Supplementary Materials: Efficient equivariant model for machine learning interatomic potentials

Ziduo Yang[1,2,†], Xian Wang[3,†], Yifan Li[1], Qiujie Lv[1,2], Calvin Yu-Chian Chen[4,5,6*], and Lei Shen[1,7*]

[1]*Department of Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, 117575, Singapore*

[2]*Artificial Intelligence Medical Research Center, School of Intelligent Systems Engineering, Shenzhen Campus of Sun Yat-sen University, Shenzhen, 518107, China*

[3]*Department of Physics, National University of Singapore, 2 Science Drive 3, 117551, Singapore*

[4]*AI for Science (AI4S)-Preferred Program, School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School, Shenzhen, 518055, China*

[5]*State Key Laboratory of Chemical Oncogenomics, School of Chemical Biology and Biotechnology, Peking University Shenzhen Graduate School, Shenzhen, 518055, China*

[6]*Guangdong L-Med Biotechnology Co., Ltd, Meizhou, Guangdong, 514699, China and*

[7]*National University of Singapore (Chongqing) Research Institute, Chongqing, 401123, China*

(*Corresponding author(s): cy@pku.edu.cn; shenlei@nus.edu.sg)

(†These authors contributed equally to this work.)

1. PROOF OF INVARIANCE AND EQUIVARIANCE

First, it can be easily verified that the predicted scalar potentials of E^{2}GNN exhibit invariance to translations and rotations. Specifically, the initial scalar feature $\mathbf{x}_i^{(0)}$ is invariant, and the update process of $\mathbf{x}_i^{(t)}$ solely relies on invariant operations, such as vector norm and dot product. Therefore, the predicted potentials of E^{2}GNN are invariant. On the other hand, the predicted vector forces are multiple linear combinations of unit vectors $\frac{\vec{\mathbf{r}}_{ji}}{\|\vec{\mathbf{r}}_{ji}\|}$, which are equivalent. A formal proof is as follows.

Let $\vec{\mathbf{g}} \in \mathbb{R}^3$ be a translation vector and $\mathbf{Q} \in \mathbb{R}^{3\times 3}$ be a unitary rotation matrix. We prove that the vector representations are invariant to the translations and equivariant to rotations. Since both translation and rotation do not affect the values of $\|\cdot\|$ or dot product, we can view them as scalars (i.e., invariant features). We then consider $\vec{\mathbf{m}}_i$ as a function of $\vec{\mathbf{x}}_i$, $\vec{\mathbf{x}}_j$ and $\vec{\mathbf{r}}_{ji}$, that is $\vec{\mathbf{m}}_i = \phi(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j, \vec{\mathbf{r}}_{ji})$. Note that $\vec{\mathbf{r}}_{ji} = \vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j$.

Invariance. Note that $(\vec{\mathbf{r}}_i + \vec{\mathbf{g}}) - (\vec{\mathbf{r}}_j + \vec{\mathbf{g}}) = \vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j$. Therefore, the output $\vec{\mathbf{m}}_i$ will be invariant to translation as

$$\vec{\mathbf{m}}_i = \phi\big(\vec{\mathbf{x}}_i, \vec{\mathbf{x}}_j, (\vec{\mathbf{r}}_i + \vec{\mathbf{g}}) - (\vec{\mathbf{r}}_j + \vec{\mathbf{g}})\big) = \phi\big(\vec{\mathbf{r}}_{ij}, \vec{\mathbf{x}}_i, \vec{\mathbf{r}}_{ji}\big) \tag{1}$$

Since $\vec{\mathbf{x}}_i^{(t+1)}$ only depends on $\vec{\mathbf{m}}_i$ and scalars $\mathbf{m}_i \oplus \|\mathbf{V}\vec{\mathbf{m}}_i\|$, we can conclude that the vector representations are invariant to translations.

Equivariance. Suppose $t = 0$ and note that $\vec{\mathbf{x}}_i^{(0)} = \vec{\mathbf{0}} \in \mathbb{R}^{F\times 3}$, we then have $\vec{\mathbf{m}}_i = \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{W}_v \mathbf{x}_j^{(0)} \circ \vec{\mathbf{r}}_{ji}$ (we ignore $\|\vec{\mathbf{r}}_{ji}\|$ for simplicity). Note that Hadamard product (i.e., $\circ$) can be viewed as left multiplication by a diagonal matrix $\mathbf{D}$. Therefore, we can rewrite Eqn. (4) in the main text as

$$\vec{\mathbf{x}}_i^{(1)} = \mathbf{D}_h \mathbf{U} \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{D}_v (\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j) \tag{2}$$

where $\mathbf{D}_v = \mathbf{W}_v \mathbf{x}_j^{(t)}$ and $\mathbf{D}_h = \mathbf{W}_h(\mathbf{m}_i \oplus \|\mathbf{V}\vec{\mathbf{m}}_i\|)$. Here, we consider the coordinates as a set of row vectors. Therefore, rotating the coordinates $\mathcal{R}$ can be achieved by multiplication with $\mathbf{Q}$ from the right. Thus, $\vec{\mathbf{x}}_i^{(1)}$ will be equivalent to rotation as

$$\begin{aligned}\vec{\mathbf{x}}_i^{(1)} &= \mathbf{D}_h \mathbf{U} \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{D}_v (\vec{\mathbf{r}}_i \mathbf{Q} - \vec{\mathbf{r}}_j \mathbf{Q}) \\ &= [\mathbf{D}_h \mathbf{U} \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{D}_v (\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j)]\mathbf{Q} \end{aligned} \tag{3}$$

