Appendix

Our implementation is based on PyTorch [53], PyG [20], `e3nn` [28] and `timm` [82]. We include code for experiments on QM9 in appendix and will release code reproducing all main results in the future.

Additionally, we update the results of IS2RE with IS2RS auxiliary task by using Noisy Nodes [30] data augmentation and summarize them in Table 7 and 8. As of May 20, 2022, Equiformer achieves the best results on IS2RE task when only IS2RE and IS2RS data are used.

	Energy MAE (eV) ↓					EwT (%) ↑				
Methods	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
GNS [30]	0.54	0.65	0.55	0.59	0.5825	-	-	-	-	-
Noisy Nodes [30]	0.47	0.51	0.48	0.46	0.4800	-	-	-	-	-
Graphormer [66]	0.4329	0.5850	0.4441	0.5299	0.4980	-	-	-	-	-
Equiformer	0.4222	0.5420	0.4231	0.4754	0.4657	7.23	3.77	7.13	4.10	5.56
+ Noisy Nodes	0.4156	0.4976	0.4165	0.4344	0.4410	7.47	4.64	7.19	4.84	6.04

Table 7: **Results on OC20 IS2RE validation set when IS2RS node-level auxiliary task is adopted during training.** "GNS" denotes the 50-layer GNS trained without Noisy Nodes data augmentation, and "Noisy Nodes" denotes the 100-layer GNS trained with Noisy Nodes. Compared to the main text, we add the result of "Equiformer + Noisy Nodes", which use data augmentation of interpolating between initial structure and relaxed struture and adding Gaussian noise as described by Noisy Nodes [30].

	Energy MAE (eV) ↓					EwT (%) ↑				
Methods	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
GNS + Noisy Nodes [30]	0.4219	0.5678	0.4366	0.4651	0.4728	9.12	4.25	8.01	4.64	6.5
Graphormer [66]†	0.3976	0.5719	0.4166	0.5029	0.4722	8.97	3.45	8.18	3.79	6.1
Equiformer + Noisy Nodes	0.4171	0.5479	0.4248	0.4741	0.4660	7.71	3.70	7.15	4.07	5.66

Table 8: **Results on OC20 IS2RE testing set when IS2RS node-level auxiliary task is adopted during training.** † denotes using ensemble of models trained with both IS2RE training and validation sets. In contrast, we use the same single Equiformer model in Table 7, which is trained with only the training set, for evaluation on the testing set.

A Additional Mathematical Background

In this section, we provide additional mathematical background on group equivariance helpful for the discussion of the proposed method. Other works [73, 81, 44, 1, 23, 5] also provide similar background. We encourage interested readers to see these works [87, 17] for more in-depth and pedagogical presentations.

A.1 Group Theory

Definition of Groups. A group is an algebraic structure that consists of a set G and a binary operator $\circ : G \times G \rightarrow G$ and is typically denoted as G. Groups satisfy the following four axioms:

1. Closure: $g \circ h \in G$ for all $g, h \in G$.
2. Identity: There exists an identity element $e \in G$ such that $g \circ e = e \circ g = g$ for all $g \in G$.
3. Inverse: For each $g \in G$, there exists an inverse element $g^{-1} \in G$ such that $g \circ g^{-1} = g^{-1} \circ g = e$.
4. Associativity: $f \circ g \circ h = (f \circ g) \circ h = f \circ (g \circ h)$ for all $f, g, h \in G$.

In this work, we focus on 3D rotation, translation and inversion. Relevant groups include:

1. The Euclidean group in three dimensions $E(3)$: 3D rotation, translation and inversion.
2. The special Euclidean group in three dimensions $SE(3)$: 3D rotation and translation.

3. The orthogonal group in three dimensions $O(3)$: 3D rotation and inversion.
4. The special orthogonal group in three dimensions $SO(3)$: 3D rotation.

Group Representations. The actions of groups define transformations. Formally, a transformation acting on vector space X parametrized by group element $g \in G$ is an injective function $T_g : X \to X$. A powerful result of group representation theory is that these transformations can be expressed as matrices which act on vector spaces via matrix multiplication. These matrices are called the group representations. Formally, a group representation $D : G \to GL(N)$ is a mapping between a group G and a set of $N \times N$ invertible matrices. The group representation $D(g) : X \to X$ maps an N-dimensional vector space X onto itself and satisfies $D(g)D(h) = D(g \circ h)$ for all $g, h \in G$.

How a group is represented depends on the vector space it acts on. If there exists a change of basis P in the form of an $N \times N$ matrix such that $P^{-1}D(g)P = D'(g)$ for all $g \in G$, then we say the two group representations are equivalent. If $D'(g)$ is block diagonal, which means that g acts on independent subspaces of the vector space, the representation $D(g)$ is reducible. A particular class of representations that are convenient for composable functions are irreducible representations or "irreps", which cannot be further reduced. We can express any group representation of $SO(3)$ as a direct sum (concatentation) of irreps [87, 17, 28]:

$$D(g) = P^{-1}\left(\bigoplus_i D_{l_i}(g)\right)P = P^{-1}\begin{pmatrix} D_{l_0}(g) & & \\ & D_{l_1}(g) & \\ & & \cdots\cdots \end{pmatrix}P \tag{9}$$

where $D_{l_i}(g)$ are Wigner-D matrices with degree l_i as metnioned in Sec. 2.3.

