Tencent AI Lab.
- [4] Department of Computer Science, Stanford University.

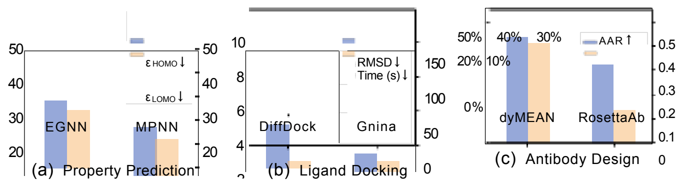


Fig. 1. Performance comparisons between geometric GNNs and traditional methods on molecular property prediction, protein-ligand docking, and an- tibody design. Notably, the methods based on geometric GNNs, including EGNN [216], DiffDock [41], and dyMEAN [142], remarkably outperform traditional MPNN [80], Gnina [179], and RossetaAb [1], on the datasets of QM9 [203], PDBBind [168], and SAbDab [50], respectively, verifying the effectiveness and efficiency of geometric GNNs over various tasks.

GNNs, EGNN [216] and PaiNN [219] additionally involve geometric vectors in message passing and node update to preserve the directional information in each layer, leading to equivariant GNNs. With group representation theory as a helpful tool, TFN [242], SE(3)-Transformer [67] and SEGNN [23] generalizes invariant scalars and equivariant vectors by viewing them as steerable vectors parameterized by high-order spherical tensors, giving rise to high-degree steerable GNNs. Built upon these fundamental approaches, geometric GNNs have made remarkable success in various applications of diverse systems, including physical dynamics simulation [67,216], molecular property prediction [15,152], protein structure prediction [9], protein generation [267,110], and RNA structure ranking [245]. Figure 1 illustrates the superior performance of geometric GNNs against traditional methods on the representative tasks.

To facilitate the research of geometric GNNs, this work presents a systematic survey focusing both on the methods and