We can conclude $\vec{\mathbf{x}}_i^{(1)}$ is equivalent to rotation. Now suppose $\vec{\mathbf{x}}_i^{(t)}$ is already equivalent to rotation. Again, we can rewrite Eqn. (4) in the main text as

$$\vec{\mathbf{x}}_i^{(t+1)} = \mathbf{D}_h\mathbf{U}\sum_{v_j\in\mathcal{N}(v_i)}\mathbf{D}_u\vec{\mathbf{x}}_j^{(t)} + \mathbf{D}_v(\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j) \tag{4}$$

where $\mathbf{D}_u = \mathbf{W}_u\mathbf{x}_j^{(t)}$. Similarly, we have

$$\begin{aligned}\vec{\mathbf{x}}_i^{(t+1)} &= \mathbf{D}_h\mathbf{U}\sum_{v_j\in\mathcal{N}(v_i)}\mathbf{D}_u\vec{\mathbf{x}}_j^{(t)}\mathbf{Q} + \mathbf{D}_v(\vec{\mathbf{r}}_i\mathbf{Q} - \vec{\mathbf{r}}_j\mathbf{Q}) \\ &= [\mathbf{D}_h\mathbf{U}\sum_{v_j\in\mathcal{N}(v_i)}\mathbf{D}_u\vec{\mathbf{x}}_j^{(t)} + \mathbf{D}_v(\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j)]\mathbf{Q}\end{aligned} \tag{5}$$

Therefore, we can claim that the vector representations are equivalent to rotations.

2. PROOF OF THE EXTENSIVITY OF ENERGY

We prove that when the simulation box size doubles, the predicted energy exactly doubles. Note that this proof is generalizable to any box size, i.e., the predicted energy is proportional to the size of the system. Let $\mathbf{x}_i^{(t)[s]}$ and $\vec{\mathbf{x}}_i^{(t)[s]}$ denote the i-th node's scalar and vector representations in the single simulation box. Let $\mathbf{x}_i^{(t)[d1]}$ and $\vec{\mathbf{x}}_i^{(t)[d1]}$ denote the node's scalar and vector representations in the double simulation box, and let $\mathbf{x}_i^{(t)[d2]}$ and $\vec{\mathbf{x}}_i^{(t)[d2]}$ be their replicated counterparts due to the enlargement of the simulation box. In E^{2}GNN, all nodes share the same weight matrix in each layer, ensuring that a node's representation remains consistent as long as its local environment does not change. When we double the simulation box size by creating a supercell, the local environment of each node is replicated, preserving the original atomic distribution and local interactions. Based on this fact, the extensivity of energy is ensured by the following proof using mathematical induction:

- **Base Case ($t = 0$):** Initially, the scalar and vector representations for both the single and double simulation boxes are the same:

$$\mathbf{x}_i^{(0)[s]} = \mathbf{x}_i^{(0)[d1]} = \mathbf{x}_i^{(0)[d2]},\quad \vec{\mathbf{x}}_i^{(0)[s]} = \vec{\mathbf{x}}_i^{(0)[d1]} = \vec{\mathbf{x}}_i^{(0)[d2]},\quad \mathbf{x}_{\mathcal{G}}^{(0)[s]} = \mathbf{x}_{\mathcal{G}}^{(0)[d]},\quad \vec{\mathbf{x}}_{\mathcal{G}}^{(0)[s]} = \vec{\mathbf{x}}_{\mathcal{G}}^{(0)[d]} \tag{6}$$

- **Induction Hypothesis:** Assume that at time step $t = k$, the scalar and vector representations for both the single and double simulation boxes are the same:

$$\mathbf{x}_i^{(k)[s]} = \mathbf{x}_i^{(k)[d1]} = \mathbf{x}_i^{(k)[d2]}, \quad \vec{\mathbf{x}}_i^{(k)[s]} = \vec{\mathbf{x}}_i^{(k)[d1]} = \vec{\mathbf{x}}_i^{(k)[d2]}, \quad \mathbf{x}_{\mathcal{G}}^{(k)[s]} = \mathbf{x}_{\mathcal{G}}^{(k)[d]}, \quad \vec{\mathbf{x}}_{\mathcal{G}}^{(k)[s]} = \vec{\mathbf{x}}_{\mathcal{G}}^{(k)[d]} \tag{7}$$

- **Induction Step:** We need to show that the scalar and vector representations for both the single and double simulation boxes remain the same at time step $t = k + 1$.

  Local Message Passing: According to the local message passing phase, each node v_i gathers messages from its neighboring nodes. Given the same initial conditions and local environments, the intermediate scalars $\mathbf{m}_i$ and vectors $\vec{\mathbf{m}}_i$ will also be the same for both the single and double simulation boxes.