A.2 Equivariance

Definition of Equivariance and Invariance. Equivariance is a property of a function $f : X \to Y$ mapping between vector spaces X and Y. Given a group G and group representations $D_X(g)$ and $D_Y(g)$ in input and output spaces X and Y, f is equivariant to G if $D_Y(g)f(x) = f(D_X(g)x)$ for all $x \in X$ and $g \in G$. Invariance corresponds to the case where $D_Y(g)$ is the identity I for all $g \in G$.

Equivariance in Neural Networks. Group equivariant neural networks are guaranteed to to make equivariant predictions on data transformed by a group. Additionally, they are found to be data-efficient and generalize better than non-symmetry-aware and invariant methods [4, 55, 22]. For 3D atomistic graphs, we consider equivariance to the Euclidean group $E(3)$, which consists of 3D rotation, translation and inversion. For translation, we operate on relative positions and therefore our networks are invariant to 3D translation. We achieve equivariance to rotation and inversion by representing our input data, intermediate features and outputs in vector spaces of $O(3)$ irreps and acting on them with only equivariant operations.

A.3 Equivariant Features Based on Vector Spaces of Irreducible Representations

Irreps Features. As discussed in Sec. 2.3 in the main text, we use type-L vectors for $SE(3)$-equivariant irreps features[1] and type-(L, p) vectors for $E(3)$-equivariant irreps features. Parity p denotes whether vectors change sign under inversion and can be either e (even) or o (odd). Vectors with $p = o$ change sign under inversion while those with $p = e$ do not. Scalar features correspond to type-0 vectors in the case of $SE(3)$-equivariance and correspond to type-$(0, e)$ in the case of $E(3)$-equivariance whereas type-$(0, o)$ vectors correspond to pseudo-scalars. Euclidean vectors in $\mathbb{R}^3$ correspond to type-1 vectors and type-$(1, o)$ vectors whereas type-$(1, e)$ vectors correspond to pseudo-vectors. Note that type-(L, e) vectors and type-(L, o) vectors are considered vectors of different types in equivariant linear layers and layer normalizations.

Spherical Harmonics. Euclidean vectors $\vec{r}$ in $\mathbb{R}^3$ can be projected into type-L vectors $f^{(L)}$ by using spherical harmonics $Y^{(L)}$: $f^{(L)} = Y^{(L)}(\frac{\vec{r}}{||\vec{r}||})$ [69]. This is equivalent to the Fourier transform of the angular degree of freedom $\frac{\vec{r}}{||\vec{r}||}$, which can be optionally weighted by $||\vec{r}||$. In the case of

[1] In SEGNN [5], they are also referred to as steerable features. We use the term "irreps features" to remain consistent with `e3nn` [28] library.

$SE(3)$-equivariance, $f^{(L)}$ transforms in the same manner as type-L vectors. For $E(3)$-equivariance, $f^{(L)}$ behaves as type-(L, p) vectors, where $p = e$ if L is even and $p = o$ if L is odd.

Vectors of Higher L and Other Parities. Although previously we have restricted concrete examples of vector spaces of $O(3)$ irreps to commonly encountered scalars (type-$(0, e)$ vectors) and Euclidean vectors (type-$(1, o)$ vectors), vector of higher L and other parities are equally physical. For example, the moment of inertia (how an object rotates under torque) transforms as a 3×3 symmetric matrix, which has symmetric-traceless components behaving as type-$(2, e)$ vectors. Elasticity (how an object deforms under loading) transforms as a rank-4 or $3 \times 3 \times 3 \times 3$ symmetric tensor, which includes components acting as type-$(4, e)$ vectors.

A.4 Tensor Product

Tensor Product for $O(3)$. We use tensor products to interact different type-(L, p) vectors. We extend our discussion in Sec. 2.4 in the main text to include inversion and type-(L, p) vectors. The tensor product denoted as $\otimes$ uses Clebsch-Gordan coefficients to combine type-(L_1, p_1) vector $f^{(L_1,p_1)}$ and type-L_2 vector $g^{(L_2,p_2)}$ and produces type-(L_3, p_3) vector $h^{(L_3,p_3)}$ as follows:

$$h_{m_3}^{(L_3,p_3)} = (f^{(L_1,p_1)} \otimes g^{(L_2,p_2)})_{m_3} = \sum_{m_1=-L_1}^{L_1} \sum_{m_2=-L_2}^{L_2} C_{(L_1,m_1)(L_2,m_2)}^{(L_3,m_3)} f_{m_1}^{(L_1,p_1)} g_{m_2}^{(L_2,p_2)} \tag{10}$$

$$p_3 = p_1 \times p_2 \tag{11}$$

The only difference of tensor products for $O(3)$ as described in Eq. 10 from those for $SO(3)$ described in Eq. 2 is that we additionally keep track of the output parity p_3 as in Eq. 11 and use the following multiplication rules: $e \times e = e$, $o \times o = e$, and $e \times o = o \times e = o$. For example, the tensor product of a type-$(1, o)$ vector and a type-$(1, e)$ vector can result in one type-$(0, o)$ vector, one type-$(1, o)$ vector, and one type-$(2, o)$ vector.