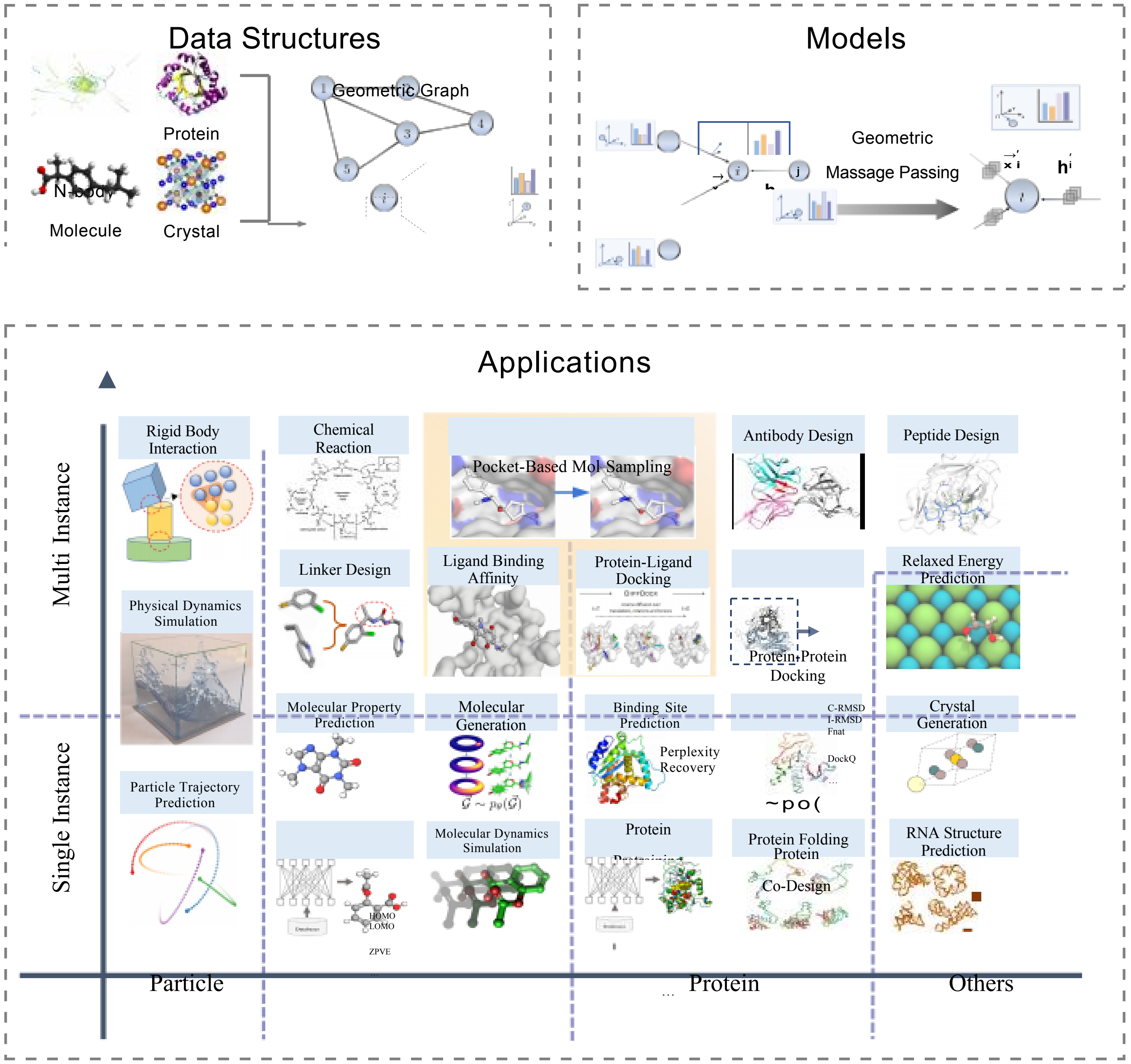


Fig. 2. Illustration of the complete input-output pipeline from data structures, models to applications. Note that most figures here for illustrating different applications are edited based on previous papers [91, 214, 133, 103, 86, 41, 125, 155, 141, 223, 30, 246, 144]. The term "instance" indicates a self-interacted system composed of multiple particles/atoms, such as a molecule or a protein. Pocket-Based Molecule Sampling, Ligand-Binding Affinity Prediction, and Protein-Ligand Docking are denoted with yellow shading to imply that all these tasks take as the multi-instance format "Molecule+Protein".

applications[1], which is structured as the following sections: In §2, we introduce necessary preliminaries on group theory and the formal definition of equivariance/invariance; In §3, we propose geometric graph as a universal data structure that will be leveraged throughout the entire survey as a bridge between real-world data and the models,i.e., geometric GNNs; In §4, we summarize existing models into invariant GNNs (§4.2) and equivariant GNNs (§4.3), while the latter is further categorized into scalarization-based models (§ 4.3.1) and high-degree steerable models (§4.3.2); Besides, we also introduce geometric graph transformers in § 4.4; In § 5, we provide a comprehensive collection of the applications that have witnessed the success of geometric GNNs on particle-

1. This work is an extended survey of our previous short version [93].

based physical systems, molecules, proteins, complexes, and other domains like crystals and RNAs.

The goal of this survey is to provide a general overview throughout data structure, model design, and applications, which constitutes an entire input-output pipeline that is instructive for machine learning practitioners to employ geometric GNNs on various scientific tasks. Recently, several related surveys have been proposed, which place main focus on methodology of geometric GNNs [52], pretrained GNNs for chemical data [276], representation learning for molecules [89, 7], and general application of artificial intelligence in diverse types of scientific systems [299]. In contrast to all of them, this survey places an emphasis on geometric graph neural networks, not only encapsulating theoretical foundations of geometric GNNs but also delivering an exhaustive summary of the related applications in domains across physics, biochemistry, and material science. Meanwhile, we discuss future prospects and interesting research directions in §6. We also release the Github repository that collects the reference, datasets, codes, benchmarks, and other resources related to geometric GNNs.

2 The basic notion of Symmetry

In this section, we will compactly introduce the basic notions related to symmetry. Readers can skip this section and get straight to the methodology part in § 3 if they are familiar with the theoretical background.

2.1 Transformation and Group

By defining symmetry, we indicate that an object of interest keeps invariant under a set of transformations. For instance, the distance between any two points in space remains constant, regardless of how we simultaneously rotate or translate these two points. Mathematically, a set of transformations forms a group (more details are referred to [58]).

Definition 1 (Group). A group G is a set of transformations with a binary operation "$\cdot$" satisfying these properties: (i) it is closed, namely, $\forall a,b \in G, a \cdot b \in G$; (ii) it is associative, namely, $\forall a,b,c \in G, (a \cdot b) \cdot c = a \cdot (b \cdot c)$; (iii) there exists an identity element $e \in G$ such that $\forall a \in G, a \cdot e = e \cdot a = a$; (iv) each element must have an inverse, namely, $\forall a \in G, \exists b \in G, a \cdot b = b \cdot a = e$, where the inverse b is denoted as a^{-1}.

We below provide some examples commonly used in the applications of this paper:

- $E(d)$ is an Euclidean group consisting of rotations, reflections and translations, acting on d-dimension vectors.
- $T(d)$ is a subgroup of Euclidean group that consists of translations.
- $O(d)$ is an orthogonal group that consists of rotations and reflections, acting on d-dimension vectors.
- $SO(d)$ is a special orthogonal group that only consists of rotations.
- $SE(d)$ is a special Euclidean group that consists of only rotations and translations.
- Lie Group is a group whose elements form a differentiable manifold. Actually, all the groups above are specific examples of Lie Group.
- S_N is a permutation group whose elements are permutations of a given set consisting of N elements.