  Local Message Updating: The local message updating phase aggregates F scalars and vectors within $\mathbf{m}_i$ and $\vec{\mathbf{m}}_i$, respectively, to obtain new scalar $\mathbf{x}_i^{(k+1)}$ and new vector $\vec{\mathbf{x}}_i^{(k+1)}$. Since the intermediate messages are the same, the updated node representations will also be the same. That is,

$$\mathbf{x}_i^{(k+1)[s]} = \mathbf{x}_i^{(k+1)[d1]} = \mathbf{x}_i^{(k+1)[d2]} \tag{8}$$

$$\vec{\mathbf{x}}_i^{(k+1)[s]} = \vec{\mathbf{x}}_i^{(k+1)[d1]} = \vec{\mathbf{x}}_i^{(k+1)[d2]} \tag{9}$$

  Global Message Distributing and Aggregating: The global message distributing phase operates before the local message passing, distributing the global scalar and vector to each node. For the single simulation box:

$$\mathbf{x}_i^{(k)[s]} = \phi(\mathbf{x}_i^{(k-1)[s]} \oplus \mathbf{x}_{\mathcal{G}}^{(k-1)[s]}) + \mathbf{x}_i^{(k-1)[s]} \tag{10}$$

$$\vec{\mathbf{x}}_i^{(k)[s]} = \mathbf{W}(\vec{\mathbf{x}}_i^{(k-1)[s]} + \vec{\mathbf{x}}_{\mathcal{G}}^{(k-1)[s]}) + \vec{\mathbf{x}}_i^{(k-1)[s]} \tag{11}$$

  For the double simulation box:

$$\mathbf{x}_i^{(k)[d1]} = \mathbf{x}_i^{(k)[d2]} = \phi(\mathbf{x}_i^{(k-1)[d1]} \oplus \mathbf{x}_{\mathcal{G}}^{(k-1)[d]}) + \mathbf{x}_i^{(k-1)[d1]} \tag{12}$$

$$\vec{\mathbf{x}}_i^{(k)[d1]} = \vec{\mathbf{x}}_i^{(k)[d2]} = \mathbf{W}(\vec{\mathbf{x}}_i^{(k-1)[d1]} + \vec{\mathbf{x}}_{\mathcal{G}}^{(k-1)[d]}) + \vec{\mathbf{x}}_i^{(k-1)[d1]} \tag{13}$$

  Given that both the global and local node representations are the same, the updated local scalars and vectors will also be the same. That is,

$$\mathbf{x}_i^{(k)[s]} = \mathbf{x}_i^{(k)[d1]} = \mathbf{x}_i^{(k)[d2]} \tag{14}$$

$$\vec{\mathbf{x}}_i^{(k)[s]} = \vec{\mathbf{x}}_i^{(k)[d1]} = \vec{\mathbf{x}}_i^{(k)[d2]} \tag{15}$$

After local message updating, the global scalar and vector are updated using the node representations:

$$\mathbf{x}_{\mathcal{G}}^{(k+1)[s]} = \phi\left(\left(\frac{1}{|\mathcal{G}_s|}\sum_{v_i \in \mathcal{G}_s} \mathbf{x}_i^{(k)}\right) \oplus \mathbf{x}_{\mathcal{G}}^{(k)[s]}\right) + \mathbf{x}_{\mathcal{G}}^{(k)[s]} \tag{16}$$

$$\vec{\mathbf{x}}_{\mathcal{G}}^{(k+1)[s]} = \mathbf{W}\left(\left(\frac{1}{|\mathcal{G}_s|}\sum_{v_i \in \mathcal{G}_s} \vec{\mathbf{x}}_i^{(k)}\right) + \vec{\mathbf{x}}_{\mathcal{G}}^{(k)[s]}\right) + \vec{\mathbf{x}}_{\mathcal{G}}^{(k)[s]} \tag{17}$$

For the double simulation box:

$$\mathbf{x}_{\mathcal{G}}^{(k+1)[d]} = \phi\left(\left(\frac{1}{|\mathcal{G}_d|}\sum_{v_i \in \mathcal{G}_d} \mathbf{x}_i^{(k)}\right) \oplus \mathbf{x}_{\mathcal{G}}^{(k)[d]}\right) + \mathbf{x}_{\mathcal{G}}^{(k)[d]} \tag{18}$$

$$\vec{\mathbf{x}}_{\mathcal{G}}^{(k+1)[d]} = \mathbf{W}\left(\left(\frac{1}{|\mathcal{G}_d|}\sum_{v_i \in \mathcal{G}_d} \vec{\mathbf{x}}_i^{(k)}\right) + \vec{\mathbf{x}}_{\mathcal{G}}^{(k)[d]}\right) + \vec{\mathbf{x}}_{\mathcal{G}}^{(k)[d]} \tag{19}$$

