Improving Equivariant Graph Neural Networks on Large Geometric Graphs via Virtual Nodes Learning

Yuelin Zhang [*1 2] Jiacheng Cen [*1 2] Jiaqi Han [*3] Zhiqiang Zhang [4] Jun Zhou [4] Wenbing Huang [1 2]

Abstract

Equivariant Graph Neural Networks (GNNs) have made remarkable success in a variety of scientific applications. However, existing equivariant GNNs encounter the efficiency issue for large geometric graphs and perform poorly if the input is reduced to sparse and local graph for speed acceleration. In this paper, we propose FastEGNN, an enhanced model of equivariant GNNs on large geometric graphs. The central idea is leveraging a small ordered set of virtual nodes to approximate the large unordered graph of real nodes. In particular, we distinguish the message passing and aggregation for different virtual nodes to encourage mutual distinctiveness, and minimize the Maximum Mean Discrepancy (MMD) between virtual and real coordinates to realize the global distributedness. FastEGNN meets all necessary E(3) symmetries, with certain universal expressivity assurance as well. Our experiments on N-body systems (100 nodes), Proteins (800 nodes) and Water-3D (8000 nodes), demonstrate that FastEGNN achieves a promising balance between accuracy and efficiency, and outperforms EGNN in accuracy even after dropping all edges in real systems like Proteins and Water-3D. Code is available at https://github.com/dhcpack/FastEGNN.

1. Introduction

Various scientific data, including chemical molecules, proteins, and other particle-based physical systems, often take the form of *geometric graphs* (Bronstein et al., 2021). Upon the typical graph representation, geometric graphs further

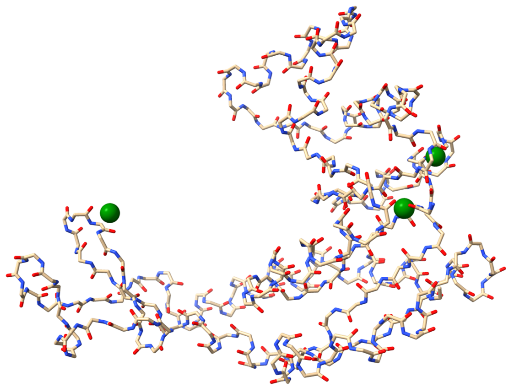


Figure 1. Our FastEGNN is able to learn distinctive virtual nodes (denoted in green) that reflect different dynamical patterns of the protein.

associate each node with specific types of geometric vectors, such as 3D atom coordinates in molecules, or 3D states (positions, velocities or spins) of each particle in general physical systems. Geometric graphs prominently display symmetries involving translations, rotations, and/or reflections, which arise from the uniformity of the physical laws governing atom (or particle) dynamics, regardless of absolute position and orientation. When addressing such data, it's crucial to integrate the aforementioned symmetries into the model design, giving rise to the research of equivariant Graph Neural Networks (GNNs) (Han et al., 2022c; Duval et al., 2023). In recent years, equivariant GNNs have made remarkable success in scientific applications, including physical dynamics simulation (Wu et al., 2024; Xu et al., 2024), protein generation (Watson et al., 2023; Ingraham et al., 2023), and many others.

Despite fruitful progress, existing equivariant GNNs will encounter the efficiency issue for large geometric graphs which exist widely in physical dynamics simulation. For instance, the message exchange in EGNN (Satorras et al., 2021), one of the most prevailing models, leads to quadratic complexity regarding the number of nodes for fully-connected graphs, which is unacceptable in practical scenarios, such as fluid dynamics simulation, where the number of particles are usually more than tens of thousands. One natural way to improve the efficiency is to decrease the edge number by,

[*]Equal contribution [1]Gaoling School of Artificial Intelligence, Renmin University of China [2]Beijing Key Laboratory of Big Data Management and Analysis Methods [3]Stanford University [4]Ant Group. Correspondence to: Wenbing Huang <hwenbing@126.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

for example, limiting the exchange of messages to local neighbors that are within a cutoff radius. Nevertheless, this comes at the cost of sacrificing message passing over nodes and probably leads to performance detriment.

In this paper, we tackle the aforementioned issue with the aid of *virtual nodes*. We refer the nodes in the original geometric graphs as *real nodes* for better discrimination. By design, each virtual node is allowed to connect all real nodes, and all other virtual nodes as well. It ensures that the message exchange between each pair of real nodes is maintained, although passing through virtual nodes. In this way, we are still able to conduct globally dense message passing and further improve the performance, even when we reduce large geometric graphs into sparse ones for efficiency consideration. Furthermore, from a physics standpoint, the force interaction among a set of particles and any other remotely located particle can be compactly translated as the interaction between an equivalent particle of the set and the distant particle (Darve, 2000). This equivalence principle greatly motivates our study, as we expect to represent the equivalent particles with our learned virtual nodes.

The central problem is how to learn these virtual nodes effectively. In principle, the virtual nodes we desire should exhibit two geometric characteristics: *mutual distinctiveness* and *global distributedness*. The former characteristic indicates that different virtual nodes should encode different aspects of the entire geometric graph, including different spatially clustering centers or other distinct geometric measurements. The second characteristic suggests that the spatial distribution of virtual nodes should align with the actual nodes, which essentially encourages the virtual nodes to function as equivalent substitutes for the real nodes. These two aspects are correlated, as the mutual distinctiveness promotes the global distributedness, and vice versa.

In summary, we propose *FastEGNN* upon the EGNN backbone (Satorras et al., 2021), with the following designs:

- We propose to learn a certain number of virtual nodes that facilitate global message passing over the sparse reduction of large geometric graphs. To the best of our knowledge, we are the first to investigate equivariant virtual nodes learning on large geometric graphs.
- Regarding the mutual distinctiveness, we treat the virtual nodes as an ordered set and formulate their 3D coordinates as different channels of a virtual matrix. The virtual-virtual message passing is designed as an E(3)-invariant inner-product of the center-translated virtual matrix, with which we elaborate the E(3)-equivariant real-virtual message passing by constructing different aggregation functions for different virtual nodes.
- As for the global distributedness, we propose to leverage Maximum Mean Discrepancy (MMD) (Borgwardt et al., 2006) which is equipped with an E(3)-invariant kernel, to enforce the alignment between virtual and real coordinates. The MMD objective is implemented by sampling a small-scale subset of real nodes.

We prove that FastEGNN meets all necessary E(3) symmetries, with certain universal expressivity assurance devised as well. Our experiments on three large-scale systems: N-body systems, proteins and Water-3D, demonstrate that FastEGNN achieves a promising balance between accuracy and efficiency, and outperforms EGNN in accuracy even when dropping all edges in proteins and water-3D.

Note that previous works (Gilmer et al., 2017; Han et al., 2022a; Kong et al., 2023) have conducted the attempt by injecting a global node into the input graph, but this injected global node is simply considered as the mean of the graph, or specifically designed via external knowledge, without satisfying the above two mentioned characteristics. We will demonstrate in our experiments that learning virtual nodes under these constraints improves performance. Besides, one might consider utilizing off-the-shelf clustering methods (such as K-means) to locate the cluster centers as virtual nodes. However, it will include more computation overhead and is not end-to-end learnable by the target task.

2. Related Work

Geometric GNNs. Geometric GNNs can be classified into invariant and equivariant models. Invariant GNNs, exemplified by SchNet (Schütt et al., 2018) and DimeNet (Klicpera et al., 2020), have made initial attempts to embed geometric information into invariance features like distance and angles. However, a notable gap still exists in achieving full equivariance within these models, leading to the study of equivariant GNNs. For example, high-degree steerable GNNs, including TFN (Thomas et al., 2018), SEGNN (Brandstetter et al., 2022), and SE(3)-Transformer (Fuchs et al., 2020), leverage the equivariance of spherical harmonics, offering commendable physical interpretability. Nevertheless, the substantial computational cost associated with spherical harmonic forms poses a significant limitation for their application in large-scale tasks. In addition to the aforementioned models, scalarization-based GNNs (Satorras et al., 2021; Köhler et al., 2019; Jing et al., 2021) present a more elegant and expedient approach to representing geometric information, relying solely on linear combinations of geometric vectors. Despite the rich literature and sophisticated architectures, the effectiveness of geometric GNNs usually relies on fully-connected geometric graphs to ensure accuracy. However, this necessity introduces square-level complexity, thereby constraining the practical implementation of geometric GNNs in large-scale scientific problems. Our work tackles these challenges, aiming to enhance the scalability and applicability of geometric GNNs in scientific domains.

Virtual nodes for GNNs. In prior investigations on conventional GNNs (Gilmer et al., 2017; Pham et al., 2017; Hwang et al., 2022), attempts were made to maintain graph connectivity by introducing virtual nodes connected to all graph nodes. This design ensures that any two nodes can exchange information through the process of reading from and writing into virtual nodes at each step of message passing. In the realm of geometric GNNs, certain existing approaches treat virtual nodes as specifically defined clusters based on prior knowledge. They are exclusively connected to nodes belonging to the same cluster, such as different objects in SGNN (Han et al., 2022a) or distinct protein chains in MEAN (Kong et al., 2023). In contrast to these methods, our approach involves end-to-end learning of virtual nodes from the dataset, eliminating the reliance on prior or external knowledge. Notably, the learning process for virtual nodes is thoughtfully designed to ensure both mutual distinctiveness and global distributedness.

3. Preliminaries

Geometric graph. A geometric graph of N nodes is defined as $\vec{\mathcal{G}} := (\vec{\boldsymbol{X}}, \boldsymbol{H}; \mathcal{E})$, where $\vec{\boldsymbol{X}} = [\vec{\boldsymbol{x}}_1, \vec{\boldsymbol{x}}_2, \cdots, \vec{\boldsymbol{x}}_N]^\top \in \mathbb{R}^{N\times 3}$ is the collection of 3D coordinates for the nodes, $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N]^\top \in \mathbb{R}^{N\times H}$ is the node feature matrix, and $\mathcal{E}$ is the set of edges representing the interactions between nodes. In the physical dynamics simulation scenarios, we are additionally provided with the velocities $\vec{\boldsymbol{V}} = [\vec{\boldsymbol{v}}_1, \vec{\boldsymbol{v}}_2, \cdots, \vec{\boldsymbol{v}}_N]^\top \in \mathbb{R}^{N\times 3}$ and optionally the edge feature $\boldsymbol{e}_{ij} \in \mathbb{R}^E$ for every edge $(i, j) \in \mathcal{E}$.