2.2 Group Representation

While the group operation "$\cdot$" is defined abstractly above, it can be realized as matrix multiplication, with the help of group representation. A representation of G is a group homomorphism $\rho(g) : G \mapsto GL(V)$ that takes as input the group element $g \in G$ and acts on the general linear group of some vector space V, satisfying $\rho(g)\rho(h) = \rho(g \cdot h)$, $\forall g, h \in G$. When $V = \mathbb{R}^d$, then $GL(V)$ contains all invertible $d \times d$ matrices and $\rho(g)$ assigns a matrix to the element g.

For the orthogonal group $O(d)$, one of its common group representations is defined by orthogonal matrices $\mathbf{O} \in \mathbb{R}^{d\times d}$ subject to $\mathbf{O}^\top \mathbf{O} = \mathbf{I}$; for $SO(d)$, the group representation is

restricted to orthogonal matrices of determinant 1, denoted as $\mathbf{R}$. The case of translation group T(d) is a bit tedious and can be derived in the projective space using homogeneous coordinates; here, for simplicity, we directly define translation as vector addition other than matrix multiplication. Note that the representation of a group is not unique, which will be further illustrated in § 4.3.2.

2.3 Equivariance and Invariance

Let $\mathcal{X}$ and $\mathcal{Y}$ be the input and output vector spaces, respectively. The function $\phi : \mathcal{X} \rightarrow \mathcal{Y}$ is called equivariant with respect to G if when we apply any transformation to the input, the output also changes via the same transformation or under a certain predictable behavior. In form, we have:

Definition 2 (Equivariance)**.** The function $\phi : \mathcal{X} \mapsto \mathcal{Y}$ is G-equivariant if it commutes with any transformation in G,

$$\phi(g \cdot x) = g \cdot \phi(x), \forall g \in G, \tag{1}$$

which, by implementing the group operation $\cdot$ with group representation, can be rewritten as:

$$\phi(\rho_{\mathcal{X}}(g)x) = \rho_{\mathcal{Y}}(g)\phi(x), \forall g \in G, \tag{2}$$

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

The choice of group representation facilitates the specialization of different scenarios. When both $\rho_{\mathcal{X}}$ and $\rho_{\mathcal{Y}}$ are trivial representations, namely, $\rho_{\mathcal{X}}(g) = \rho_{\mathcal{Y}}(g) = \mathbf{I}$[2], ϕ becomes a trivial function; when $\rho_{\mathcal{Y}}(g) = \mathbf{I}$, ϕ is called an invariant function, demonstrating that invariance is just a special case of equivariance.

It is able to verify that equivariance induces the following desirable properties. (i) **Linearity:** any linear combination of equivariant functions is still equivariant. (ii) **Composability:** the composition of two equivariant functions (if they can be composed) yields an equivariant function. Therefore, equivariance for each layer of a network implies that a whole network is equivariant. (iii) **Inheritability:** if a function is equivariant with respect to group G_1 and group G_2, then this function must beequivariant with respect to the direct product of these two groups, i.e. $G_1 \times G_2$ under a corresponding definition of product group operation or group representation. This implies that proving equivariance of each transformation individually is sufficient to prove equivariance of joint transformations.

In the following context, the variable x is instantiated as a geometric graph, the group transformation $\rho(g)$ becomes the transformation of geometric graphs, and the function ϕ is designed as an invariant/equivariant GNN.

3 Data Structure: from Graph to Geometric Graph

This section formally defines graph and geometric graph, and depicts how they differ from each other. Table 1 summarizes the notations we used throughout this paper.

2. Note that the identity transformation $\mathbf{I}$ could have different dimensions in the input space $\mathcal{X}$ and output space $\mathcal{Y}$.

TABLE 1

Basic notations and definitions throughout this survey.