Here, $\mathcal{G}_s$ and $\mathcal{G}_d$ represent the graphs derived from the single and double simulation box systems, respectively, with $|\mathcal{G}_d| = 2|\mathcal{G}_s|$. Note that:

$$\frac{1}{|\mathcal{G}_s|}\sum_{v_i \in \mathcal{G}_s} \mathbf{x}_i^{(k)[s]} = \frac{1}{|\mathcal{G}_d|}\sum_{v_i \in \mathcal{G}_d} \mathbf{x}_i^{(k)[d]} \tag{20}$$

$$\frac{1}{|\mathcal{G}_s|}\sum_{v_i \in \mathcal{G}_s} \vec{\mathbf{x}}_i^{(k)[s]} = \frac{1}{|\mathcal{G}_d|}\sum_{v_i \in \mathcal{G}_d} \vec{\mathbf{x}}_i^{(k)[d]} \tag{21}$$

Therefore, we have:

$$\mathbf{x}_{\mathcal{G}}^{(k+1)[s]} = \mathbf{x}_{\mathcal{G}}^{(k+1)[d]} \tag{22}$$

$$\vec{\mathbf{x}}_{\mathcal{G}}^{(k+1)[s]} = \vec{\mathbf{x}}_{\mathcal{G}}^{(k+1)[d]} \tag{23}$$

By induction, we have shown that the scalar representations $\mathbf{x}_i^{(t)[s]}$, $\mathbf{x}_i^{(t)[d1]}$, $\mathbf{x}_i^{(t)[d2]}$ and vector representations $\vec{\mathbf{x}}_i^{(t)[s]}$, $\vec{\mathbf{x}}_i^{(t)[d1]}$, $\vec{\mathbf{x}}_i^{(t)[d2]}$ remain consistent at each time step t. Therefore, at the final time step T, these representations will still be identical for both the single and double simulation boxes. Therefore, the total energy in the double simulation box is exactly twice that in the single simulation box, ensuring the extensivity of energy:

$$e^{[d]} = \sum_{v_i \in \mathcal{G}_d} e_i^{[s]} = 2\sum_{v_i \in \mathcal{G}_s} e_i^{[s]} = 2e^{[s]} \tag{24}$$

3. HYPERPARAMETER CONFIGURATION

The hyperparameter configuration for all deep learning models is presented in **Table S1**. These settings are primarily based on references from the Open Catalyst Project (OCP) (`https://github.com/Open-Catalyst-Project/ocp`), MDsim (`https://github.com/kyonofx/MDsim`), and MACE (`https://github.com/ACEsuit/mace`) with default settings. To ensure a fair comparison, all models use the same input graphs, with a cutoff distance D set to 6 and a maximum number of neighbors set to 20 for the OC20 dataset to reduce computational cost. For the MD17 dataset, D is set to 5 and N to 50. The maximum epochs are set to 2000, with a ReduceLROnPlateau scheduler for reducing the learning rate when performance does not improve within five epochs. Early stopping is employed when there is no improvement for 50 epochs. Note that we do not tune the hyperparameters for MACE except for the learning rate; instead, we use the typical settings recommended by the project (`https://github.com/ACEsuit/mace`).

It's important to note that the Vanilla model differs from PaiNN_Direct in several ways, resulting in the performance differences observed. First, in the message passing phase, the Vanilla model uses $\vec{\mathbf{r}}_{ij}$ instead of $\vec{\mathbf{r}}_{ji}$ as in PaiNN, which provides clearer interpretability. Specifically, the term $\sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{W}_v \mathbf{x}_j^{(t)}) \circ \lambda_v(\|\vec{\mathbf{r}}_{ji}\|) \circ \frac{\vec{\mathbf{r}}_{ji}}{\|\vec{\mathbf{r}}_{ji}\|}$ now directly represents the total force exerted on atom i. Second, the OCP implementation of PaiNN uses two gated equivariant blocks to transform vector representations into force predictions, while E^{2}GNN simplifies this by employing a linear mapping, reducing the overall model complexity. Third, in the OCP implementation, PaiNN defaults to an "on-the-fly" approach for converting crystal or molecular structures into graph representations, whereas E^{2}GNN uses a "static" method. These different graph-building strategies also impact training dynamics and model performance; further details can be found at `https://github.com/Open-Catalyst-Project/ocp` and `https://github.com/Shen-Group/E2GNN`.

TABLE S1: The hyperparameter configuration for all models

	CGCNN
atom_embedding_size	128
fc_feat_size	128
num_fc_layers:	3

num_graph_conv_layers	2
num_gaussians	100
α	1
β	30
learning rate	0.0001
SchNet	
hidden_channels	1024
num_filters	256
num_interactions	3
num_gaussians	200
α	1
β	50
learning rate	0.0001
MACE	
r_max	6
num_bessel	8
num_polynomial_cutoff	6
max_ell	3
interaction_cls	RealAgnosticResidualInteractionBlock
interaction_cls_first	RealAgnosticResidualInteractionBlock
num_interactions	2
hidden_irreps	o3.Irreps('128x0e + 128x1o')
MLP_irreps	o3.Irreps('16x0e')
gate	silu
correlation	3
radial_type	bessel
α	1
β	30
learning rate	0.01
DimeNet++	

hidden_channels	192
out_emb_channels	192
num_blocks	3
num_radial	6
num_spherical	7
num_before_skip	1
num_after_skip	2
num_output_layers	3
GemNet-dT	
num_spherical	7
num_radial	128
num_blocks	3
emb_size_atom	512
emb_size_edge	512
emb_size_trip	64
emb_size_rbf	16
emb_size_cbf	16
emb_size_bil_trip	64
num_before_skip	1
num_after_skip	2
num_concat	1
num_atom:	3
rbf	gaussian
envelope	polynomial
envelope (exponent)	5
cbf	spherical_harmonics
extensive	True
output_init	HeOrthogonal
activation	silu
α	1