Clebsch-Gordan Coefficients. The Clebsch-Gordan coefficients for $SO(3)$ are computed from integrals over the basis functions of a given irreducible representation, e.g., the real spherical harmonics, as shown below and are tabulated to avoid unnecessary computation.

$$C_{(L_1,m_1)(L_2,m_2)}^{(L_3,m_3)} = |L_1 m_1; L_2 m_2\rangle \langle L_3 m_3| = \int d\Omega Y_{m_1}^{(L_1)*}(\Omega) Y_{m_2}^{(L_2)*}(\Omega) Y_{m_3}^{(L_3)}(\Omega) \tag{12}$$

For many combinations of L_1, L_2, and L_3, the Clebsch-Gordan coefficients are zero. The gives rise to the following selection rule for non-trivial coefficients: $-|L_1 + L_2| \leq L_3 \leq |L_1 + L_2|$.

Examples of Tensor Products. Tensor products generally define the interaction between different type-(L, p) vectors in a symmetry-preserving manner and consist of common operations as follows:

1. Scalar-scalar multiplication: scalar $(L = 0, p = e) \otimes$ scalar $(L = 0, p = e) \to$ scalar $(L = 0, p = e)$.
2. Scalar-vector multiplication: scalar $(L = 0, p = e) \otimes$ vector $(L = 1, p = o) \to$ vector $(L = 1, p = o)$.
3. Vector dot product: vector $(L = 1, p = o) \otimes$ vector $(L = 1, p = o) \to$ scalar $(L = 0, p = e)$.
4. Vector cross product: vector $(L = 1, p = o) \otimes$ vector $(L = 1, p = o) \to$ pseudo-vector $(L = 1, p = e)$.

B Related Works

B.1 Graph Neural Networks for 3D Atomistic Graphs

Graph neural networks (GNNs) are well adapted to perform property prediction of atomic systems because they can handle discrete and topological structures. There are two main ways to represent

atomistic graphs [76], which are chemical bond graphs, sometimes denoted as 2D graphs, and 3D spatial graphs. Chemical bond graphs use edges to represent covalent bonds without considering 3D geometry. Due to their similarity to graph structures in other applications, generic GNNs [31, 29, 42, 85, 80, 6] can be directly applied to predict their properties [60, 57, 58, 36, 35]. On the other hand, 3D spatial graphs consider positions of atoms in 3D spaces and therefore 3D geometry. Although 3D graphs can faithfully represent atomistic systems, one challenge of moving from chemical bond graphs to 3D spatial graphs is to remain invariant or equivariant to geometric transformation acting on atom positions. Therefore, invariant neural networks and equivariant neural networks have been proposed for 3D atomistic graphs, with the former leveraging invariant information like distances and angles and the latter operating on geometric tensors like type-L vectors.

B.2 Invariant GNNs

Previous works [64, 84, 77, 26, 25, 54, 49, 68, 43] extract invariant information from 3D atomistic graphs and operate on the resulting invariant graphs. They mainly differ in leveraging different geometric information such as distances, bond angles (3 atom features) or dihedral angles (4 atom features). SchNet [64] uses relative distances and proposes continuous-filter convolutional layers to learn local interaction between atom pairs. DimeNet series [26, 25] incorporate bond angles by using triplet representations of atoms. SphereNet [49] and GemNet [43, 27] further extend to consider dihedral angles for better performance. In order to consider directional information contained in angles, they rely on triplet or quadruplet representations of atoms. In addition to being memory-intensive [70], they also change graph structures by introducing higher-order interaction terms [11], which would require non-trivial modifications to generic GNNs in order to apply them to 3D graphs. In contrast, the proposed Equiformer uses equivariant irreps features to consider directional information without complicating graph structures and therefore can directly inherit the design of generic GNNs.

B.3 Attention and Transformer

Graph Attention. Graph attention networks (GAT) [80, 6] use multi-layer perceptrons (MLP) to calculate attention weights in a similar manner to message passing networks. Subsequent works using graph attention mechanisms follow either GAT-like MLP attention [8, 41] or Transformer-like dot product attention [88, 24, 67, 18, 41, 45]. In particular, Kim *et al.* [41] compares these two types of attention mechanisms empirically under a self-supervised setting. Brody *et al.* [6] analyzes their theoretical differences and compares their performance in general settings.

Graph Transformer. A different line of research focuses on adapting standard Transformer networks to graph problems [18, 59, 45, 86, 66]. They adopt dot product attention in Transformers [79] and propose different approaches to incorporate graph-related inductive biases into their networks. GROVE [59] includes additional message passing layers or graph convolutional layers to incorporate local graph structures when calculating attention weights. SAN [45] proposes to learn position embeddings of nodes with full Laplacian spectrum. Graphormer [86] proposes to encode degree information in centrality embeddings and encode distances and edge features in attention biases. The proposed Equiformer belongs to one of these attempts to generalize standard Transformers to graphs and is dedicated to 3D graphs. To incorporate 3D-related inductive biases, we adopt an equivariant version of Transformers with irreps features and propose novel equivariant graph attention.

C Details of Architecture

C.1 Equivariant Operation Used in Equiformer

We illustrate the equivariant operations used in Equiformer in Fig. 2 and provide an alternative visualization of depth-wise tensor products in Fig. 3.

C.2 Equiformer Architecture

For simplicity and because most works we compare with do not include equivariance to inversion, we adopt $SE(3)$-equivariant irreps features in Equiformer for experiments in the main text and note that $E(3)$-equivariant irreps features can be easily incorporated into Equiformer.