Notation	Description
	Data Structure
$\mathcal{G} := (\mathbf{A}, \mathbf{H})$	A graph $\mathcal{G}$ containing N nodes, with adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$ and node feature matrix $\mathbf{H} \in \mathbb{R}^{N\times C_h}$.
$\vec{\mathcal{G}} := (\mathbf{A}, \mathbf{H}, \vec{\mathbf{X}})$	A geometric graph $\vec{\mathcal{G}}$ containing N nodes, with adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{H}$ as above, and additionally a 3D coordinate matrix $\vec{\mathbf{X}} \in \mathbb{R}^{N\times 3}$.
$\mathcal{N}_i$	The neighborhood of node i.
$\mathbf{h}_i \in \mathbb{R}^{C_h}$	The scalar feature of node i.
$\vec{\mathbf{x}}_i \in \mathbb{R}^3$	The 3D coordinate of node i.
$\vec{\mathbf{V}}_i \in \mathbb{R}^{3\times C}$	The multi-channel 3D vector of node i.
$\mathbf{V}_i^{(l)} \in \mathbb{R}^{(2l+1)\times C_l}$	The type-l irreducible vector of node i.
$\mathcal{V}_i^{(L)} := \{\mathbf{V}_i^{(l)}\}_{l\in L}$	The set consisting of irreducible vectors of all types $l \in L$.
$\mathbf{e}_{ij} \in \mathbb{R}^{C_e}$	The edge feature from node j to i.
	Operator
$\mathfrak{G}, \mathfrak{g}$	The group $\mathfrak{G}$ and its group element $\mathfrak{g}$.
$\rho_{\mathcal{X}}(\mathfrak{g})$	The group representation $\rho_{\mathcal{X}}(\mathfrak{g})$ of the transformation $\mathfrak{g}$ in the vector space $\mathcal{X}$.
$\times, \otimes$	The operators between two vectors including cross product $\times$ and Kronecker product $\otimes$
$\otimes_{cg}, \otimes_{cg}^{\mathbf{W}}, \otimes_{cg}^{\mathcal{W}}$	Clebsch-Gordan (CG) tensor product, optionally with a learnable parameter $\mathbf{W}$ and a learnable parameter set $\mathcal{W}$.
$Y^{(l)}(\vec{\mathbf{x}}) \in \mathbb{R}^{2l+1}$	The type-l vector constructed by spherical harmonics of $\vec{\mathbf{x}} \in S^2$: $Y^{(l)}(\vec{\mathbf{x}}) = [Y_{-l}^{(l)}, Y_{-l+1}^{(l)}, \cdots, Y_{l-1}^{(l)}, Y_l^{(l)}]$.
$Y^{(L)}(\vec{\mathbf{x}}) := \{Y^{(l)}(\vec{\mathbf{x}})\}_{l\in L}$	A set consisting of spherical harmonics of all types $l \in L$.
$\mathbf{D}^{(l)}(\mathfrak{g})$	The l-th degree Wigner-D matrix of the rotation transformation $\mathfrak{g} \in SO(3)$.
	Neural Network
$\phi, \psi, \varphi, \sigma$	Functions implemented with MLP.

3.1 Graph

Conventional studies on graphs usually focus on their relational topology. Examples include social networks, citation networks, etc. In the domain of AI-Driven Drug Design (AIDD), they are usually referred to as 2D graphs [275].

Definition 3 (Graph)**.** A graph is defined as $\mathcal{G} := (\mathbf{A}, \mathbf{H})$ where $\mathbf{A} \in [0, 1]^{N\times N}$ is the adjacency matrix with N being the number of nodes, and $\mathbf{H} \in \mathbb{R}^{N\times C_h}$ is the node feature matrix with C_h being the dimension of the feature.

Along with the definition of graph, we also describe some vital concepts derived. We denote the set of nodes as $\mathcal{V}$ and the set of edges as $\mathcal{E}$. Correspondingly, the neighborhood of node i, marked as $\mathcal{N}_i$, is specified to be $\mathcal{N}_i := \{j : (v_i, v_j) \in \mathcal{E}\}$. The graph can additionally contain some edge features $\mathbf{e}_{ij} \in \mathbb{R}^{C_e}$ for edge (v_i, v_j).

Transformations on graphs: $\mathfrak{g} \cdot \mathcal{G}$**.** One can arbitrarily change the order of nodes without changing the topology of the graph. With the language of group representation, the permutation transformation of a graph is denoted as $\mathfrak{g} \cdot \mathcal{G} := (\mathbf{P}_\mathfrak{g}\mathbf{A}\mathbf{P}_\mathfrak{g}^\top, \mathbf{P}_\mathfrak{g}\mathbf{H})$, where $\mathbf{P}_\mathfrak{g}$ is the permutation matrix of the transformation $\mathfrak{g} \in S_N$3. We denote the equivalence in terms of permutation as $\mathcal{G} \simeq \mathfrak{g} \cdot \mathcal{G}$.

As a concrete example, molecules can be viewed as graphs, where the nodes v_i are instantiated as the atoms, and the node features $\mathbf{H}$ are the one-hot encoding of the atomic numbers, a row for each atom. The edges $\mathbf{A}$ are either the existence of chemical bonds or constructed based on relative distance between atoms under a cut-off threshold, and the respective

3. The permutation of $\mathbf{A}$ can also be written in the form of group representation by first vectorizing $\mathbf{A}$ as Vec($\mathbf{A}$) and then conducting $(\mathbf{P}_\mathfrak{g} \otimes \mathbf{P}_\mathfrak{g})$Vec($\mathbf{A}$). Here $\otimes$ defines the Kronecker product, and $\mathbf{P}_\mathfrak{g} \otimes \mathbf{P}_\mathfrak{g}$ is the 2-order representation of the permutation matrix.