β	30
learning rate	0.0005
PaiNN	
hidden_channels	512
num_layers	4
num_rbf	128
α	1
β	50
learning rate	0.0001
E^{2}GNN	
hidden_channels	512
num_layers	4
num_rbf	128
α	1
β	50
learning rate	0.0002

4. MODEL PERFORMANCE ON OC20-50K DATASET

For the OC20-50K ($N = 50,000$) dataset, as shown in **Table S2**, E^{2}GNN consistently surpasses benchmark models in energy and force predictions across all external validation sets. The reduced performance of GemNet-dT on OC-50K may be attributed to its large number of parameters, which can lead to overfitting on a relatively small dataset.

5. MOLECULAR DYNAMICS SIMULATIONS

A. Experimental settings

We incorporate E^{2}GNN-based forces with the Atomic Simulation Environment (ASE) [1] to perform MD simulations. The Lennard-Jones potential is also implemented using ASE with the default settings: sigma = 1.0, epsilon = 1.0, r_c = None, r_o = None, and smooth =

TABLE S2. Comparison results of the proposed E^{2}GNN and baselines on S2EF task of four external validation sets of OC20 in terms of energy MAE (meV) and forces MAE (meV/Å), where all models are trained on the same OC20-50K.

Model	ID		OOD Ads.		OOD Cat.		OOD Both	
	Energy	Forces	Energy	Forces	Energy	Forces	Energy	Forces
CGCNN	1125	75.3	1255	79.9	1111	74.4	1386	91.5
SchNet	1138	67.9	1255	73.3	1121	67.4	1394	84.9
MACE	752	59.2	867	68.1	752	58.9	1013	77.2
PaiNN	603	64.8	772	69.8	605	64.2	895	81.2
PaiNN_Direct	593	57.3	721	61.7	613	56.3	861	73.1
DimeNet++	647	59.2	752	68.2	646	58.7	875	78.5
GemNet-dT	641	58.7	784	63.8	733	58.1	1029	75.4
E^{2}GNN (ours)	**531**	**51.2**	**626**	**56.2**	**555**	**50.7**	**750**	**66.0**

False.

B. Datasets

1. *Solid LiPS*

In this simulation, we use $Li_{6.75}P_3S_{11}$, a crystalline superionic lithium conductor widely used in battery development. The simulation cell comprises 83 atoms and serves as a representative system for studying kinetic properties in materials through MD simulations. We utilize the dataset from Batzner et al.'s work [2]. This AIMD trajectory consists of a total of 25,000 structures, with 19,000 randomly sampled structures used for training, 1,000 for validation, and the remainder for testing. The simulation employs a time step of 0.25 fs and a temperature of 520 K, with a Nosé–Hoover thermostat, which aligns with those used in MDSim[3].

2. Liquid H_2O

Water, essential in biological and chemical processes, presents complexities in thermodynamics and phase behavior that make it challenging to simulate. For this system, our dataset comprises 100,000 structures, each containing 92 atoms. Of these, 72,000 structures are randomly sampled for training, 8,000 for validation, and the remaining 20,000 for testing. The simulation is set with a time step of 0.5 fs, a temperature of 300 K, and uses a Langevin thermostat. The duration of the simulation is 50 ps.

3. Gas CH_4

Methane is a potent greenhouse gas with significant implications for climate change and serves as the primary component of natural gas, a vital global energy source. Its molecular dynamics properties are crucial for understanding environmental impacts, energy applications, and broader scientific phenomena. For methane, our dataset is made up of 100,000 structures, each containing 100 atoms. From this, 72,000 structures are randomly sampled for training, 8,000 for validation, and the remaining 20,000 for testing. The simulation adopts a time step of 0.5 fs, a temperature of 300 K, and a Langevin thermostat. The simulation's duration is set at 50 ps.

6. PERFORMANCE INDICATORS

Mean absolute error (MAE). The performance of energy and force predictions is quantified using MAE, calculated as follows:

$$\mathrm{MAE}(e) = \frac{1}{N}\sum_{n=1}^{N} |e_n - e_n^l| \tag{25}$$

$$\mathrm{MAE}(\vec{\mathbf{F}}) = \frac{1}{3NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{k=1}^{3} |\vec{\mathbf{F}}_{nmk} - \vec{\mathbf{F}}^l_{nmk}| \tag{26}$$

Distribution of interatomic distances. The distribution of interatomic distances ($h(r)$) offers a concise representation of a system's 3D structure and has been explored in prior research [3]. For a specific configuration denoted as x, $h(r)$ is determined using the equation:

$$h(r) = \frac{1}{N(N-1)} \sum_{i}^{N} \sum_{j \neq i}^{N} \delta(r - \|x_i - x_j\|) \tag{27}$$

In this equation, r represents the distance from a reference particle, while N denotes the total particle count. The indices i and j identify the atom pairs contributing to the distance statistics. The function δ is the Dirac Delta function, employed to extract value distributions. To determine the ensemble average, $h(r)$ is computed and then averaged across frames.