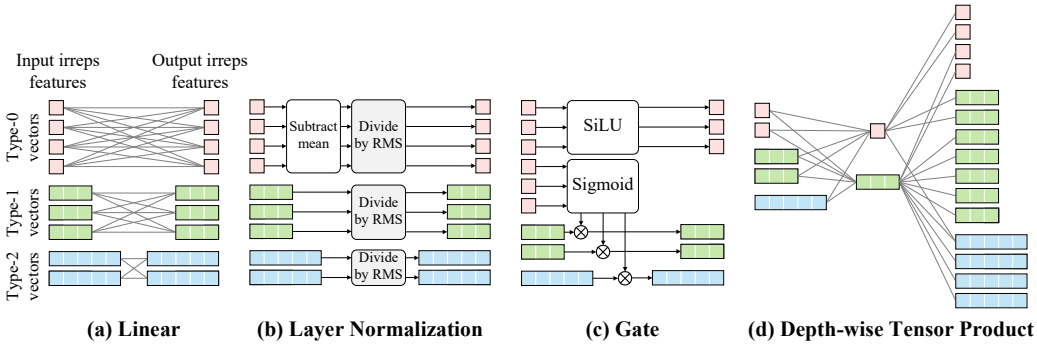


Figure 2: **Equivariant operations used in Equiformer.** **(a)** Each gray line between input and output irreps features contain one learnable weight. Note that the number of output channels can be different from that of input channels. **(b)** "RMS" denotes the root mean square value (RMS) along the channel dimension. For simplicity, in this figure, we have removed multiplying by γ. **(c)** Gate layers are equivariant activation functions where non-linearly transformed scalars are used to gate non-scalar irreps features. **(d)** The left two irreps features correspond to the two input irreps features, and the rightmost one is the output irreps feature. The two gray lines connecting two vectors in the input irreps features and one vector in the output irreps feature form a path and contain one learnable weight. We only show $SE(3)$-equivariant operations in this figure and note that they can be directly generalized to $E(3)$-equivariant features.

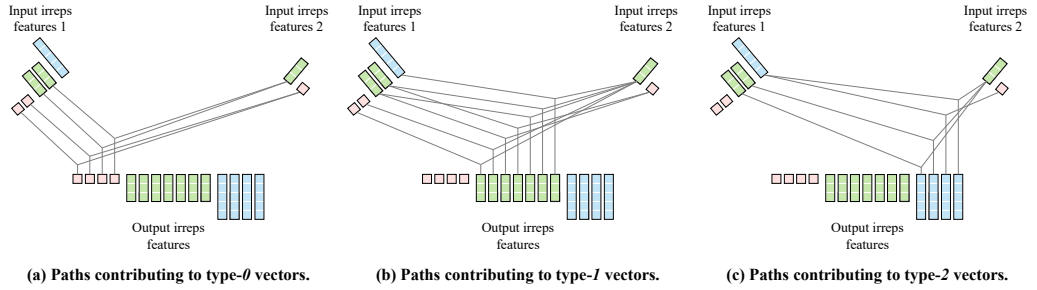


Figure 3: **An alternative visualization of the depth-wise tensor product.** We follow the visualization of tensor products in `e3nn` [28] and separate paths into three parts based on the types of output vectors.

We define architectural hyper-parameters like the number of channels in some layers in Equiformer, which are used to specify the detailed architectures in Sec. D and Sec. E.

We use d_{embed} to denote embedding dimension, which defines the dimension of most irreps features. Specifically, all irreps features x_i, y_i in Fig. 1 have dimension d_{embed} unless otherwise stated. Besides, we use d_{sh} to represent the dimension of spherical harmonics embeddings of relative positions in all depth-wise tensor products.

For equivariant graph attention in Fig. 1(b), the first two linear layers have the same output dimension d_{embed}. The output dimension of depth-wise tensor products (DTP) are determined by that of input irreps features. Equivariant graph attention consists of h parallel attention functions, and the value vector in each attention function has dimension d_{head}. We refer to h and d_{head} as the number of heads and head dimension, respectively. By default, we set the number of channels in scalar feature $f_{ij}^{(0)}$ to be the same as the number of channels of type-0 or type-$(0, e)$ vectors in v_{ij}. When non-linear messages are adopted in v_{ij}, we set the dimension of output irreps features in gate activation to be $h \times d_{head}$. Therefore, we can use two hyper-parameters h and d_{head} to specify the detailed architecture of equivariant graph attention.

As for feed forward networks (FFNs), we denote the dimension of output irreps features in gate activation as d_{ffn}. The FFN in the last Transformer block has output dimension $d_{feature}$, and we

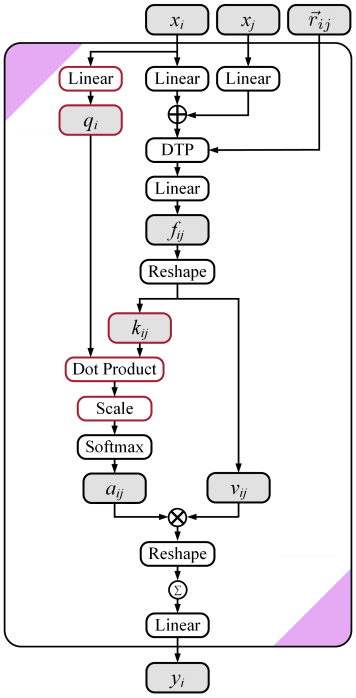


Figure 4: **Architecture of equivariant dot product attention without non-linear message passing.** In this figure, "$\otimes$" denotes multiplication, "$\oplus$" denotes addition, and "DTP" stands for depth-wise tensor product. $\sum$ within a circle denotes summation over all neighbors. Gray cells indicate intermediate irreps features. We highlight the difference of dot product attention from multi-layer perceptron attention in red. Note that key k_{ij} and value v_{ij} are irreps features and therefore f_{ij} in dot product attention typically has more channels than that in multi-layer perceptron attention.

set d_{ffn} of the last FFN, which is followed by output head, to be $d_{feature}$ as well. Thus, two hyper-parameters d_{ffn} and $d_{feature}$ are used to specify architectures of FFNs and the output dimension after Transformer blocks.