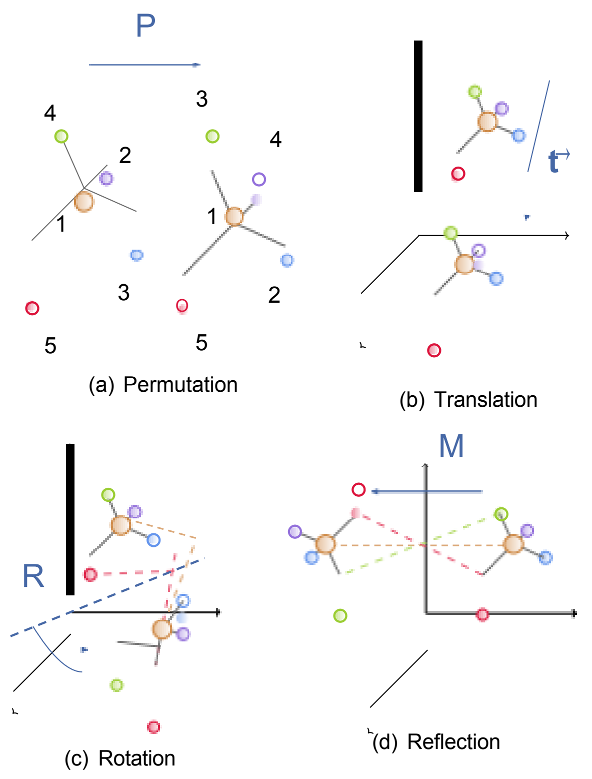


Fig. 3. Examples of transformations on geometric graphs.

edge features $\mathbf{e}_{ij}$ can be assigned as the type of the chemical bond and/or the relative distance.

3.2 Geometric Graph

In many applications, the graphs we tackle contain not only the topological connections and node features, but also certain geometric information. Again, in the example of a molecule, we may additionally be informed of some geometric quantities in the Euclidean space, e.g., the positions of the atoms in 3D coordinates[4]. Such quantities are of particular interest in that they encapsulate rich directional information that depicts the geometry of the system. With the geometric information, one can go beyond working on limited perception of the graph topology, but instead to a broader picture of the entire configuration of the system in 3D space, where important information, such as the relative orientation of the neighboring nodes and directional quantities like velocities, could be better exploited. Hence, in this section, we begin with the definition of geometric graphs, which are usually referred to as 3D graphs [24].

Definition 4 (Geometric Graph). Ageometric graph $\vec{\mathcal{G}}$ is defined as $\vec{\mathcal{G}} := (\mathbf{A}, \mathbf{H}, \vec{\mathbf{X}})$, where $\mathbf{A} \in [0, 1]^{N \times N}$ istheadjacency matrix, $\mathbf{H} \in \mathbb{R}^{N \times C_h}$ is the node feature matrix with dimension C_h, and $\vec{\mathbf{X}} \in \mathbb{R}^{N \times 3}$ are the 3D coordinates of all nodes.

The i-th rows of $\mathbf{H}$ and $\vec{\mathbf{X}}$, namely, $\mathbf{h}_i \in \mathbb{R}^{C_h}$ and $\vec{\mathbf{x}}_i \in \mathbb{R}^3$ denote the feature and coordinate of nodev_i, respectively. In the above definition, we distinguish the coordinate matrix $\vec{\mathbf{X}}$ from other quantities $\mathbf{A}$ and $\mathbf{H}$, and geometric graph $\vec{\mathcal{G}}$ from graph $\mathcal{G}$, with an over-right arrow "→", indicating that they contain geometric and directional information. Note that there could be other geometric variables besides $\vec{\mathbf{X}}$ in a geometric graph, such as velocity, force, and so on. Then the shape of $\vec{\mathbf{X}}$ is extended from $N \times 3$ to $N \times 3 \times C$ where C denotes the number of channels. In this section, we assume $C = 1$ for conciseness, while more complete examples are shown in §5.

4. Although we mainly focus on 3D space, most of our analyses can be extended to$d$-dimensional space where d is an arbitrary integer.