Radial distribution function (RDF). RDF, a valuable simulation observable, characterizes the structural and thermodynamic properties of the system. By its definition, the RDF elucidates the variation of density as a function of distance from a reference particle, as shown in **Fig. S1(a)**. The RDF for a particular configuration x is computed as follows:

$$\mathrm{RDF}(r) = \frac{1}{4\pi r^2} \frac{1}{N\rho} \sum_{i=1}^{N} \sum_{j \neq i}^{N} \delta(r - \|x_i - x_j\|) \tag{28}$$

In this equation, r represents the distance from a reference particle, N is the total number of particles, and i, j are indices referring to the atom pairs contributing to distance statistics. Additionally, ρ denotes the system's density, and δ is the Dirac Delta function, applied for value distribution extraction. The ensemble average is calculated by averaging $\mathrm{RDF}(r)$ over frames. The final RDF MAE is obtained by integrating over r:

$$\mathrm{MAE}(\mathrm{RDF}) = \int_{r=0}^{\infty} |\langle \mathrm{RDF}(r) \rangle - \langle \hat{\mathrm{RDF}}(r) \rangle| dr \tag{29}$$

where $\langle \cdot \rangle$ indicates the averaging operator, $\langle \mathrm{RDF}(r) \rangle$ is the reference equilibrium RDF, and $\langle \hat{\mathrm{RDF}}(r) \rangle$ is the RDF predicted by the model.

Stability criterion. In the study, stability is defined as the ability to retain low-energy configurations [3]. For systems with periodic boundary conditions, we monitor stability by tracking equilibrium statistics. We denote a simulation as 'unstable' at time T if

$$\int_{r=0}^{\infty} \|\langle \mathrm{RDF}(r) \rangle - \langle \hat{\mathrm{RDF}}(r) \rangle_{t=T}^{T+\tau} \| dr > \Delta \tag{30}$$

where τ refers to a short time window and Δ represents the stability threshold. We set $\Delta = 1.0$, which is the same as in MDsim [3]. The stability of a model is subsequently assessed based on how long it maintains stability during the simulation.

Diffusivity. The diffusivity coefficient D quantifies the time-correlation of the translational displacement, as illustrated in **Fig. S1(b)**, and can be computed from the mean square displacement:

$$D = \lim_{t\to\infty} \frac{1}{6t}\frac{1}{N'}\sum_{i=1}^{N'}\left[x_i(t) - x_i(0)\right]^2 \tag{31}$$

where $x_i(t)$ is the coordinate of particle i at time t, and N' is the number of particles being tracked in the system. For the LiPS system, we monitor Li-ion Diffusivity and track all 27 Li-ions, which is the same as that used in MDSim [3]. As the definition implies, D is a quantity that converges with a longer simulation time. Accurate recovery of D requires sufficiently long trajectories sampled from the Hamiltonian with MLFFs. In this paper, we only compute diffusivity for stable trajectories of at least 40 ps for LiPS, which is the same as that used in MDSim [3].

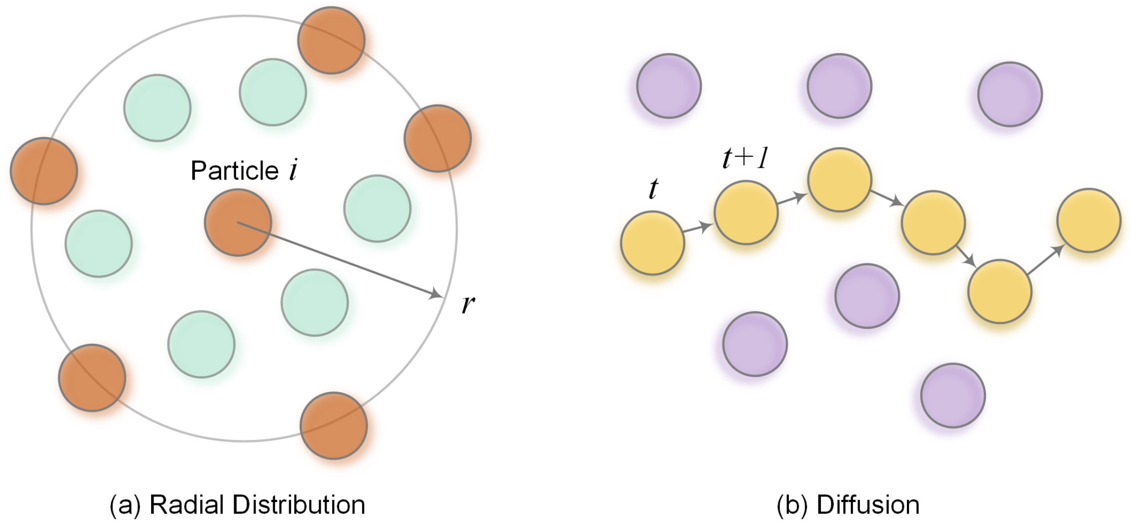

FIG. S1. Illustrations of the benchmarked metrics: (a) Radial distribution and (b) Diffusion.