Irreps features contain channels of vectors with degrees up to L_{max}. We denote C_L type-L vectors as (C_L, L) and $C_{(L,p)}$ type-(L, p) vectors as $(C_{(L,p)}, L, p)$ and use brackets to represent concatenations of vectors. For example, the dimension of irreps features containing 256 type-0 vectors and 128 type-1 vectors can be represented as $[(256, 0), (128, 1)]$.

C.3 Dot Product Attention

We illustrate the dot product attention without non-linear message passing used in ablation study in Fig. 4. The architecture is adapted from SE(3)-Transformer [23]. The difference from multi-layer perceptron attention lies in how we obtain attention weights a_{ij} from f_{ij}. We split f_{ij} into two irreps features, key k_{ij} and value v_{ij}, and obtain query q_i with a linear layer. Then, we perform scaled dot product [79] between q_i and k_{ij} for attention weights.

D Details of Experiments on QM9

D.1 Additional Comparison between $SE(3)$ and $E(3)$ Equivariance

We train two versions of Equiformers, one with $SE(3)$-equivariant features denoted as "Equiformer" and the other with $E(3)$-equivariant features denoted as "$E(3)$-Equiformer", and we compare them in Table 9. Including equivariance to inversion further improves the performance on QM9 dataset.

As for Table 1, we compare "Equiformer" with other works since most of them do not include equivariance to inversion.

Methods	Task Units	α bohr^3	$\Delta\varepsilon$ meV	$\varepsilon_{\text{HOMO}}$ meV	$\varepsilon_{\text{LUMO}}$ meV	μ D	C_ν cal/mol K
Equiformer		.056	33	17	16	.014	.025
$E(3)$-Equiformer		.054	32	16	16	.013	.024

Table 9: **Ablation study of $SE(3)$/$E(3)$ equivariance on QM9 testing set.** "Equiformer" operates on $SE(3)$-equivariant features while "$E(3)$-Equiformer" uses $E(3)$-equivariant features. Including inversion further improves mean absolute errors.

Hyper-parameters	Value or description
Optimizer	AdamW
Learning rate scheduling	Cosine learning rate with linear warmup
Warmup epochs	5
Maximum learning rate	5×10^{-4}
Batch size	128
Number of epochs	300
Weight decay	5×10^{-3}
Dropout rate	0.1, 0.2
Cutoff radius (Å)	5
Number of radial bases	128 for Gaussian radial basis, 8 for radial bessel basis
Hidden sizes of radial functions	64
Number of hidden layers in radial functions	2
	Equiformer
Number of Transformer blocks	6
Embedding dimension d_{embed}	$[(128, 0), (64, 1), (32, 2)]$
Spherical harmonics embedding dimension d_{sh}	$[(1, 0), (1, 1), (1, 2)]$
Number of attention heads h	4
Attention head dimension d_{head}	$[(32, 0), (16, 1), (8, 2)]$
Hidden dimension in feed forward networks d_{ffn}	$[(384, 0), (192, 1), (96, 2)]$
Output feature dimension $d_{feature}$	$[(512, 0)]$
	$E(3)$-Equiformer
Number of Transformer blocks	6
Embedding dimension d_{embed}	$[(128, 0, e), (32, 0, o), (32, 1, e), (32, 1, o), (16, 2, e), (16, 2, o)]$
Spherical harmonics embedding dimension d_{sh}	$[(1, 0, e), (1, 1, o), (1, 2, e)]$
Number of attention heads h	4
Attention head dimension d_{head}	$[(32, 0, e), (8, 0, o), (8, 1, e), (8, 1, o), (4, 2, e), (4, 2, o)]$
Hidden dimension in feed forward networks d_{ffn}	$[(384, 0, e), (96, 0, o), (96, 1, e), (96, 1, o), (48, 2, e), (48, 2, o)]$
Output feature dimension $d_{feature}$	$[(512, 0, e)]$

Table 10: **Hyper-parameters for QM9 dataset.** We denote C_L type-L vectors as (C_L, L) and $C_{(L,p)}$ type-(L, p) vectors as $(C_{(L,p)}, L, p)$ and use brackets to represent concatenations of vectors.

D.2 Training Details

We normalize ground truth by subtracting mean and dividing by standard deviation. For the task of U, U_0, G, and H, where single-atom reference values are available, we subtract those reference values from ground truth before normalizing.

We train Equiformer with 6 blocks with $L_{max} = 2$ following SEGNN [5]. We choose Gaussian radial basis [64, 68, 43, 66] for the first six tasks in Table 1 and radial Bessel basis [26, 25] for the others. We apply dropout [71] to attention weights a_{ij}. The dropout rate is 0.1 for the task of R^2 and 0.2 for others. Table 10 summarizes the hyper-parameters for the QM9 dataset. The detailed description of architectural hyper-parameters can be found in Sec. C.2.