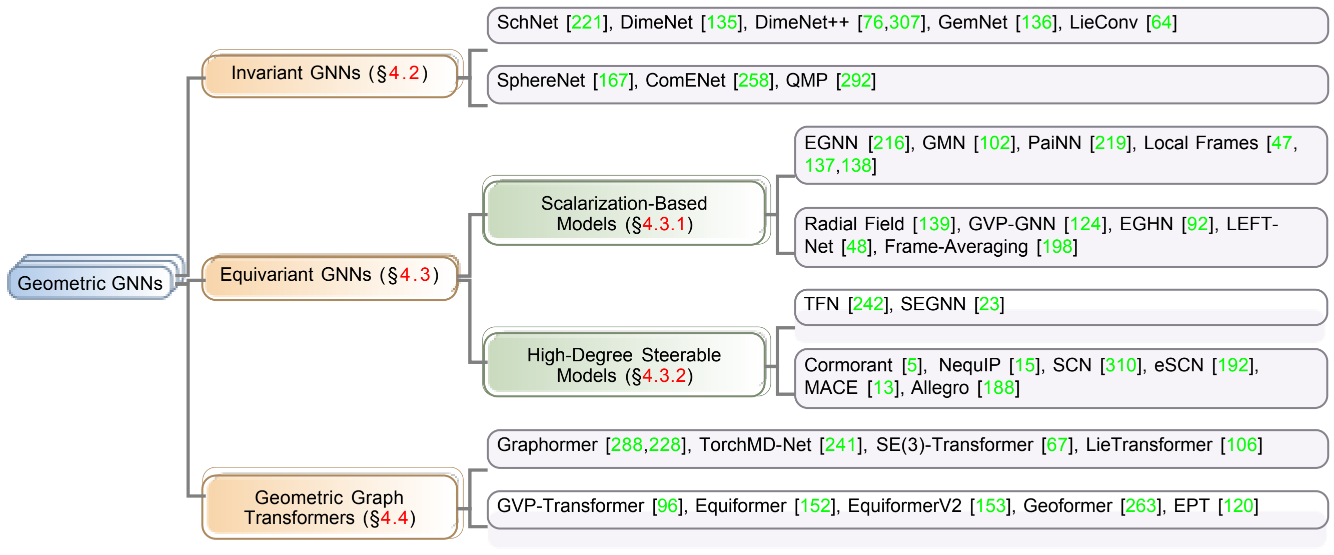

Geometric GNNs
Invariant GNNs (§4.2)
SchNet [221], DimeNet [135], DimeNet++ [76,307], GemNet [136], LieConv [64]
SphereNet [167], ComENet [258], QMP [292]
Equivariant GNNs (§4.3)
Scalarization-Based Models (§4.3.1)
EGNN [216], GMN [102], PaiNN [219], Local Frames [47, 137,138]
Radial Field [139], GVP-GNN [124], EGHN [92], LEFT-Net [48], Frame-Averaging [198]
High-Degree Steerable Models (§4.3.2)
TFN [242], SEGNN [23]
Cormorant [5], NequIP [15], SCN [310], eSCN [192], MACE [13], Allegro [188]
Geometric Graph Transformers (§4.4)
Graphormer [288,228], TorchMD-Net [241], SE(3)-Transformer [67], LieTransformer [106]
GVP-Transformer [96], Equiformer [152], EquiformerV2 [153], Geoformer [263], EPT [120]

Fig. 4. The taxonomy of geometric GNNs introduced in Section 4.

Transformations on geometric graphs: $g \cdot \vec{G}$. In contrast to graphs,transformations on geometric graphs are not limited to node permutation. We summarize the transformations of interest below:

- **Permutation**, which is defined as $g \cdot \vec{G} := (\mathbf{P}_g \mathbf{A} \mathbf{P}_g^\top, \mathbf{P}_g \mathbf{H}, \mathbf{P}_g \vec{\mathbf{X}})$, where $\mathbf{P}_g$ is the permutation matrix representation of $g \in S_n$;
- **Orthogonal transformation (rotation and reflection)**, which is defined as $g \cdot \vec{G} := (\mathbf{A}, \mathbf{H}, \vec{\mathbf{X}} \mathbf{O}_g)$, where $\mathbf{O}_g$ is the orthogonal matrix representation of $g \in O(3)$;
- **Translation**, which is defined as $g \cdot \vec{G} := (\mathbf{A}, \mathbf{H}, \vec{\mathbf{X}} + \vec{\mathbf{t}}_g)$, where $\vec{\mathbf{t}}_g$ is the translation vector of $g \in T(3)$;

We always have the equivalence $\vec{G} \simeq g \cdot \vec{G}$. We can combine orthogonal transformation and translation into Euclidean transformation on geometric graphs, namely, $g \cdot \vec{G} := (\mathbf{A}, \mathbf{H}, \vec{\mathbf{X}} \mathbf{O}_g + \vec{\mathbf{t}}_g)$ for $g \in E(3)$. Here, the Euclidean group $E(3)$ is a semidirect product [254] of orthogonal transformation and translation, denoted as $E(3) = T(3) \ltimes O(3)$. We can similarly define SE(3) transformation by considering only rotation and translation. We sometimes call $\mathbf{H}$ invariant features (or scalars), since they are independent to E(3) transformation, and call $\vec{\mathbf{X}}$ equivariant features (or vectors) that correlate toE(3) transformations. Figure 3 demonstrates the example of transformation on geometric graph.

Geometric graphs are powerful and general tools to model a variety of objects in scientific tasks, including small molecules [221, 216], proteins [10, 110], crystals [175, 118], physical point clouds [102, 91], and many others. We will provide more details in § 5.

4 Model: Geometric GNNs

In this section, we first recap the general form of Message Passing Neural Network (MPNN) on topological graphs. Then we introduce different types of geometric GNNs that are able to process geometric graphs: invariant GNNs, equivariant GNNs, as well as geometric graph transformers. Finally, we briefly present the works that discuss the expressivity of geometric GNNs. Fig. 4presents the taxonomy of geometric GNNs in this section.