Equivariance. Formally, a function ϕ is equivariant *w.r.t.* group G if $\phi(\rho_{\mathcal{X}}(g)\boldsymbol{x}) = \rho_{\mathcal{Y}}(g)\phi(\boldsymbol{x}), \forall g \in G$, where $\rho_{\mathcal{X}}(g)$ and $\rho_{\mathcal{Y}}(g)$ are the representation of transformation g in the input space $\mathcal{X}$ and output space $\mathcal{Y}$, respectively. In this work, we focus on the 3D Euclidean group E(3), whose elements $g \in \mathrm{E}(3)$ can be realized by an orthogonal matrix $\boldsymbol{O} \in \mathbb{R}^3, \boldsymbol{O}^\top\boldsymbol{O} = \boldsymbol{I}$ and a translation vector $\vec{\boldsymbol{t}} \in \mathbb{R}^3$. By this means, every E(3)-equivariant GNN ϕ on geometric graph $\vec{\mathcal{G}}$ taking as input coordinates $\vec{\boldsymbol{X}}$, velocities $\vec{\boldsymbol{V}}$, and node features $\boldsymbol{H}$ should satisfy the following constraint:

$$\vec{\boldsymbol{X}}'\boldsymbol{O} + \vec{\boldsymbol{t}}, \boldsymbol{H}' = \phi(\vec{\boldsymbol{X}}\boldsymbol{O} + \vec{\boldsymbol{t}}, \vec{\boldsymbol{V}}\boldsymbol{O}, \boldsymbol{H}, \mathcal{E}), \forall(\boldsymbol{O}, \vec{\boldsymbol{t}}) \in \mathrm{E}(3), \tag{1}$$

where $\vec{\boldsymbol{X}}', \boldsymbol{H}' = \phi(\vec{\boldsymbol{X}}, \vec{\boldsymbol{V}}, \boldsymbol{H}, \mathcal{E})$ are the updated coordinates and node features. Note that the velocities $\vec{\boldsymbol{V}}$ are not affected by the translation $\vec{\boldsymbol{t}}$. We do not specify permutation equivariance in Eq. (1), since it has been intrinsically encoded in GNNs.

Problem formulation. We mainly focus on the position prediction task which is fundamental to many physical simulation scenarios (Fuchs et al., 2020; Satorras et al., 2021). Specifically, given the initial geometric graph with coordinates $\vec{\boldsymbol{X}}$, node features $\boldsymbol{H}$, velocities $\vec{\boldsymbol{V}}$, edges $\mathcal{E}$ and their attributes $\{\boldsymbol{e}_{ij} : (i, j) \in \mathcal{E}\}$, we seek to predict $\vec{\boldsymbol{X}}'$ as the positions after a fixed time Δt. It is an E(3)-equivariant task, since if we rotate or translate the initial positions, the predicted positions should rotate or translate in accordance. That is, we aim to design an E(3)-equivariant function ϕ that meet the constraint of Eq. (1).

4. Our FastEGNN

In this section, we introduce the details of our proposed FastEGNN. We first present the representation of virtual nodes in § 4.1. Then, we design the message passing and aggregation among the real and virtual nodes in § 4.2, and provide the learning objectives of our model in § 4.3. Finally, we will explain the benefit of involving virtual nodes in large geometric graph learning in § 4.4.

As claimed in Introduction, FastEGNN is designed to ensure the mutual distinctiveness and the global distributedness of the virtual nodes. We will specify that the mutual distinctiveness is guaranteed by the independent parameterization of the virtual nodes in § 4.1, and separate message passing and aggregation with different functions in § 4.2. As for the global distributedness, we achieve this property by employing the MMD loss as a regularization term in § 4.3.

4.1. Geometric graphs with virtual nodes

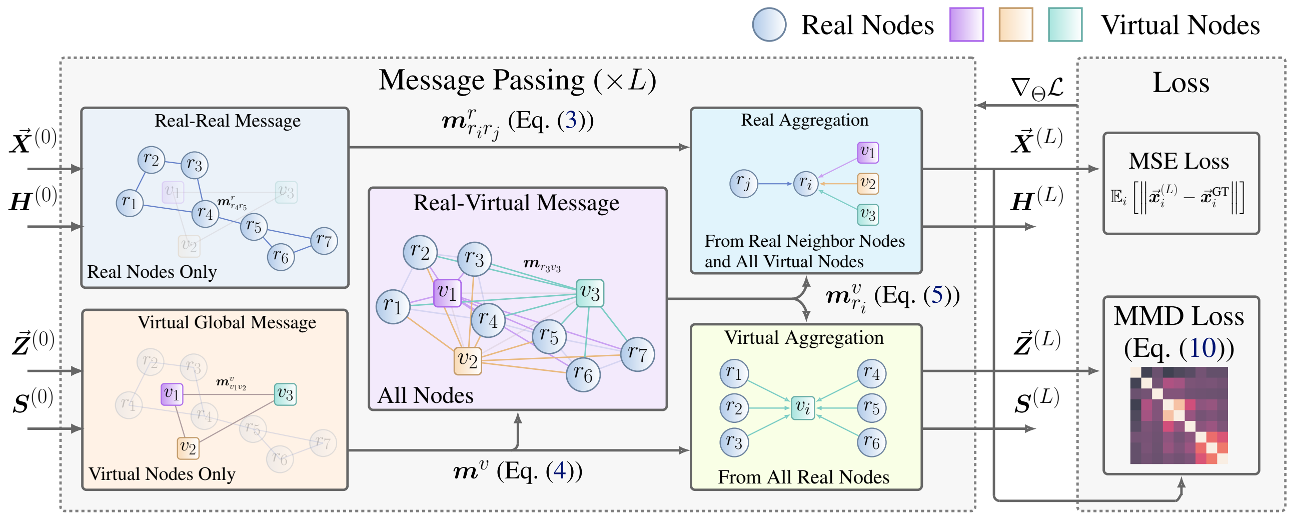
With the input geometric graph $\vec{\mathcal{G}}$ as defined before, we create an augmented geometric graph $\vec{\mathcal{G}}^v$ by adding a set of C virtual nodes $(\vec{\boldsymbol{Z}}, \boldsymbol{S})$, where $\vec{\boldsymbol{Z}} \in \mathbb{R}^{3\times C}$ and $\boldsymbol{S} \in \mathbb{R}^{H\times C}$ refer to the 3D coordinates and invariant features, respectively. Each virtual node is connected to all real nodes in $\vec{\mathcal{G}}$, as well as each other virtual node. This augments the original edge set $\mathcal{E}$ with new connections, leading to $\mathcal{E}^v$. Overall, $\vec{\mathcal{G}}^v := (\vec{\boldsymbol{X}}, \vec{\boldsymbol{V}}, \boldsymbol{H}; \vec{\boldsymbol{Z}}, \boldsymbol{S}; \mathcal{E}^v)$.

As the virtual nodes are not observed in the input data, we need to handcraft the initialization of their values. We adopt the following initialization strategy for the virtual nodes: $\vec{\boldsymbol{Z}} = \frac{1}{N}\sum_{i=1}^N \vec{\boldsymbol{x}}_i \boldsymbol{1}^\top$, where each channel in $\vec{\boldsymbol{Z}}$ is initialized as the Center-of-Mass (CoM) of the input geometric graph, ensuring E(3)-equivariance; each channel of $\boldsymbol{S} \in \mathbb{R}^{H\times C}$ is initialized as independent learnable parameters that are optimized during training. In form, the virtual nodes are initialized as a function of the real nodes, namely, $\vec{\boldsymbol{Z}}, \boldsymbol{S} = \varphi_{\text{int}}(\vec{\boldsymbol{X}}, \boldsymbol{H})$, which should abide by:

$$\boldsymbol{O}\vec{\boldsymbol{Z}} + \vec{\boldsymbol{t}}, \boldsymbol{S} = \varphi_{\text{int}}(\boldsymbol{P}\vec{\boldsymbol{X}}\boldsymbol{O} + \vec{\boldsymbol{t}}, \boldsymbol{P}\boldsymbol{H}), \tag{2}$$

for any orthogonal transformation $\boldsymbol{O} \in \mathbb{R}^{3\times 3}$, translation $\vec{\boldsymbol{t}} \in \mathbb{R}^3$, and permutation $\boldsymbol{P} \in \{0, 1\}^{N\times N}$. $\vec{\boldsymbol{Z}}$ is E(3)-equivariant and permutation-invariant with respect to $\vec{\boldsymbol{X}}$.

One crucial point we would like to emphasize is that the channels of $\vec{\boldsymbol{Z}}$ (and $\boldsymbol{S}$) actually form an *ordered* set; in other words, different channel, namely, different virtual node, is



Real Nodes
Virtual Nodes
Message Passing ($\times L$)
Real-Real Message
Real Nodes Only
$\boldsymbol{m}^r_{r_i r_j}$ (Eq. (3))
Real Aggregation
From Real Neighbor Nodes and All Virtual Nodes
Real-Virtual Message
All Nodes
Virtual Global Message
Virtual Nodes Only
$\boldsymbol{m}^v$ (Eq. (4))
$\boldsymbol{m}^v_{r_i}$ (Eq. (5))
Virtual Aggregation
From All Real Nodes
$\vec{\boldsymbol{X}}^{(0)}$
$\boldsymbol{H}^{(0)}$
$\vec{\boldsymbol{Z}}^{(0)}$
$\boldsymbol{S}^{(0)}$
$\vec{\boldsymbol{X}}^{(L)}$
$\boldsymbol{H}^{(L)}$
$\vec{\boldsymbol{Z}}^{(L)}$
$\boldsymbol{S}^{(L)}$
$\nabla_\Theta \mathcal{L}$
Loss
MSE Loss
$\mathbb{E}_i \left[\left\| \vec{\boldsymbol{x}}_i^{(L)} - \vec{\boldsymbol{x}}_i^{\mathrm{GT}} \right\| \right]$
MMD Loss (Eq. (10))

Figure 2. The overall architecture of FastEGNN. $(\vec{\boldsymbol{X}}, \boldsymbol{H})$ are the real coordinates and features; $(\vec{\boldsymbol{Z}}, \boldsymbol{S})$ are the virtual coordinates and features. Each layer contains 5 components: Real-Real Message $\boldsymbol{m}^r_{r_i, r_j}$, Virtual Global Message $\boldsymbol{m}^v$, Real-Virtual Message $\boldsymbol{m}^v_{r_i}$, Real Aggregation, and Virtual Aggregation. The real-real and real-virtual edges are displayed with different colors to indicate that the message passing and aggregation functions over the edges are different.

endowed with different meaning and can not be organized in arbitrary permutation. This restriction, along with the separate message passing mechanism in the next subsection, serves our purpose of attaining the mutual distinctiveness amongst the virtual nodes. Besides, such orderliness also facilitates the derivation of the message function with universal expressivity, which will be detailed in § 4.4.