7. MODEL PERFORMANCE ON MD17 DATASET

The MD17 dataset comprises eight molecular dynamics simulations featuring small organic molecules [4]. Each of these simulations represents a unique trajectory for a single molecule, covering a broad range of conformations. The purpose of this task is to predict energies and forces corresponding to each of these trajectories. We opt for a training set of 19,000 entries, a validation set of 1,000 entries, and a test set of 10,000 entries. It's worth emphasizing that the same sets are utilized across all model training, validation, and testing phases. **Table S3** presents the comparative results of the models. E^{2}GNN demonstrates superior performance to DimeNet++ across 15 targets, and outperforms GemNet-dT on 9 out of the 16 targets. This indicates that models based on two-body interactions, such as E^{2}GNN, are able to compete with models based on many-body interactions, such as DimeNet++ and GemNet-dT. These findings attest to the efficacy of E^{2}GNN, even when compared with more complex models based on many-body interactions.

TABLE S3. Mean absolute errors on MD17 dataset for energy and force predictions in terms of energy MAE (meV) and forces MAE (meV/Å), respectively. In each cell, the errors are written in energy MAE and forces MAE pairs.

	CGCNN	SchNet	PaiNN	DimeNet++	GemNet-dT	E^{2}GNN
Aspirin	144.7, 610.4	12.1, 21.3	19.7, 3.4	17.8, 5.5	9.9, 3.8	**8.5**, **2.9**
Benzene	7.4, 67.8	3.9, 12	80.0, **5.9**	10, 8.3	**2.8**, 6.5	14.7, 6.0
Ethanol	24.1, 200.5	2.2, 4.6	9.0, 2.4	9.7, 2.0	**2.2**, **1.3**	2.7, 1.6
Malonaldehyde	44.5, 338.4	3.7, 8.2	9.3, 3.1	11.4, 3.0	**3.9**, **2.4**	4.4, 2.5
Naphthalene	34.9, 170.5	9.9, 11.5	8.1, 1.3	10.5, 2.5	**2.8**, 1.3	3.1, **0.8**
Salicylic acid	40.7, 198.4	8.1, 14.9	11.6, 2.2	16.9, 5.3	5.8, 3.2	**5.0**, **2.0**
Toluene	26.4, 153.8	6.0, 9.9	7.2, 1.4	9.7, 2.2	**2.8**, 1.3	3.1, **1.1**
Uracil	28.2, 183.6	5.3, 10.6	6.5, 1.4	11.3, 2.4	3.2, 1.7	**3.0**, **1.0**

8. MODEL PERFORMANCE ON ISO17 DATASET

The ISO17 dataset constitutes a collection of MD trajectories simulated for 129 distinct organic isomers [5]. Despite all isomers having the same C7O2H10 composition, they each have a unique structure. Each isomer's trajectory consists of 5,000 snapshots, yielding a total of 645,000 unique snapshots. These snapshots are divided into three non-overlapping sets according to the original work [5]: a training set and two testing sets. The training set contains 80% of the randomly selected snapshots from the MD trajectories of 80% of the 129 isomers. The first test set, termed as the 'test within' set, contains the remaining 20% of snapshots drawn from the same molecules used in the training set. The second, more challenging, test set comprises all snapshots from the other 20% of the 129 molecules, not included in the training set, and is referred to as the 'test other' set. The first test set aims to gauge the capability of GNNs to interpolate the forces of unfamiliar geometries for known molecules, i.e., molecules present in the training data. In contrast, the second test set evaluates E^{2}GNN's proficiency at generalizing to unfamiliar molecules, i.e., molecules never encountered during training. A validation set, including 4K examples, is utilized to select the optimal model for testing. We compare E^{2}GNN with the originally reported results of SchNet [5] and GNNFF [6]. Note that the training, validation, and test sets are the same for the three methods. The performance is measured based on energy MAE and forces MAE.

Table S4 summarizes the test results. In the 'test within' scenario, E^{2}GNN achieves an energy MAE of 0.004, representing a substantial improvement of 75% over SchNet's performance. Similarly, the forces MAE of E^{2}GNN is 0.003, marking an impressive improvement of 91.67% over the previous best model, GNNFF. In the more challenging 'test other' scenario, E^{2}GNN achieves an energy MAE of 0.050, reflecting an enhancement of 51.92% compared to SchNet's performance. Moreover, E^{2}GNN's forces MAE is 0.029, which is 67.05% superior to the performance of GNNFF. These results demonstrate that E^{2}GNN significantly boosts performance in both in-domain and out-of-domain scenarios compared to the previous best models.