We use one A6000 GPU with 48GB to train each model and summarize the computational cost of training for one epoch as follows. Training $E(3)$-Equiformer for one epoch takes about 14.75 minutes. The time of training Equiformer, Equiformer with linear messages (indicated by index 2 in Table 5), and Equiformer with linear messages and dot product attention (indicated by index 3 in Table 5) for one epoch is 11 minutes, 6.6 minutes and 7.1 minutes, respectively.

Methods	Energy MAE (eV) ↓					EwT (%) ↑				
	ID	OOD Ads	OOD Cat	OOD Both	Average	ID	OOD Ads	OOD Cat	OOD Both	Average
Equiformer	0.5088	0.6271	0.5051	0.5545	0.5489	4.88	2.93	4.92	2.98	3.93
$E(3)$-Equiformer	0.5035	0.6385	0.5034	0.5658	0.5528	5.10	2.98	5.10	3.02	4.05

Table 11: **Ablation study of $SE(3)$/$E(3)$ equivariance on OC20 IS2RE validation set.** "Equiformer" operates on $SE(3)$-equivariant features while "$E(3)$-Equiformer" uses $E(3)$-equivariant features.

Hyper-parameters	Value or description
Optimizer	AdamW
Learning rate scheduling	Cosine learning rate with linear warmup
Warmup epochs	2
Maximum learning rate	2×10^{-4}
Batch size	32
Number of epochs	20
Weight decay	1×10^{-3}
Dropout rate	0.2
Cutoff radius (Å)	5
Number of radial basis	128
Hidden size of radial function	64
Number of hidden layers in radial function	2
Equiformer	
Number of Transformer blocks	6
Embedding dimension d_{embed}	$[(256, 0), (128, 1)]$
Spherical harmonics embedding dimension d_{sh}	$[(1, 0), (1, 1)]$
Number of attention heads h	8
Attention head dimension d_{head}	$[(32, 0), (16, 1)]$
Hidden dimension in feed forward networks d_{ffn}	$[(768, 0), (384, 1)]$
Output feature dimension $d_{feature}$	$[(512, 0)]$
$E(3)$-Equiformer	
Number of Transformer blocks	6
Embedding dimension d_{embed}	$[(256, 0, e), (64, 0, o), (64, 1, e), (64, 1, o)]$
Spherical harmonics embedding dimension d_{sh}	$[(1, 0, e), (1, 1, o)]$
Number of attention heads h	8
Attention head dimension d_{head}	$[(32, 0, e), (8, 0, o), (8, 1, e), (8, 1, o)]$
Hidden dimension in feed forward networks d_{ffn}	$[(768, 0, e), (192, 0, o), (192, 1, e), (192, 1, o)]$
Output feature dimension $d_{feature}$	$[(512, 0, e)]$

Table 12: **Hyper-parameters for OC20 dataset under the setting of training without IS2RS auxiliary task.** We denote C_L type-L vectors as (C_L, L) and $C_{(L,p)}$ type-(L, p) vectors as $(C_{(L,p)}, L, p)$ and use brackets to represent concatenations of vectors.

E Details of Experiments on OC20

E.1 Additional Comparison between $SE(3)$ and $E(3)$ Equivariance

We train two versions of Equiformers, one with $SE(3)$-equivariant features denoted as "Equiformer" and the other with $E(3)$-equivariant features denoted as "$E(3)$-Equiformer", and we compare them in Table 11. Including inversion improves the MAE results on ID and OOD Cat sub-splits but degrades the performance on the other sub-splits. Overall, using $E(3)$-equivariant features results in slightly inferior performance. We surmise the reasons are as follows. First, inversion might not be the key bottleneck. Second, including inversion would break type-1 vectors into two parts, type-$(1, e)$ and type-$(1, o)$ vectors. They are regarded as different types in equivariant linear layers and layer normalizations, and therefore, the directional information captured in these two types of vectors can only exchange in depth-wise tensor products. Third, we mainly tune hyper-parameters for Equiformer with $SE(3)$-equivariant features, and it is possible that using $E(3)$-equivariant features would favor different hyper-parameters.

For Table 2, 3 and 4 in the main text and Table 7 and 8 in appendix, we compare "Equiformer" with other works since most of them do not include equivariance to inversion.

E.2 Training Details

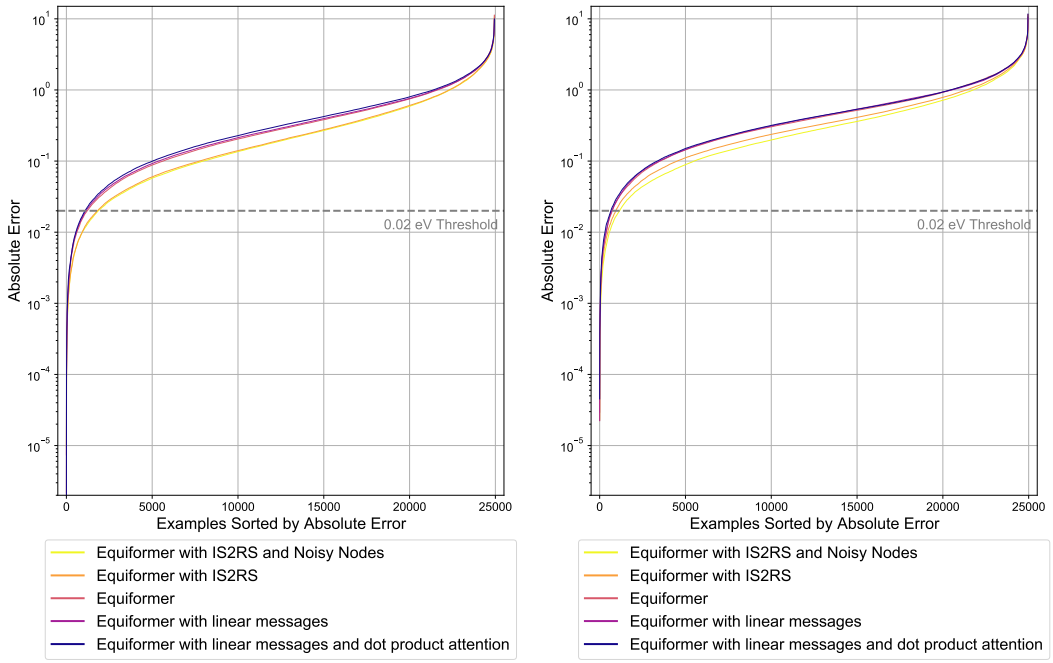
IS2RE without Node-Level Auxiliary Task. We use hyper-parameters similar to those for QM9 dataset and summarize in Table 12. The detailed description of architectural hyper-parameters can be found in Sec. C.2.