4.2. Message Passing and Aggregation

After obtaining the initialized virtual nodes, we conduct the update of all virtual nodes and all real nodes via E(3)-equivariant message passing, as illustrated in Figure 2. This includes real-real message computation, virtual global message calculation, real-virtual message derivation, real node aggregation, and virtual node aggregation. We introduce each component in detail below.

We denote by $(\vec{\boldsymbol{X}}^{(0)}, \vec{\boldsymbol{V}}^{(0)}, \boldsymbol{H}^{(0)}; \vec{\boldsymbol{Z}}^{(0)}, \boldsymbol{S}^{(0)}; \mathcal{E}^v)$ the initialization, and specify the l-th layer with superscript l.

Real-real message. We first derive the message between all the real nodes in the way akin to EGNN (Satorras et al., 2021), which is given by:

$$\boldsymbol{m}^r_{ij} = \varphi_1(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}, \|\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{x}}_j^{(l)}\|^2, \boldsymbol{e}_{ij}), \tag{3}$$

where φ_1 is a Multi-Layer Perceptron (MLP).

Virtual global message. Distinct virtual node represents distinct aspect of the geometric graph. Therefore, exhausting the correlation between each pair of the virtual nodes is able the reflect the entire behavior of the input system. To do so, we compute an invariant matrix extracted from the inner-product of the center-translated virtual coordinates:

$$\boldsymbol{m}^v = (\vec{\boldsymbol{Z}}^{(l)} - \bar{\boldsymbol{x}}\boldsymbol{1}^\top)^\top (\vec{\boldsymbol{Z}}^{(l)} - \bar{\boldsymbol{x}}\boldsymbol{1}^\top), \tag{4}$$

where $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^N \vec{\boldsymbol{x}}_i$ is the CoM of the geometric graph. Clearly, $\boldsymbol{m}^v \in \mathbb{R}^{C\times C}$ is an E(3)-invariant matrix, and indeed an universal approximation of any E(3)-invariant function of the virtual coordinates $\vec{\boldsymbol{Z}}^{(l)}$, according to the proof of Lemma 2 in Huang et al. (2022).

Real-virtual message. The next step is to compute the message between the real nodes and the virtual nodes by:

$$\boldsymbol{m}^v_i = \varphi_2\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{S}^{(l)}, \bigoplus_{c=1}^{C} \|\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{z}}_c^{(l)}\|^2, \boldsymbol{m}^v\right), \tag{5}$$

where $\bigoplus$ defines the concatenations along the channel dimension, and φ_2 is an MLP. Eq. (5) returns the message between each real node i and the global virtual set, which will be consistently applied during the later real and virtual node aggregation processes. Again, Eq. (5) is E(3)-invariant. In our implementation, we find that computing message between real node i and virtual node c separately, namely, $\boldsymbol{m}^v_{ic} = \varphi_2\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{s}_c^{(l)}, \|\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{z}}_c^{(l)}\|^2, \boldsymbol{m}^v_c\right)$ enables more effective training and better performance, in contrast to the global message from all virtual nodes in Eq. (5). Here, $\boldsymbol{m}^v_c$ denotes the c-th column of $\boldsymbol{m}^v$.

Real aggregation. Given the real-real message $\boldsymbol{m}^r_{ij}$ and real-virtual message $\boldsymbol{m}^v_i$, we are ready to derive the message aggregation for real node i as follows:

$$\vec{\boldsymbol{x}}_i^{(l+1)} = \vec{\boldsymbol{x}}_i^{(l)} + \alpha_i \sum_{j\in\mathcal{N}(i)} (\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{x}}_j^{(l)})\varphi_x^r(\boldsymbol{m}_{ij}^r) + \frac{1}{C}\sum_{c=1}^{C}(\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{z}}_c^{(l)})\varphi_x^v(\boldsymbol{m}_{ic}^v) + \varphi_v(\boldsymbol{h}_i^{(l)})\vec{\boldsymbol{v}}_i^{(0)}, \tag{6}$$

$$\boldsymbol{h}_i^{(l+1)} = \boldsymbol{h}_i^{(l)} + \varphi_h\left(\boldsymbol{h}_i^{(l)}, \bigoplus_{c=1}^{C} \boldsymbol{m}_{ic}^v, \alpha_i \sum_{j\in\mathcal{N}(i)} \boldsymbol{m}_{ij}^r\right), \tag{7}$$

where $\alpha_i = \frac{1}{|\mathcal{N}(i)|}$, $\varphi_x^r, \varphi_x^v, \varphi_v$ are MLPs outputting one-dimensional scalars, and φ_h is also an MLP with the output dimension complying with that of $\boldsymbol{h}_i^{(l+1)}$. The update of $\vec{\boldsymbol{x}}_i^{(l+1)}$ (and similarly $\boldsymbol{h}_i^{(l+1)}$) is inspired by EGNN but further extended with the message from the virtual nodes, namely, the term $\sum_{c=1}^{C}(\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{z}}_j^{(l)})\varphi_x^v(\boldsymbol{m}_i^v)$. Previous idea of adding a global node in Han et al. (2022a) is more like a degenerated version of Eq. (6) where φ_x^r and φ_x^v are shared.

Virtual aggregation. Now, we devise the virtual node aggregation as:

$$\vec{\boldsymbol{z}}_c^{(l+1)} = \vec{\boldsymbol{z}}_c^{(l)} + \frac{1}{N}\sum_{i=1}^{N}(\vec{\boldsymbol{z}}_c^{(l)} - \vec{\boldsymbol{x}}_i^{(l)})\varphi_Z(\boldsymbol{m}_{ic}^v), \tag{8}$$

$$\boldsymbol{s}_c^{(l+1)} = \boldsymbol{s}_c^{(l)} + \varphi_S\left(\boldsymbol{s}_c^{(l)}, \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{m}_{ic}^v\right), \tag{9}$$

where, $\varphi_Z \in \mathbb{R}$ and φ_S are also MLPs. To be specific, the calculation $\sum_{i=1}^{N}(\vec{\boldsymbol{z}}_c^{(l)} - \vec{\boldsymbol{x}}_i^{(l)})\varphi_Z(\boldsymbol{m}_{ic}^v)$ in Eq. (8) aggregates all messages from all real nodes, multiplied with different scalar ($\varphi_Z(\boldsymbol{m}_{ic}^v)$). In this way, the update of different virtual node is independent to each other, for the sake of encouraging better mutual distinctiveness.

We have the flowing theoretical assurance by our design:

Proposition 4.1. *If the initialization of the virtual nodes satisfies Eq. (2), then after Eqs. (3) to (9), the output coordinates $\vec{\boldsymbol{X}}^{(L)}$ are E(3)-equivariant and permutation-equivaraint, the virtual coordinates $\vec{\boldsymbol{Z}}^{(L)}$ are E(3)-equivariant and permutation-invariant, with respect to the input $\vec{\boldsymbol{X}}^{(0)}$.*

4.3. Learning Objectives

In this subsection, we exploit the MMD objective (Borgwardt et al., 2006) to enforce the alignment between virtual and real coordinates. MMD is widely used in the research of domain adaption. Here, we derive $\mathcal{L}_{\text{MMD}}$ without the original term $k(\vec{\boldsymbol{x}}_i^{(L)}, \vec{\boldsymbol{x}}_j^{(L)})$ as:

$$\frac{1}{C^2}\sum_{i=1}^{C}\sum_{j=1}^{C} k(\vec{\boldsymbol{z}}_i^{(L)}, \vec{\boldsymbol{z}}_j^{(L)}) - \frac{1}{NC}\sum_{i=1}^{N}\sum_{j=1}^{C} k(\vec{\boldsymbol{x}}_i^{(L)}, \vec{\boldsymbol{z}}_j^{(L)}), \tag{10}$$

where the RBF kernel $k(\vec{\boldsymbol{x}}, \vec{\boldsymbol{y}}) = \exp(-\frac{\|\vec{\boldsymbol{x}}-\vec{\boldsymbol{y}}\|^2}{2\sigma^2})$ is E(3)-invariant, hence the MMD loss is also E(3)-invariant. Interestingly, minimizing the first term in Eq. (10) is actually enlarging the divergence between the virtual nodes, while maximizing the second term enhances the similarity between the virtual nodes and the real ones. Note that we can just sample a small-scale subset of the real nodes for the MMD calculation at each iteration to avoid redundant costs.

The overall training loss $\mathcal{L}$ is a combination of the MSE loss between the predicted positions and the ground truth, together with the proposed auxiliary E(3)-invariant MMD loss to expand the coverage of the virtual nodes towards the data distribution:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}}(\vec{\boldsymbol{X}}^{(L)}, \vec{\boldsymbol{X}}_{\text{GT}}) + \lambda\mathcal{L}_{\text{MMD}}(\vec{\boldsymbol{X}}_{\text{GT}}, \vec{\boldsymbol{Z}}^{(L)}), \tag{11}$$

where λ is the balancing factor between MSE and MMD.