4.1 Message Passing Neural Networks

Graph Neural Networks (GNNs) are favorable to operate on graphs with the help of the message-passing mechanism, which facilitates the information propagation along the graph structure by updating node embeddings through neighborhood aggregation. To be specific, message-passing GNNs implement $\phi(\mathcal{G})$ on topological graphs $\mathcal{G}$ by iterating the following message-passing process in each layer [80],

$$\mathbf{m}_{ij} = \phi_{msg}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{e}_{ij}), \tag{3}$$

$$\mathbf{h}'_i = \phi_{upd}(\mathbf{h}_i, \{\mathbf{m}_{ij}\}_{j \in \mathcal{N}_i}), \tag{4}$$

where $\phi_{msg}(\cdot)$ and $\phi_{upd}(\cdot)$ are the message computation and feature update function, respectively. The node features $\mathbf{h}_i$, $\mathbf{h}_j$ and edge feature $\mathbf{e}_{ij}$ is first synthesized by the message function to obtain the message $\mathbf{m}_{ij}$. The messages within the neighborhood are then aggregated with one set function and leveraged to update the node features $\mathbf{h}'_i$ combined with the input $\mathbf{h}_i$.

GNNs defined by Eqs. (3) and (4) are always permutation equivariant but not inherently E(3)-equivariant. When mentioningequivariance or invariance in what follows, this paper mainly discusses the latter unless otherwise specified.

4.2 Invariant Graph Neural Networks

Moving forward to the geometric domain, there are various tasks that require the model we propose to be invariant with regard to Euclidean transformations. For instance, for the task of molecular property prediction, the predicted energy should remain unchanged regardless of any rotation/translation of all atom coordinates. Embedding such inductive bias is crucial as it essentially conforms to the physical rule of our 3D world.

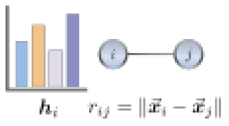
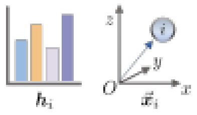
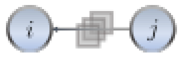
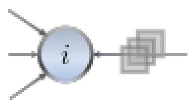
In form, invariant GNNs update invariant features as $\mathbf{H}' = \phi(\vec{\mathcal{G}})$ with the function ϕ satisfying:

$$\phi(g \cdot \vec{\mathcal{G}}) = \phi(\vec{\mathcal{G}}), \forall g \in E(3). \tag{5}$$

To design such function, invariant GNNs usually transform equivariant coordinates $\vec{\mathbf{X}}$ to invariant scalars that are unaffected by Euclidean transformations. Early invariant GNNs can date back to DTNN [222], MPNN [80] and MV-GNN [176], where relative distances are applied for edge construction.

TABLE 2

Illustrations of representative models for invariant GNNs, scalarization-based GNNs and high-degree steerable GNNs.

	Invariant GNNs (e.g. SchNet [221])	Scalarization-Based Models (e.g. EGNN [216])	High-Degree Steerable Models (e.g. TFN [242])
			$Y^{(l)}(x),\ l \in \mathbb{L}$
Message Computation 	$\mathbf{m}_{ij} = \sigma_2(r_{ij})\sigma_1(\mathbf{h}_j)$	$\mathbf{m}_{ij} = \sigma_1(\mathbf{h}_i, \mathbf{h}_j, \|\vec{x}_i - \vec{x}_j\|^2, \mathbf{e}_{ij})$ $\vec{m}_{ij} = (\vec{x}_i - \vec{x}_j)\sigma_2(\mathbf{m}_{ij})$	$\vec{\mathbb{M}}_{ij}^{(L)} = \mathbb{Y}^{(L)}\left(\frac{\vec{x}_{ij}}{\|\vec{x}_{ij}\|}\right) \otimes_{cg}^{W} \vec{\mathbb{V}}_j^{(L)}$
Feature Update 	$\mathbf{h}'_i = \sigma_3\left(\mathbf{h}_i, \Sigma_{j\in N_i} \mathbf{m}_{ij}\right)$	$\mathbf{h}'_i = \sigma_3\left(\mathbf{h}_i, \sum_{j\in\mathcal{N}_i} \mathbf{m}_{ij}\right)$ $\vec{x}'_i = \vec{x}_i + \gamma \sum_{j\in\mathcal{N}_i} \vec{m}_{ij}$	$\vec{V}_i'^{(L)} = \vec{V}_i^{(L)} + \sigma\left(\vec{V}_i^{(L)}, \Sigma_{j\in N(i)} \vec{M}_{ij}^{(L)}\right)$

Recent works further elaborate the use of various invariant scalars ranging from relative distances to angles or dihedral angles between edges, upon the message passing mechanism in Eq. (3)-Eq. (4). We introduce several representative works below.