TABLE S4. Comparison among SchNet, GNNFF, and E^{2}GNN in terms of energy MAE (eV) and forces MAE (eV/Å) on ISO17 database. In each cell, the errors are written in energy MAE and forces MAE pairs. "-" indicates lack of data.

	SchNet	GNNFF	E^{2}GNN
Test within	0.016, 0.043	-, 0.036	**0.004**, **0.003**
Test other	0.104, 0.095	-, 0.088	**0.050**, **0.029**

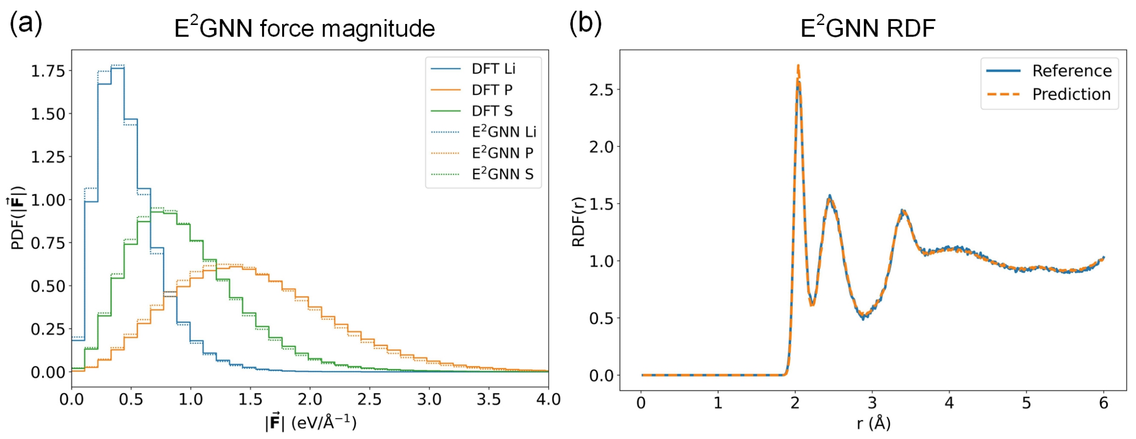

FIG. S2. Testing E^2GNN on a larger system consisting of 2241 atoms. (a) Probability density function (PDF) of the force magnitude $|\vec{\mathbf{F}}|$ for different atom types (Li, P, and S) from the DFT and E^2GNN simulations. (b) RDF comparison between the DFT reference and E^2GNN predictions for the same system.

9. IMPACT OF THE GLOBAL NODE ON MD SIMULATIONS

The introduction of a global node in machine learning models may, in certain scenarios, lead to inaccuracies when distant parts of a system are improperly influenced by global changes. To further investigate this, we tested E^2GNN on a larger system consisting of 2241 atoms. This system was generated by constructing a $3 \times 3 \times 3$ supercell based on the LiPS structure. To evaluate the performance of E^2GNN in this larger system, we compared the force magnitude distributions and RDF from the resulting ML trajectory to those from DFT simulations without using a supercell. As shown in **Fig. S2**, the force magnitude distributions and RDF are consistent between the supercell trajectory predicted by our ML model and the reference DFT trajectory without the supercell. This similarity implies that long-range communication between atoms, introduced by the global node, does not significantly alter the system's physical behavior, particularly in terms of pairwise distances. Table S5 presents the MAEs in energy and forces for both the large ($3 \times 3 \times 3$ supercell) and the small (unit cell) LiPS system. The results show that the energy and force MAEs remain almost unchanged by the introduction of the global node, further demonstrating that its presence has a minimal impact on the overall accuracy of MD simulations. Additional investigations into the role and influence of the global node will be explored in future work.

TABLE S5. The MAEs in energy (meV) and forces (meV/Å) for both the large ($3 \times 3 \times 3$ supercell) and the small (unit cell) LiPS system

System	Energy (↓)	Forces (↓)
Large LiPS	3.844	1.531
Small LiPS	3.845	1.531

[1] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, *et al.*, The atomic simulation environment—a python library for working with atoms, Journal of Physics: Condensed Matter **29**, 273002 (2017).

[2] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky, E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials, Nature communications **13**, 2453 (2022).

[3] X. Fu, Z. Wu, W. Wang, T. Xie, S. Keten, R. Gomez-Bombarelli, and T. S. Jaakkola, Forces are not enough: Benchmark and critical evaluation for machine learning force fields with molecular simulations, Transactions on Machine Learning Research (2023).

[4] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, Machine learning of accurate energy-conserving molecular force fields, Science advances **3**, e1603015 (2017).

[5] K. Schütt, P.-J. Kindermans, H. E. Sauceda Felix, S. Chmiela, A. Tkatchenko, and K.-R. Müller, Schnet: A continuous-filter convolutional neural network for modeling quantum interactions, Advances in neural information processing systems **30** (2017).

[6] C. W. Park, M. Kornbluth, J. Vandermause, C. Wolverton, B. Kozinsky, and J. P. Mailoa, Accurate and scalable graph neural network force field and molecular dynamics with direct force architecture, npj Computational Materials **7**, 73 (2021).