4.4. Efficient Learning on Large Geometric Graphs

The main complexity of FastEGNN lies in the aggregation processes for each node in Eq.(6-7), where the number of the summation operations for all real nodes is $NK + NC$, with K denoting neighbor size. To improve the efficiency for large graphs, one solution is decreasing the value of K, which yet will diminish the interaction between the real nodes. Owing to the involvement of the virtual nodes, in this subsection we will explain that such diminution is alleviated and our FastEGNN will still perform well even when K is decreased to zero, if the distribution of the virtual coordinates is well aligned with the real coordinates.

We focus on the update of the real coordinate $\vec{\boldsymbol{x}}_i$ by aggregating messages from its neighbors. Without loss of generality, we assume its neighbors to be the whole set $\vec{\boldsymbol{X}}$. The update is denoted as the function $\vec{\boldsymbol{x}}_i' = f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{X}})$, which should be E(3)-equivariant *w.r.t.* the both inputs, and permutation-invariant *w.r.t.* $\vec{\boldsymbol{X}}$. According to Proposition 10 in Villar et al. (2021), we have the following result.

Proposition 4.2 ((Villar et al., 2021)). *The update function must take the form $f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{X}}) = \vec{\boldsymbol{x}}_i + \sum_{j=1}^{N}(\vec{\boldsymbol{x}}_j - \vec{\boldsymbol{x}}_i)\psi(\vec{\boldsymbol{x}}_j - \vec{\boldsymbol{x}}_i, \vec{\boldsymbol{x}}_1 - \vec{\boldsymbol{x}}_i, \cdots, \vec{\boldsymbol{x}}_{j-1} - \vec{\boldsymbol{x}}_i, \vec{\boldsymbol{x}}_{j+1} - \vec{\boldsymbol{x}}_i, \cdots, \vec{\boldsymbol{x}}_N - \vec{\boldsymbol{x}}_i)$, where $\psi : \mathbb{R}^{3N} \to \mathbb{R}$ is orthogonality-invariant, and permutation-invariant with respect to the last $N-1$ inputs.*

While this result is theoretically elegant, it is not so practically helpful, since the universal form of ψ is still unknown.

Existing equivariant GNNs (such as EGNN) only exploit its reduced but not sufficiently expressive version.

In our model, we need to aggregate the message from the virtual nodes $\vec{\boldsymbol{Z}}$. The update function becomes $\vec{\boldsymbol{x}}_i' = f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}})$. Notably, different from $\vec{\boldsymbol{X}}$, the matrix $\vec{\boldsymbol{Z}}$ is an ordered set by our design, which entails that the permutation invariance of f is no longer required. This enables us to derive a more informative form of f as follows.

Proposition 4.3. *The update function must take the form* $f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) = \vec{\boldsymbol{x}}_i + \sum_{c=1}^{C}(\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i)\psi_c\left(\bigoplus_{c=1}^{C}\|\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i\|^2, \boldsymbol{m}^v\right)$, *where* $\psi_c : \mathbb{R}^{C+C^2} \to \mathbb{R}$ *is an arbitrary non-linear function, and* $\boldsymbol{m}^v$ *is an E(3)-invariant term by Eq. (5).*

The proof is based on Proposition 1 in Han et al. (2022a), and provided in Theorem A.6. Proposition 4.3 is practically realizable by just implementing ψ_c via MLP, which is actually the form in our FastEGNN (Eq. (5)). More importantly, Proposition 4.3 suggests that $f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}})$ is able to universally approximate the messages from all the real nodes, if the virtual coordinates $\vec{\boldsymbol{Z}}$ can well approximate the real ones $\vec{\boldsymbol{X}}$. As presented in the last subjection, we achieve this goal by penalizing the MMD loss. Our latter experiments will also support the theoretical analyses here, showing that FastEGNN with a small value of C can still perform promisingly even when all edges are dropped, while EGNN behaves poorly.

5. Experiments

In this section, we conduct comprehensive evaluations of FastEGNN on three challenging benchmarks of physical simulation on large geometric graphs. We describe the general experiment setup in § 5.1, and present our main results in § 5.2. Ablation studies are also performed and delivered in § 5.3. Moreover, we generalize FastEGNN by taking other backbones into account in § 6

5.1. Experimental Setups

Datasets. We comprehensively benchmark our FastEGNN on three large-scale simulation scenarios, including:

- N**-body system** (Satorras et al., 2021; Kipf et al., 2018). In this simulation, each system comprises $N = 100$ charged particles with random charge $c_i \in \{0, 1\}$, whose movements are driven by Coulomb force. The graph is constructed in a fully-connected manner and the edge features are the product of the charges c_ic_j for each pair of nodes, following standard practice (Satorras et al., 2021). We use 5000 samples for training, 2000 for validation, and 2000 for testing. The task is to predict the final positions after $\Delta t = 10$ frames given the initial positions and velocities of the particles.
- **Protein MD** (Han et al., 2022b). The protein molecular dynamics dataset is processed from MDAnalysis (Gowers et al., 2016), which depicts a long-range AdK equilibrium MD trajectory (Seyler & Beckstein, 2017). Following previous work (Han et al., 2022b), we model the dynamics of the backbone atoms, leading to 855 nodes per data point. A total number of 55108 edges (on average) are connected between the atoms within a distance cutoff of 10Å. The dataset has been split into train/validation/test sets that contain 2481/827/878 frame pairs respectively, with the time span $\Delta t = 15$.
- **Water-3D** (Sanchez-Gonzalez et al., 2020). We further evaluate FastEGNN on the challenging benchmark Water-3D, a large-scale particle-based fluid simulation dataset generated with smoothed-particle hydrodynamics (SPH). The dataset records the dynamics of water falling in a box, with 1000 trajectories for training, 100 for validation, and 100 for testing. There are on average a large amount of 7806 particles and 94999 edges (on average) in each system, where the edges are connected with a cutoff of 0.035.

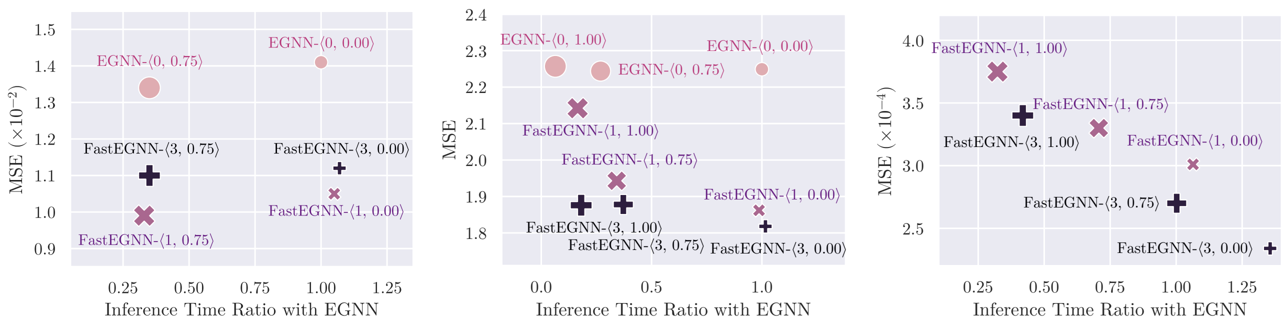
Baselines. We compare FastEGNN with the following baselines: the simplest equivariant model Linear dynamics (Satorras et al., 2021), the non-equivariant Message-Passing Neural Network (MPNN) (Gilmer et al., 2017), the invariant GNN SchNet (Schütt et al., 2018), and the equivariant GNNs including Tensor Field Networks (TFN) (Thomas et al., 2018), Radial Field (RF) (Köhler et al., 2019), and EGNN (Satorras et al., 2021). We also evaluate EGNN*, a variant of EGNN that removes all edges in the graph, as a reference.

Implementation. For our FastEGNN, we implement several instantiations of it by exploring different combinations of the number of virtual nodes C and the edge dropping rate p, denoted as FastEGNN-$\langle C, p\rangle$. We employ the following edge dropping strategy: We sort all edges based on the distance between the connected nodes $\|\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_j\|_2$ and drop the top $p\%$ longest edges. For other hyper-parameters on all three datasets such as the number of layers, hidden dimension, and learning rate, we defer detailed descriptions in Table 5 in Appendix C.2.

Metrics. 1. *MSE*: We use the Mean Squared Error (MSE) between the predicted position and the ground truth on the testing set as the metric to measure the prediction accuracy. **2.** *Relative Time*: To demonstrate the speed-up effect of FastEGNN, we also benchmark the inference time for all models to traverse through the entire testing set, and compute their relative scales *w.r.t.* the inference time of EGNN.

Table 1. MSE and Inference time ratio with EGNN (Satorras et al., 2021) on N-body System ($\sim$ 100 nodes), Protein Dynamics ($\sim$ 800 nodes), and Water-3D ($\sim$ 8000 nodes). FastEGNN-$\langle C, p\rangle$ denotes the model with C virtual nodes and edge dropping rate as p. We also report the results of EGNN* which indicates the EGNN model with all edges dropped. Note that **for Protein Dynamics and Water-3D, all input graphs have already been reduced to sparse local graphs to enable the efficient implementation for all methods**. Since TFN is much more time-consuming than other models, we chose not to test it on the Water-3D data set.

	N-body System		Protein Dynamics		Water-3D	
	MSE ($\times 10^{-2}$)	Relative Time	MSE	Relative Time	MSE ($\times 10^{-4}$)	Relative Time
Linear	12.66	0.01	2.26	0.01	14.06	0.01
MPNN (Gilmer et al., 2017)	3.06	0.62	150.56	0.62	5299.30	0.61
SchNet (Schütt et al., 2018)	24.83	1.39	2.56	1.40	35.02	2.44
RF (Köhler et al., 2019)	5.76	0.26	2.25	0.25	12.94	0.14
TFN (Thomas et al., 2018)	1.62	17.02	2.26	17.81	–	–
EGNN (Satorras et al., 2021)	1.41	1.00	2.25	1.00	6.00	1.00
EGNN* (Satorras et al., 2021)	11.62	0.03	2.26	0.06	12.38	0.11
FastEGNN-$\langle 3,\ 0.00\rangle$	1.12 ± 0.04	1.07	$\mathbf{1.82 \pm 0.02}$	1.39	$\mathbf{2.49 \pm 0.14}$	1.36
FastEGNN-$\langle 3,\ 0.75\rangle$	$\mathbf{1.10 \pm 0.10}$	0.35	1.88 ± 0.01	0.96	2.83 ± 0.13	1.00
FastEGNN-$\langle 3,\ 1.00\rangle$	9.52 ± 0.02	0.15	1.88 ± 0.01	0.80	3.40 ± 0.04	0.42



(a) N-body System (b) Protein Dynamics (c) Water-3D

Figure 3. Ablation studies on the number of virtual nodes C and the edge dropping rate p. We eliminate the results of all methods when $p = 1$ on N-body and the performance of EGNN on Water-3D, since these values are poor to display along with the reported ones. Besides, FastEGNN with 10 virtual nodes is also evaluated, with the results provided in Table 6 of Appendix.