IS2RE with IS2RS Node-Level Auxiliary Task. We increase the number of Transformer blocks to 18 as deeper networks can benefit more from IS2RS node-level auxiliary task [30]. We follow the same hyper-parameters in Table 12 except that we increase maximum learning rate to 5×10^{-4} and set $d_{feature}$ to $[(512, 0), (256, 1)]$. Additionally, we use stochastic depth with probability 0.05 [37]. Inspired by Graphormer [66], we add an extra equivariant graph attention module after the last layer normalization to predict relaxed structures and use a linearly decayed weight for loss associated with IS2RS, which starts at 15 and decays to 1. When Noisy Nodes [30] data augmentation is used, we increase the number of epochs to 40.

We use two A6000 GPUs, each with 48GB, to train models when IS2RS is not included during training. Training Equiformer takes about 43.6 hours. Training Equiformer with linear messages (indicated by index 2 in Table 6) and Equiformer with linear messages and dot product attention (indicated by index 3 in Table 6) takes 30.4 hours and 33.1 hours, respectively. We use four A6000 GPUs to train Equiformer models when IS2RS node-level auxiliary task is adopted during training. Training Equiformer without Noisy Nodes [30] data augmentation takes about 3 days and training with Noisy Nodes takes 6 days. We note that the proposed Equiformer in Table 8 achieves competitive results even with much less computation. Specifically, training "Equiformer + Noisy Nodes" takes about 24 GPU-days when A6000 GPUs are used. The training time of "GNS + Noisy Nodes" [30] is 56 TPU-days. "Graphormer" [66] uses ensemble of 31 models and requires 372 GPU-days to train all models when A100 GPUs are used.

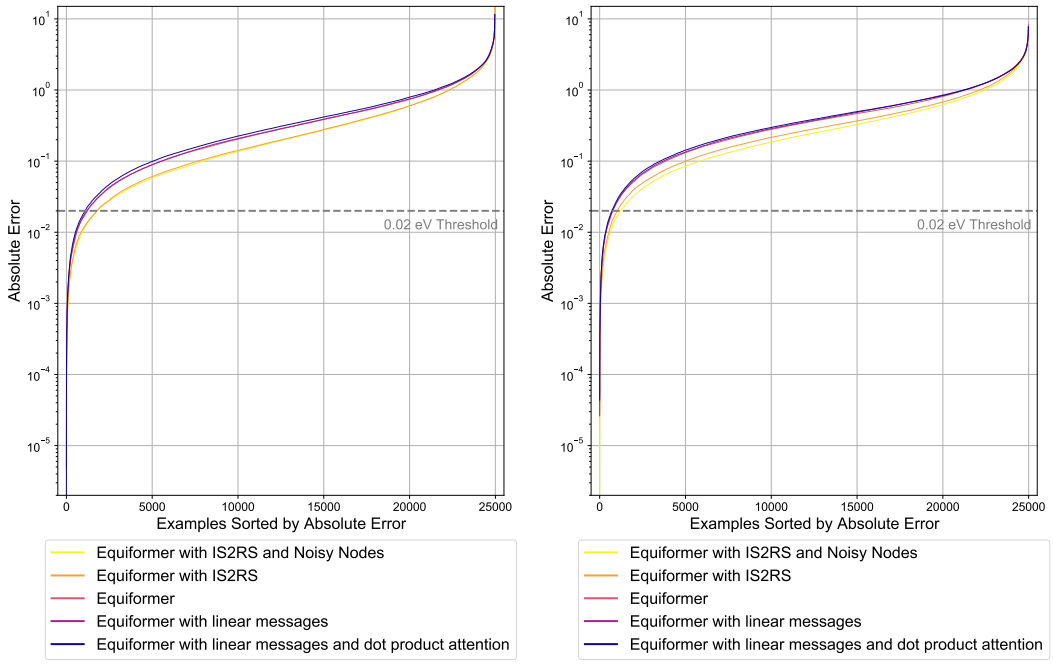
E.3 Error Distributions

We plot the error distributions of different Equiformer models on different sub-splits of OC20 IS2RE validation set in Fig. 5. For each curve, we sort the absolute errors in ascending order for better visualization and have a few observations. First, for each sub-split, there are always easy examples, for which all models achieve significantly low errors, and hard examples, for which all models have high errors. Second, the performance gains brought by different models are non-uniform among different sub-splits. For example, using MLP attention and non-linear messages improves the errors on the ID sub-split but is not that helpful on the OOD Ads sub-split. Third, when IS2RS node-level auxiliary task is not included during training, using stronger models mainly improves errors that are beyond the threshold of 0.02 eV, which is used to calculate the metric of energy within threshold (EwT). For instance, on the OOD Both sub-split, using non-linear messages, which corresponds to red and purple curves, improves the absolute errors for the 15000th through 20000th examples. However, the improvement in MAE does not translate to that in EwT as the errors are still higher than the threshold of 0.02 eV. This explains why using non-linear messages in Table 6 improves MAE from 0.5657 to 0.5545 but results in almost the same EwT.