SchNet [221]. This work designs a continuous filter convolution conditional on relative distances $r_{ij} = \|\vec{x}_i - \vec{x}_j\|$. In particular, it re-implementsEq. (3) as

$$\mathbf{m}_{ij} = \sigma_2(r_{ij})\sigma_1(\mathbf{h}_j), \tag{6}$$

where the message is calculated as the multiplication between the continues convolution filter and the neighbor embedding, and the functions σ are all Multi-Layer Perceptrons (MLPs).

DimeNet [135]. By observing that using relative distances alone is unable to encode directional information, DimeNet proposes directional message passing which takes as input not only relative distances but also angles between adjacent edges. The main component to compute the message embedding of each directional edge (from j to i) is given by:

$$\boldsymbol{m}'_{ji} = \sigma_{\text{msg}}(\boldsymbol{m}_{ji}, \sum_{k\in\mathcal{N}_j\setminus\{i\}} \sigma_{\text{int}}(\boldsymbol{m}_{kj}, \boldsymbol{e}_{\text{RBF}}^{(ji)}, \boldsymbol{e}_{\text{CBF}}^{(kji)})), \tag{7}$$

where $\mathbf{e}_{RBF}^{(ji)}$ denotes the radial basis function representation of relative distance d_{ji}; $\mathbf{e}_{CBF}^{(kji)}$[5] computes the joint representation of relative distance d_{kj} and angle $\alpha_{(kj,ji)}$ between edge (v_k, v_j) and (v_j, v_i), with the help of spherical Bessel functions and spherical harmonics. In [135], Eq. (7) is applied as an interaction block before an embedding block that derives the message $\mathbf{m}_{ji}$ based on $\mathbf{e}_{RBF}^{(ji)}$ and hidden features $\mathbf{h}_i$ and $\mathbf{h}_j$. The updated messages $\mathbf{m}'_{ji}$ of all neighbor nodes are then utilized to update hidden feature $\mathbf{h}_i$. A faster version of DimeNet is proposed later, dubbed DimeNet++ [76,307].

GemNet [136]. To achieve universal expressivity, GemNet further takes dihedral angles into account, formulating two-hop directional message passing based on quadruplets of nodes. Basically, it replaces the message embeddings from Eq. (7) in DimeNet [135] with the following form:

$$\boldsymbol{m}'_{ji} = \sigma_{\text{msg}}\left(\boldsymbol{m}_{ji}, \sum_{\substack{k\in\mathcal{N}_i\setminus\{j\}\\ l\in\mathcal{N}_k\setminus\{i,j\}}} \sigma_{\text{int}}\left(\boldsymbol{m}_{lk}, \boldsymbol{e}_{\text{RBF}}^{(lk)}, \boldsymbol{e}_{\text{CBF}}^{(ikl)}, \boldsymbol{e}_{\text{SBF}}^{(jikl)}\right)\right), \tag{8}$$

5. Here CBF is short for Circular Bessel Function.

where, $\mathbf{e}_R^{(kl}B)_F$ and $\mathbf{e}_C^{(k}_{l}BF)$ are defined as above; $\mathbf{e}_S^{(jik}BF^{l)}$[6] are

calculated by, the spherical Bessel function of relative distance d_{ji}, and spherical harmonics of angle $\alpha_{ji,ik}$ and dihedral angle $\alpha_{ji,kl}$. The input of Eq. (8) additionally integrates hidden features $\mathbf{h}_i$ and $\mathbf{h}_j$ for more expressivity in its original formulation. Note that GemNet can be modified to enable equivariant output by multiplying the output with the associated direction, which belongs to scalarization based equivariant GNNs introduced in the next subsection.

LieConv [64]. LieConv is formulated as follows.

$$\mathbf{m}_{ij} = \sigma\left(\log(u_i^{-1} u_j)\right)\mathbf{h}_j, \tag{9}$$

$$h_i' = \frac{1}{|\mathcal{N}(i)| + 1}\left(h_i + \sum_{j\in\mathcal{N}(i)} m_{ij}\right), \tag{10}$$

where $u_i \in G$ is a lift of $\vec{\mathbf{x}}_i$, the logarithm log maps each group

member onto the Lie Algebra g that is a vector space, and σ is a parametric MLP. Besides, Eq. (10) conducts normalization by the division of the number of all nodes,i.e. $N(i) + 1$. It is clear that LieConv only specifies the update of node features

h_i while keeping the geometric vectors $\vec{\mathbf{x}}_i$ unchanged. That

means LieConv is invariant.

In addition to the above models, SphereNet [167] is another prevailing invariant GNN. Similar to GemNet, SphereNet also exploits relative distances, angles, and torsion angles for geometric modeling, which is able to distinguish almost all 3D graph structures. Moreover, its proposed spherical message passing (SMP) enables fast and accurate 3D molecular learning on large-scale molecules. ComENet [258] is another type of invariant model which incorporates 3D information

6. Here SBF is short for Spherical Bessel Function.