5.2. Main Results

The main quantitative results are presented in Table 1. We have the following observations: **1.** Our FastEGNN yields the lowest simulation error on all three large-scale benchmarks, consistently outperforming the competitive baselines by a significant margin. For instance, FastEGNN yields 28% and 24% improvement in terms of MSE over the best-performed baseline EGNN on N-body system and protein MD datasets, respectively. On the most challenging dataset Water-3D with an average of 7806 particles in each system, FastEGNN reaches a remarkably low simulation error of 3.40×10^{-4}, as opposed to EGNN with an MSE of 6.00×10^{-4}. The strong results unanimously demonstrate the superiority of FastEGNN in learning to simulate physical dynamics and especially scaling to large and complicated systems. **2.** The equivariant GNNs including our FastEGNN generally perform better than the non-equivariant MPNN and the invariant GNN SchNet, demonstrating the importance of preserving physical symmetry for simulation. **3.** With the help of edge sampling, FastEGNN is able to conduct inference in a substantially faster manner. Such effect is evident in the Relative Time metric, where FastEGNN with an edge dropping rate of 0.75 or 1.00 increase the inference speed. Notably, FastEGNN with 75% of the edges dropped delivers the lowest MSE on N-body system while only using 0.340 of the inference time of EGNN.

4. When all edges are preserved, FastEGNN performs favorably across all datasets compared with EGNN, thanks to the proposed virtual node learning technique that boosts the model expressivity. Interestingly, we also observe the performance is only slightly influenced even when a large proportion of the edges are dropped. For example on Protein Dynamics, FastEGNN-$\langle 3, 1.00\rangle$ with all edges dropped still obtains an MSE of 1.88, close to 1.82 when no edges are dropped, remarkably lower than the MSE of 2.25 produced by EGNN. By contrast, EGNN* performs poorly in all cases, with performance close to Linear Dynamics. Overall, the results show that our virtual learning enhances the expres-

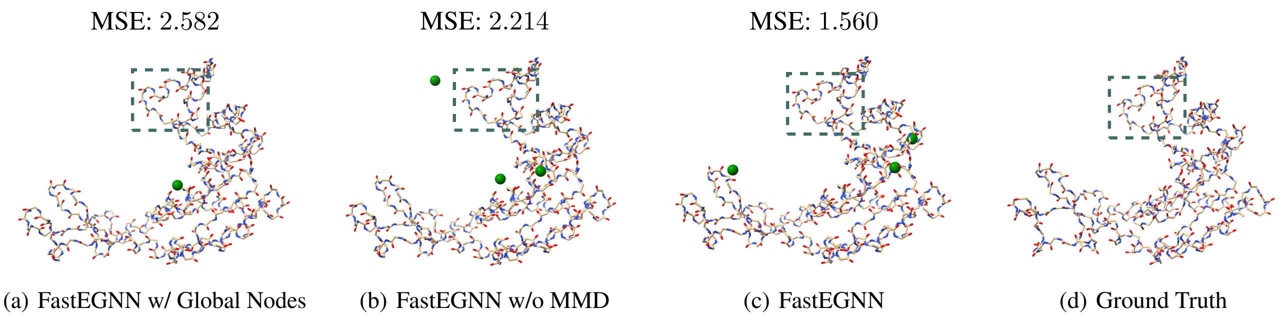


Figure 4. Visualization of different variants of FastEGNN on Protein Dynamics. We illustrate the predicted position of the protein, as well as the learned location of the virtual nodes (denoted in green). The dashed box marks highlight the region of interest where FastEGNN yields more accurate prediction. For FastEGNN w/ Global Nodes, the learned coordinates of 3 virtual nodes are almost the same.

Table 2. Ablation studies on the design of virtual node learning. Experiments are conducted on protein MD dataset and all variants of FastEGNN in this experiment contains 3 virtual nodes. Results are collected with edge dropping rates as 0.5, 0.75, and 1. The performance of EGNN is also reported.

	0.5	0.75	1
EGNN	2.249	2.244	2.257
FastEGNN w/ Global Nodes	1.888	1.967	2.121
FastEGNN w/o MMD	1.882	1.883	1.923
FastEGNN	**1.863**	**1.878**	**1.876**

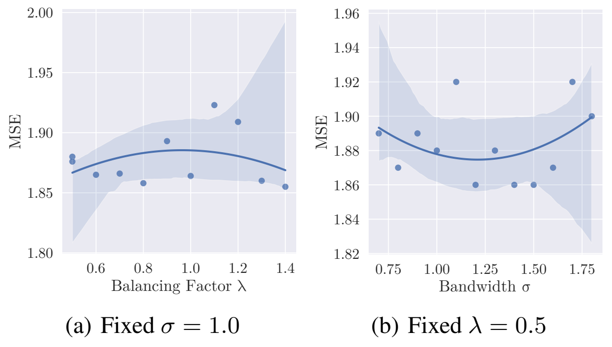


Figure 5. Influence of different hyper-parameters of the MMD loss on protein MD dataset with the model FastEGNN-$\langle 3, 0.75\rangle$.

sivity of the model and improves the performance, while also enabling edge dropping for higher inference speed with minimal sacrifice in prediction accuracy.

5.3. Ablation Studies

The number of virtual nodes. We investigate the effect of the number of virtual nodes together with the edge dropping rate on N-body system and protein MD datasets. We sweep the number of virtual nodes in $\{1, 3, 10\}$ and the edge dropping rate in $\{0.00, 0.75, 1.00\}$, with the results illustrated in Fig. 3. We also involve the results of EGNN with the same edge dropping rates for reference. It is clear that when using the same edge dropping rate, FastEGNN consistently outperforms EGNN, with the aid of virtual node modeling. Interestingly, we find that using 1 virtual node already leads to the best performance on N-body, while the lowest error is achieved with 3 virtual nodes on Protein MD dataset and Water-3D, due to the higher complexity and broader spatial coverage of protein MD and Water-3D. Moreover, when keeping the same edge dropping rate, FastEGNN with virtual nodes only adds very little inference time over EGNN, thanks to our carefully designed message computation and aggregation scheme for the virtual nodes.

Modeling virtual nodes as ordered set. We have proposed to model the virtual nodes as an ordered set instead of an unordered set, which enforces mutual distinctiveness as an inductive bias. We demonstrate the validity of such design in Table 2, where we implement a variant of FastEGNN that treats virtual nodes as global nodes with permutation-equivariant message passing in between. We discover that naively viewing them as global nodes leads to worse performance and indistinguishable assignment of the virtual nodes (Fig. 4(a)), with MSE increases from 1.888 to 2.121 while the dropping rate increases from 0.5 to 1. Our approach instead empowers the virtual nodes with orders and different roles, which generally yields lower MSE.

The impact of the MMD loss. From Table 2, we observe that without enforcing the MMD loss during training, FastEGNN becomes less effective in encouraging the diverse roles of the virtual nodes and thus incurs larger error. The effect of MMD is further illustrated in Fig. 4, where the learned positions of the virtual nodes without MMD loss fail to capture the underlying geometric structure of the protein, while our FastEGNN produces reasonable allocation of the virtual nodes over spatial distribution of the data.

Table 3. MSE of EGNN (Satorras et al., 2021), RF (Köhler et al., 2019), TFN (Thomas et al., 2018), SchNet (Schütt et al., 2018) and their enhanced models involving virtual nodes learning.

	MSE				
Dropping Rate	0.00	0.50	0.75	0.90	1.00
EGNN (Satorras et al., 2021)	2.25	2.25	2.24	2.25	2.26
FastEGNN-1	1.86	1.91	1.94	1.92	2.14
FastEGNN-3	1.82	1.86	1.88	1.86	1.88
FastEGNN-10	1.87	1.89	1.95	1.89	1.97
RF (Köhler et al., 2019)	2.25	2.26	2.26	2.26	2.26
FastRF-3	2.07	2.02	2.02	2.02	2.05
TFN (Thomas et al., 2018)	2.25	2.26	2.26	2.26	–
FastTFN-3	1.84	2.01	1.90	2.06	–
SchNet (Schütt et al., 2018)	2.56	2.57	2.56	2.56	2.60
FastSchNet-3	1.99	2.01	2.01	2.06	2.08

Sensitivity analysis. We study the sensitivity of the model *w.r.t.* the hyper-parameters in the MMD loss, namely σ, which is the bandwidth in the MMD kernel, and λ, which is the balancing factor in the hybrid loss. The results are depicted in Fig. 5. We observe that our FastEGNN in general remains non-sensitive to the choices of both σ and λ, with the MSEs being generally lower than 1.9, outperforming other models by a significant margin.

6. More Backbones Besides EGNN

We explore the generality of our method by further introducing virtual nodes to three-widely used GNNs, including RF, TFN and SchNet, leading to FastRF, FastTFN and FastSchNet. The implementation details are deferred to Appendix B. The enhanced models are evaluated on the Protein-Dynamics dataset under different edge dropping rates. Results in Table 3 demonstrate that FastRF-3, FastTFN-3 and FastSchNet-3 (with 3 virtual nodes) remarkably outperform RF, TFN and SchNet in all cases. Overall, FastEGNN-3 still achieves the best performance, probably because the EGNN architecture is more suitable for this task, compared to RF, TFN and SchNet.

7. Conclusion

We propose FastEGNN, an advanced model that operates on large geometric graphs. The core insight lies in constructing an ordered set of virtual nodes to perform expressive message passing that enjoys both distinctiveness and distributedness. The entire framework is also guaranteed the critical E(3)-equivariance for enhanced performance. Comprehensive evaluations on 100-body simulation, protein molecular dynamics, and particle-based fluid simulation Water-3D consistently demonstrate the superiority of FastEGNN in terms of achieving remarkably lower simulation error and significant speed-up due to sparsification.

Impact Statement

The proposed method and experimental discovery by this paper are able to advance the field of machine learning (Ren et al., 2023; Zhou et al., 2024; Jiang et al., 2023; Xie et al., 2022; Ng et al., 2024). We do not recognize any significant negative societal impact of this work that should be highlighted here.

Acknowledgments

This work was jointly supported by the following projects: the National Science and Technology Major Project under Grant 2020AAA0107300, the National Natural Science Foundation of China (No. 62376276); Beijing Nova Program (No. 20230484278); Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100020098); Ant Group through CCF-Ant Research Fund (CCF-AFSG RF20220204); the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (23XNKJ19); Public Computing Cloud, Renmin University of China.

References

Borgwardt, K. M., Gretton, A., Rasch, M. J., Kriegel, H.-P., Schölkopf, B., and Smola, A. J. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J., and Welling, M. Geometric and physical quantities improve e(3) equivariant message passing. In *ICLR*, 2022.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

Darve, E. The fast multipole method: numerical implementation. *Journal of Computational Physics*, 160(1): 195–240, 2000.

Duval, A., Mathis, S. V., Joshi, C. K., Schmidt, V., Miret, S., Malliaros, F. D., Cohen, T., Lio, P., Bengio, Y., and Bronstein, M. A hitchhiker's guide to geometric gnns for 3d atomic systems. *arXiv preprint arXiv:2312.07511*, 2023.

Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *NeurIPS*, volume 33, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Gowers, R. J., Linke, M., Barnoud, J., Reddy, T. J., Melo, M. N., Seyler, S. L., Domanski, J., Dotson, D. L., Buchoux, S., Kenney, I. M., et al. Mdanalysis: a python package for the rapid analysis of molecular dynamics simulations. In *Proceedings of the 15th python in science conference*, volume 98, pp. 105. SciPy Austin, TX, 2016.

Han, J., Huang, W., Ma, H., Li, J., Tenenbaum, J. B., and Gan, C. Learning physical dynamics with subequivariant graph neural networks. In *Advances in Neural Information Processing Systems*, 2022a.

Han, J., Huang, W., Xu, T., and Rong, Y. Equivariant graph hierarchy-based neural networks. In *NeurIPS*, 2022b.

Han, J., Rong, Y., Xu, T., and Huang, W. Geometrically equivariant graph neural networks: A survey. *arXiv preprint arXiv:2202.07230*, 2022c.

Han, J., Cen, J., Wu, L., Li, Z., Kong, X., Jiao, R., Yu, Z., Xu, T., Wu, F., Wang, Z., et al. A survey of geometric graph neural networks: Data structures, models and applications. *arXiv preprint arXiv:2403.00485*, 2024.

Huang, W., Han, J., Rong, Y., Xu, T., Sun, F., and Huang, J. Equivariant graph mechanics networks with constraints. In *ICLR*, 2022.

Hwang, E., Thost, V., Dasgupta, S. S., and Ma, T. An analysis of virtual nodes in graph neural networks for link prediction. In *The First Learning on Graphs Conference*, 2022.

Ingraham, J. B., Baranov, M., Costello, Z., Barber, K. W., Wang, W., Ismail, A., Frappier, V., Lord, D. M., Ng-Thow-Hing, C., Van Vlack, E. R., et al. Illuminating protein space with a programmable generative model. *Nature*, pp. 1–9, 2023.

Jiang, N.-F., Zhao, X., Zhao, C.-Y., An, Y.-Q., Tang, M., and Wang, J.-Q. Pruning-aware sparse regularization for network pruning. *Machine Intelligence Research*, 20(1): 109–120, 2023.

Jing, B., Eismann, S., Suriana, P., Townshend, R. J. L., and Dror, R. Learning from protein structure with geometric vector perceptrons. In *ICLR*, 2021.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *ICML*, 2018.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *ICLR*, 2020.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

Kong, X., Huang, W., and Liu, Y. Conditional antibody design as 3d equivariant graph translation. In *The Eleventh International Conference on Learning Representations*, 2023.

Ng, M. K., Wu, H., and Yip, A. Stability and generalization of hypergraph collaborative networks. *Machine Intelligence Research*, 21(1):184–196, 2024.

Pham, T., Tran, T., Dam, H., and Venkatesh, S. Graph classification via deep learning with virtual nodes. *arXiv preprint arXiv:1708.04357*, 2017.

Ren, W.-Q., Qu, Y.-B., Dong, C., Jing, Y.-Q., Sun, H., Wu, Q.-H., and Guo, S. A survey on collaborative dnn inference for edge intelligence. *Machine Intelligence Research*, 20(3):370–395, 2023.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Satorras, V. G., Hoogeboom, E., and Welling, M. E(n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., and Müller, K.-R. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

Sestak, F., Schneckenreiter, L., Hochreiter, S., Mayr, A., and Klambauer, G. Vn-egnn: Equivariant graph neural networks with virtual nodes enhance protein binding site identification. In *NeurIPS 2023 Workshop: New Frontiers in Graph Learning*, 2023.

Seyler, S. and Beckstein, O. Molecular dynamics trajectory for benchmarking mdanalysis. *URL: https://figshare. com/articles/Molecular_dynamics_ trajectory_for_benchmarking_MDAnalysis/5108170, doi*, 10:m9, 2017.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Villar, S., Hogg, D. W., Storey-Fisher, K., Yao, W., and Blum-Smith, B. Scalars are universal: Equivariant machine learning, structured like classical physics. In *NeurIPS*, 2021.

Watson, J. L., Juergens, D., Bennett, N. R., Trippe, B. L., Yim, J., Eisenach, H. E., Ahern, W., Borst, A. J., Ragotte, R. J., Milles, L. F., et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976): 1089–1100, 2023.

Wu, L., Hou, Z., Yuan, J., Rong, Y., and Huang, W. Equivariant spatio-temporal attentive graph networks to simulate physical dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

Xie, J., Liu, S.-Y., and Chen, J.-X. A framework for distributed semi-supervised learning using single-layer feedforward networks. *Machine Intelligence Research*, 19(1): 63–74, 2022.

Xu, M., Han, J., Lou, A., Kossaifi, J., Ramanathan, A., Azizzadenesheli, K., Leskovec, J., Ermon, S., and Anandkumar, A. Equivariant graph neural operator for modeling 3d dynamics. *arXiv preprint arXiv:2401.11037*, 2024.

Zhou, H., Ren, J., Deng, H., Cheng, X., Zhang, J., and Zhang, Q. Interpretability of neural networks based on game-theoretic interactions. *Machine Intelligence Research*, pp. 1–22, 2024.

A. Proof

Theorem A.1 (Propostion 4.1)**.** *If the initialization of the virtual nodes satisfies Eq. (2), then after Eqs. (3) to (9), the output coordinates $\vec{\boldsymbol{X}}^{(L)}$ are E(3)-equivariant and permutation-equivaraint, the virtual coordinates $\vec{\boldsymbol{Z}}^{(L)}$ are E(3)-equivariant and permutation-invariant, with respect to the input $\vec{\boldsymbol{X}}^{(0)}$.*

Proof. Consider a sequence composed of functions $\{\phi_i : \mathcal{X}^{(i-1)} \to \mathcal{X}^{(i)}\}_{i=1}^{N}$ equivariant to a same group G, the equivariance lead to an interesting property that

$$\phi_N \circ \cdots \circ \phi_{i+1} \circ \rho_{\mathcal{X}^{(i)}}(g)\phi_i \circ \cdots \circ \phi_1 = \phi_N \circ \cdots \circ \phi_{j+1} \circ \rho_{\mathcal{X}^{(j)}}(g)\phi_j \circ \cdots \circ \phi_1,$$

holds for all $i, j = 1, 2, \dots, N$ and $g \in G$, which means that the group elements g can be freely exchanged in the composite sequence of equivariant functions. In particular, if one of the equivariant functions (*e.g.* ϕ_k) is replaced by an invariant function, the group element g will be absorbed, that means

$$\phi_N \circ \cdots \circ \phi_k \circ \cdots \circ \phi_{i+1} \circ \rho_{\mathcal{X}^{(i)}}(g)\phi_i \circ \cdots \circ \phi_1 = \phi_N \circ \cdots \circ \phi_1.$$

holds for all $g \in G$ but only $i = 1, 2, \dots, k$. Although $\phi_N \circ \cdots \circ \phi_k$ is still equivariant, because the group elements must be input starting from ϕ_1, the overall $\phi_N \circ \cdots \circ \phi_1$ is still an invariant function.

Since Eq. (2) is the definition of a function with $\mathrm{E}(3)$-equivariance and permutation-invaraince, and the pooling operators like `sum` or `mean` maintain permutation-equivariance of the whole model. Now to prove the $\mathrm{E}(3)$-equivarianceof the whole model, we choose to give a stronger conclusion, namely each of Eqs. (3) to (5), (7) and (9) is an $\mathrm{E}(3)$-invariant function, and each of Eqs. (6) and (8) is an $\mathrm{E}(3)$-equivariant function.

It is obvious that each input in Eqs. (3) to (5), (7) and (9) is either a constant or an inner product term, thus ensuring the $\mathrm{E}(3)$-invariance of each function. And each of Eqs. (6) and (8) is linear combination of three-dimensional vectors with an $\mathrm{E}(3)$-invariant weight, thus ensuring the $\mathrm{E}(3)$-equivariance of each function. □

Theorem A.2. *The $\mathrm{E}(3)$-equivaraint function $f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}})$ in Theorem 4.3 can be decomposed into an $\mathrm{O}(3)$-equivaraint and translation-invariant function of $\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i$, and the addition of $\vec{\boldsymbol{x}}_i$.*

Proof. Since $f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}})$ is equivaraint to translation $\vec{\boldsymbol{t}}$, considering $\vec{\boldsymbol{t}} = -\vec{\boldsymbol{x}}_i$, we get

$$f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) = f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) - \vec{\boldsymbol{x}}_i + \vec{\boldsymbol{x}}_i = f(\vec{\boldsymbol{x}}_i - \vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) + \vec{\boldsymbol{x}}_i = f(\vec{\boldsymbol{0}}, \vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) + \vec{\boldsymbol{x}}_i := h(\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) + \vec{\boldsymbol{x}}_i.$$

□

Lemma A.3. *For any O(3)-equivariant function $\hat{f}(\vec{\boldsymbol{Z}})$, it must fall into the subspace spanned by the columns of $\vec{\boldsymbol{Z}}$, namely, there exists a function $s(\vec{\boldsymbol{Z}})$, satisfying $\hat{f}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}s(\vec{\boldsymbol{Z}})$.*

Proof. The proof is given by (Villar et al., 2021). Essentially, suppose $\vec{\boldsymbol{Z}}^{\perp}$ is the orthogonal complement of the column space of $\vec{\boldsymbol{Z}}$. Then there must exit functions $s(\vec{\boldsymbol{Z}})$ and $s^{\perp}(\vec{\boldsymbol{Z}})$, satisfying $\hat{f}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}s(\vec{\boldsymbol{Z}}) + \vec{\boldsymbol{Z}}^{\perp}s^{\perp}(\vec{\boldsymbol{Z}})$. We can always find an orthogonal transformation $\boldsymbol{O}$ allowing $\boldsymbol{O}\vec{\boldsymbol{Z}} = \vec{\boldsymbol{Z}}$ while $\boldsymbol{O}\vec{\boldsymbol{Z}}^{\perp} = -\vec{\boldsymbol{Z}}^{\perp}$. With this transformation $\boldsymbol{O}$, we have $\hat{f}(\boldsymbol{O}\vec{\boldsymbol{Z}}) = \hat{f}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}s(\vec{\boldsymbol{Z}}) + \vec{\boldsymbol{Z}}^{\perp}s^{\perp}(\vec{\boldsymbol{Z}})$, and $\boldsymbol{O}\hat{f}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}s(\vec{\boldsymbol{Z}}) - \vec{\boldsymbol{Z}}^{\perp}s^{\perp}(\vec{\boldsymbol{Z}})$. The equivariance property of $\hat{f}$ implies $\vec{\boldsymbol{Z}}s(\vec{\boldsymbol{Z}}) + \vec{\boldsymbol{Z}}^{\perp}s^{\perp}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}s(\vec{\boldsymbol{Z}}) - \vec{\boldsymbol{Z}}^{\perp}s^{\perp}(\vec{\boldsymbol{Z}})$, which derives $s^{\perp}(\vec{\boldsymbol{Z}}) = 0$. Hence, the proof is concluded. □

Lemma A.4. *If the O(3)-equivariant function $\hat{f}(\vec{\boldsymbol{Z}})$ lies in the subspace spanned by the columns of $\vec{\boldsymbol{Z}}$, then there exists a function σ satisfying $\hat{f}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}\sigma(\vec{\boldsymbol{Z}}^{\top}\boldsymbol{Z})$.*

Proof. The proof is provided by Corollary 2 in (Huang et al., 2022). The basic idea is that $\hat{f}(\vec{\boldsymbol{Z}})$ can be transformed to $\hat{f}(\vec{\boldsymbol{Z}}) = \vec{\boldsymbol{Z}}\eta(\vec{\boldsymbol{Z}})$ where $\eta(\vec{\boldsymbol{Z}})$ is O(3)-invariant. According to Lemma 1-2 in (Huang et al., 2022), $\eta(\vec{\boldsymbol{Z}})$ must be written as $\eta(\vec{\boldsymbol{Z}}) = \sigma(\vec{\boldsymbol{Z}}^{\top}\vec{\boldsymbol{Z}})$, which completes the proof. □

Theorem A.5. *There exists a surjection from dao to $(\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)^{\top}(\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)$, that is*

$$\exists \text{ a funcion } \zeta,\ (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)^{\top}(\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) = \zeta\left(\bigoplus_{c=1}^{C} \|\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i\|^2, (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}})^{\top}(\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}})\right).$$

Proof. First, we simplify the notation as

$$\boldsymbol{A} \coloneqq (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)^\top (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i), \quad \boldsymbol{D} \coloneqq (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}})^\top (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}}), \quad \boldsymbol{Y} \coloneqq \bigoplus_{c=1}^{C} \|\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i\|^2.$$

Since $\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i = (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}}) - (\vec{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})$, letting $\vec{\boldsymbol{Z}}' = \vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}}$, $\vec{\boldsymbol{x}}' = \vec{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}$, $\vec{\boldsymbol{z}}'_c = \vec{\boldsymbol{Z}}'_{:,c}$, we have

$$\boldsymbol{A} = (\vec{\boldsymbol{Z}}' - \vec{\boldsymbol{x}}')^\top (\vec{\boldsymbol{Z}}' - \vec{\boldsymbol{x}}'), \quad \boldsymbol{D} = \vec{\boldsymbol{Z}}'^\top \vec{\boldsymbol{Z}}', \quad \boldsymbol{Y} = \bigoplus_{c=1}^{C} \|\vec{\boldsymbol{z}}'_c - \vec{\boldsymbol{x}}'\|^2.$$

Thus, we find the following equations:

$$\begin{aligned}
\boldsymbol{A}_{mn} &= \langle \vec{\boldsymbol{z}}'_m - \vec{\boldsymbol{x}}',\ \vec{\boldsymbol{z}}'_n - \vec{\boldsymbol{x}}' \rangle \\
&= \langle \vec{\boldsymbol{z}}'_m,\ \vec{\boldsymbol{z}}'_n \rangle + \frac{1}{2} \left(\langle \vec{\boldsymbol{x}}',\ \vec{\boldsymbol{x}}' - 2\vec{\boldsymbol{z}}'_m \rangle + \langle \vec{\boldsymbol{x}}',\ \vec{\boldsymbol{x}}' - 2\vec{\boldsymbol{z}}'_n \rangle \right) \\
&= \langle \vec{\boldsymbol{z}}'_m,\ \vec{\boldsymbol{z}}'_n \rangle + \frac{1}{2} \left(\left(\|\vec{\boldsymbol{z}}'_m - \vec{\boldsymbol{x}}'\|^2 - \|\vec{\boldsymbol{z}}'_m\|^2 \right) + \left(\|\vec{\boldsymbol{z}}'_n - \vec{\boldsymbol{x}}'\|^2 - \|\vec{\boldsymbol{z}}'_n\|^2 \right) \right) \\
&= \boldsymbol{D}_{mn} + \frac{1}{2} \left((\boldsymbol{Y}_m - \boldsymbol{D}_{mm}) + (\boldsymbol{Y}_n - \boldsymbol{D}_{nn}) \right),
\end{aligned}$$

and

$$\begin{aligned}
\boldsymbol{Y}_m &= \boldsymbol{A}_{mm}, \\
\boldsymbol{D}_{mn} &= \boldsymbol{A}_{mn} - \frac{1}{2}(\boldsymbol{A}_{mm} + \boldsymbol{A}_{nn}) + \frac{1}{2}(\boldsymbol{D}_{mm} + \boldsymbol{D}_{nn}),
\end{aligned}$$

which means there is a a surjection from $\bigoplus_{c=1}^{C} \|\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i\|^2, (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}})^\top (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}})$ to $(\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)^\top (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)$.

□

Theorem A.6 (Propostion 4.3)**.** *The update function must take the form $f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) = \vec{\boldsymbol{x}}_i + \sum_{c=1}^{C} (\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i) \psi_c \left(\bigoplus_{c=1}^{C} \|\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i\|^2, \boldsymbol{m}^v \right)$, where $\psi_c : \mathbb{R}^{C+C^2} \to \mathbb{R}$ is an arbitrary non-linear function, and $\boldsymbol{m}^v$ is an E(3)-invariant term by Eq.* (5).

Proof. According to Theorem A.2, there exists an $\mathrm{O}(3)$-equivaraint and transform-invariant function h, thus

$$f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) = h(\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) + \vec{\boldsymbol{x}}_i.$$

Then, from Lemma A.4, we rewrite the $\mathrm{O}(3)$-equivarint h into a product of the vector $\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i$ and a scalar function η with an inner-product input, that is,

$$f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) = (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) \eta \left((\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i)^\top (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) \right) + \vec{\boldsymbol{x}}_i.$$

Finally, with Theorem A.5, we transform it into a more expressive function as

$$f(\vec{\boldsymbol{x}}_i, \vec{\boldsymbol{Z}}) = (\vec{\boldsymbol{Z}} - \vec{\boldsymbol{x}}_i) \psi \left(\bigoplus_{c=1}^{C} \|\vec{\boldsymbol{z}}_c - \vec{\boldsymbol{x}}_i\|^2, (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}})^\top (\vec{\boldsymbol{Z}} - \bar{\boldsymbol{x}}) \right) + \vec{\boldsymbol{x}}_i,$$

which is equivalent to Eq. (5). □

B. Generality Analysis

We introduce virtual nodes to RF, TFN and SchNet to explore the generality of our proposed method. Here is implementation details.

B.1. Implementation of FastRF

For FastRF, the implementation is similar to FastEGNN by only removing the update of hidden features $\boldsymbol{h}$.

B.2. Implementation of FastTFN

To implement FastTFN, we first reduce the number of the channels used in the original TFN to be 1, yielding single-channel TFN, which performs almost the same as the original TFN but is found to be more compatible with virtual node learning. Upon single-channel TFN, we design 4 blocks in total. In each block, we first employ one TFN layer to obtain the updated real coordinates, which are then fed to the following equations to combine messages from virtual nodes:

$$\vec{\boldsymbol{x}}_i^{(l+1)} = \left[\vec{\boldsymbol{x}}_i^{(l)} + \sum_{j\in\mathcal{N}_i} \mathbb{Y}^{\mathbb{L}}\left(\frac{\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{x}}_j^{(l)}}{\|\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{x}}_j^{(l)}\|}\right) \otimes_{\text{cg}}^{\mathbb{W}} \left(\boldsymbol{h}_j^{(l)} \| \vec{\boldsymbol{v}}_j^{(0)}\right)\right]_{\texttt{upd\_by\_TFN}} + \left[\frac{1}{C}\sum_{c=1}^{C}(\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{z}}_c^{(l)})\varphi_x^v(\boldsymbol{m}_{ic}^v)\right],$$

where the $\vec{\boldsymbol{x}}_i^{(l)}$ is the coordinate of real node i, and $\mathbb{Y}^{\mathbb{L}}(\cdot)$ embeds the distance between real node i and j into spherical harmonics with type in set $\mathbb{L} = \{0, 1, 2, \ldots, \texttt{max\_degree}\}$. $\otimes_{\text{cg}}^{\mathbb{W}}(\cdot,\cdot)$ means Clebsch-Gordan (CG) tensor product with $\mathbb{W}$ being the set of weight (Han et al., 2024), $\boldsymbol{h}_j$ and $\vec{\boldsymbol{v}}_j$ are features (type-0) and velocity (type-1) of real node j respectively. Additionally, the update formula for virtual nodes is same as FastEGNN in Eq. (8).

B.3. Implementation of FastSchNet

The design of FastSchNet is similar as FastTFN. It also leverages the coordinates updated by SchNet and messages from virtual nodes to update the real nodes, which is:

$$\vec{\boldsymbol{x}}_i^{(l+1)} = \left[\vec{\boldsymbol{x}}_i^{(l)} + \sum_{j\in\mathcal{N}_i} \sigma(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}, \|\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{x}}_j^{(l)}\|)(\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{x}}_j^{(l)})\right]_{\texttt{upd\_by\_SchNet}} + \left[\frac{1}{C}\sum_{c=1}^{C}(\vec{\boldsymbol{x}}_i^{(l)} - \vec{\boldsymbol{z}}_c^{(l)})\varphi_x^v(\boldsymbol{m}_{ic}^v)\right],$$

The update of virtual nodes is also in Eq. (8).

C. Experiment Details

C.1. Dataset Details

Table 4. Basic parameters of three datasets. It is worth noting that only the N-body task uses a fully connected method to construct the graph, and other tasks have initially deleted some edges. We counted the number (or average) of samples, nodes and edges of all tasks, and converted them into an *Initial Cutoff Rate* item.

	N-body System	Protein Dynamics	Water-3D
Number of Samples (Train / Valid / Test)	$5,000/2,000/2,000$	$2,481/827/863$	$15,000/1,500/1,500$
Number of Nodes (N)	100	855	$7,806$ (Avg)
Number of Edges ($\|\mathcal{E}\|$)	$9,900$	$55,107$ (Avg)	$94,999$ (Avg)
Initial Cutoff Rate ($\eta_0 = (1 - \|\mathcal{E}\|/N(N-1)) \times 100\%$)	0.00%	92.45% (Avg)	99.82% (Avg)

C.2. Implementation Details

Table 5. Hyper-parameters of FastEGNN

Hyperparameter	N-body System	Protein Dynamics	Water-3D
Bandwidth σ in Eq. (10)	1.5	1.0	1.5
Balancing factor λ in Eq. (11)	0.03	0.50	0.01
Number of Virtual Channel C	$\{1,3,10\}$	$\{1,3,10\}$	$\{1,3,10\}$
Learning Rate	5e-4	5e-4	5e-4
Weight Decay	1e-12	1e-12	1e-12
Epochs	2500	800	800
Initial Cutoff Threshold	`fully_connected`	$\lambda = 10$Å	$R = 0.035$
Initial Cutoff Rate (η_0)	0	0.9245	0.9982
Dropping Edge Rate (p)	$\{0, 0.75, 1\}$	$\{0, 0.75, 1\}$	$\{0, 0.75, 1\}$
Total Cutoff Rate ($1-(1-\eta_0)(1-\eta)$)	$\{0, 0.75, 1\}$	$\{0.9245, 0.9811, 1\}$	$\{0.9982, 9996, 1\}$

C.3. More Experiment Results

Table 6. MSE and time-consuming ratio with EGNN (Satorras et al., 2021) on N-body System, Molecular Dynamics on Proteins and Physical Simulation. FastEGNN-$\langle C, p\rangle$ here means the experiment is set with corresponding hyper-parameters.

	N-body System		Protein Dynamics		Water-3D	
	MSE ($\times 10^{-2}$)	Relative Time	MSE	Relative Time	MSE ($\times 10^{-4}$)	Relative Time
Linear	12.66	0.01	2.26	0.01	14.06	0.01
MPNN (Gilmer et al., 2017)	3.06	0.62	150.56	0.62	5299.30	0.61
SchNet (Schütt et al., 2018)	24.83	1.39	2.56	1.40	35.02	2.44
RF (Köhler et al., 2019)	5.76	0.26	2.25	0.25	12.94	0.14
GVP-GNN (Jing et al., 2021)	7.59	2.17	–	2.19	–	3.84
TFN (Thomas et al., 2018)	1.62	17.02	2.26	17.81	–	–
EGNN (Satorras et al., 2021)	1.41	1.00	2.25	1.00	6.00	1.00
EGNN-$\langle 0,\ 0.75\rangle$	1.34	0.29	2.24	0.27	9.87	0.33
EGNN-$\langle 0,\ 1.00\rangle$	11.62	0.03	2.26	0.06	12.38	0.11
FastEGNN-$\langle 1,\ 0.00\rangle$	1.05	1.05	1.86	1.35	3.01	1.06
FastEGNN-$\langle 1,\ 0.75\rangle$	0.99	0.33	1.94	0.96	3.30	0.71
FastEGNN-$\langle 1,\ 1.00\rangle$	9.72	0.13	2.14	0.76	3.75	0.32
FastEGNN-$\langle 3,\ 0.00\rangle$	1.12	1.07	1.82	1.39	2.49	1.36
FastEGNN-$\langle 3,\ 0.75\rangle$	1.10	0.35	1.88	0.96	2.83	1.00
FastEGNN-$\langle 3,\ 1.00\rangle$	9.52	0.15	1.88	0.80	3.40	0.42
FastEGNN-$\langle 10,\ 0.00\rangle$	1.19	1.15	1.87	1.44	–	–
FastEGNN-$\langle 10,\ 0.75\rangle$	0.99	0.42	1.95	0.97	–	–
FastEGNN-$\langle 10,\ 1.00\rangle$	9.38	0.19	1.97	0.80	–	–

Table 7. We further introduce virtual nodes to RF (Köhler et al., 2019), SchNet (Schütt et al., 2018) and TFN (Thomas et al., 2018) to explore the generality of our method. We call them FastRF-`channels`, FastSchNet-`channels` and FastTFN-`channels` respectively. Together with former mentioned FastEGNN and all base models, we report their MSE and time-comsuming ratio compared with EGNN (Satorras et al., 2021) on Protein Dynamics Dataset in this table. Experiments show that our virtual node strategy enhances the performance of all these three models under all Dropping Rates, thus verify its effectiveness.

MSE

Dropping Rate	0.00	0.50	0.75	0.90	1.00
Linear	2.26	2.26	2.26	2.26	2.26
MPNN (Gilmer et al., 2017)	150.56	546.76	804.37	738.33	269.84
EGHN (Han et al., 2022b)	1.84	2.02	2.08	2.24	2.24
EGNN (Satorras et al., 2021)	2.25	2.25	2.24	2.25	2.26
FastEGNN-1	1.86	1.91	1.94	1.92	2.14
FastEGNN-3	1.82	1.86	1.88	1.86	1.88
FastEGNN-10	1.87	1.89	1.95	1.89	1.97
RF (Köhler et al., 2019)	2.25	2.26	2.26	2.26	2.26
FastRF-3	2.07	2.02	2.02	2.02	2.05
SchNet (Schütt et al., 2018)	2.56	2.57	2.56	2.56	2.60
FastSchNet-3	1.99	2.01	2.01	2.06	2.08
TFN (Thomas et al., 2018)	2.25	2.26	2.26	2.26	–
FastTFN-3	1.84	2.01	1.90	2.06	–

Relative Time

Dropping Rate	0.00	0.50	0.75	0.90	1.00
Linear	0.01	0.01	0.01	0.01	0.01
MPNN (Gilmer et al., 2017)	0.62	0.32	0.17	0.08	0.03
EGHN (Han et al., 2022b)	4.90	4.52	4.35	4.16	4.26
EGNN (Satorras et al., 2021)	1.00	0.51	0.27	0.12	0.06
FastEGNN-1	1.35	1.08	0.96	0.92	0.76
FastEGNN-3	1.39	1.08	0.96	0.92	0.80
FastEGNN-10	1.44	1.10	0.97	0.92	0.80
RF (Köhler et al., 2019)	0.25	0.13	0.07	0.05	0.04
FastRF-3	0.94	0.52	0.31	0.19	0.11
SchNet (Schütt et al., 2018)	1.40	0.73	0.40	0.20	0.08
FastSchNet-3	2.17	1.52	1.24	1.12	1.00
TFN (Thomas et al., 2018)	2.50	1.83	1.63	1.55	–
FastTFN-3	3.17	2.40	2.16	2.05	–

Table 8. We implement VN-EGNN as proposed by Sestak et al. (2023) and compare the MSE loss and time-consuming ratios among EGNN, FastEGNN, and VN-EGNN. The results are presented in the table below. It should be noted that VN-EGNN is not an equivariant model, and it reports extremely higher prediction loss when rotations or translations are applied to the input graphs.

MSE

Dropping Rate	0.00	0.50	0.75	0.90	1.00
EGNN (Satorras et al., 2021)	2.25	2.25	2.24	2.25	2.26
FastEGNN-3	1.82	1.86	1.88	1.86	1.88
VN-EGNN-3 (Sestak et al., 2023) (test equivariance)	–	–	–	–	2703
VN-EGNN-3	1.84	1.88	1.90	1.97	1.92

Relative Time

Dropping Rate	0.00	0.50	0.75	0.90	1.00
EGNN (Satorras et al., 2021)	1.00	0.51	0.27	0.12	0.06
FastEGNN-3	1.39	1.08	0.96	0.92	0.80
VN-EGNN-3 (Sestak et al., 2023)	2.70	2.08	1.87	1.75	1.33