(a) **ID sub-split.**

(b) **OOD Ads sub-split.**



(c) **OOD Cat sub-split.**

(d) **OOD Both sub-split.**

Figure 5: **Error distributions of different Equiformer models on different sub-splits of OC20 IS2RE validation set.**

References

[1] Brandon Anderson, Truong-Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Conference on Neural Information Processing (NeurIPS)*, 2019.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arxiv preprint arxiv:1607.06450*, 2016.

[3] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *Conference on Neural Information Processing (NeurIPS)*, 2020.

[4] Simon Batzner, Albert Musaelian, Lixin Sun, Mario Geiger, Jonathan P. Mailoa, Mordechai Kornbluth, Nicola Molinari, Tess E. Smidt, and Boris Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1), May 2022.

[5] Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J Bekkers, and Max Welling. Geometric and physical quantities improve e(3) equivariant message passing. In *International Conference on Learning Representations (ICLR)*, 2022.

[6] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Conference on Neural Information Processing (NeurIPS)*, 2020.

[8] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. Relational graph attention networks. *arxiv preprint arxiv:1904.05811*, 2019.

[9] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020.

[10] Lowik Chanussot*, Abhishek Das*, Siddharth Goyal*, Thibaut Lavril*, Muhammed Shuaibi*, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, Aini Palizhati, Anuroop Sriram, Brandon Wood, Junwoong Yoon, Devi Parikh, C. Lawrence Zitnick, and Zachary Ulissi. Open catalyst 2020 (oc20) dataset and community challenges. *ACS Catalysis*, 2021.

[11] Zhengdao Chen, Lisha Li, and Joan Bruna. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[12] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning (ICML)*, 2016.

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

[14] George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arxiv preprint arxiv:1810.04805*, 2019.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[17] Mildred S Dresselhaus, Gene Dresselhaus, and Ado Jorio. *Group theory*. Springer, Berlin, Germany, 2008 edition, March 2007.

[18] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. 2020.

[19] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*, 2017.

[20] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[21] Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. *arXiv preprint arXiv:2002.12880*, 2020.

[22] Nathan Frey, Ryan Soklaski, Simon Axelrod, Siddharth Samsi, Rafael Gomez-Bombarelli, Connor Coley, and Vijay Gadepally. Neural scaling of deep chemical models. *ChemRxiv*, 2022.

[23] Fabian Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks. In *Conference on Neural Information Processing (NeurIPS)*, 2020.

[24] Hongyang Gao and Shuiwang Ji. Graph representation learning via hard and channel-wise attention networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[25] Johannes Gasteiger, Shankari Giri, Johannes T. Margraf, and Stephan Günnemann. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. In *Machine Learning for Molecules Workshop, NeurIPS*, 2020.

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

[27] Johannes Gasteiger, Muhammed Shuaibi, Anuroop Sriram, Stephan Günnemann, Zachary Ulissi, C. Lawrence Zitnick, and Abhishek Das. How do graph networks generalize to large and diverse molecular systems? *arxiv preprint arxiv:2204.02782*, 2022.

[28] Mario Geiger, Tess Smidt, Alby M., Benjamin Kurt Miller, Wouter Boomsma, Bradley Dice, Kostiantyn Lapchevskyi, Maurice Weiler, Michał Tyszkiewicz, Simon Batzner, Dylan Madisetti, Martin Uhrin, Jes Frellsen, Nuri Jung, Sophia Sanborn, Mingjian Wen, Josh Rackers, Marcel Rød, and Michael Bailey. e3nn/e3nn: 2022-04-13, April 2022.

[29] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017.

[30] Jonathan Godwin, Michael Schaarschmidt, Alexander L Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Veličković, James Kirkpatrick, and Peter Battaglia. Simple GNN regularisation for 3d molecular property prediction and beyond. In *International Conference on Learning Representations (ICLR)*, 2022.

[31] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Conference on Neural Information Processing (NeurIPS)*, 2017.

[32] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[33] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[34] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[35] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[36] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[37] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, 2016.

[38] Weile Jia, Han Wang, Mohan Chen, Denghui Lu, Lin Lin, Roberto Car, Weinan E, and Linfeng Zhang. Pushing the limit of molecular dynamics with ab initio accuracy to 100 million atoms with machine learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press, 2020.

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

[40] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. 2021.

[41] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations (ICLR)*, 2021.

[42] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[43] Johannes Klicpera, Florian Becker, and Stephan Günnemann. Gemnet: Universal directional graph neural networks for molecules. In *Conference on Neural Information Processing (NeurIPS)*, 2021.

[44] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. In *Advances in Neural Information Processing Systems 32*, pages 10117–10126, 2018.

[45] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *Conference on Neural Information Processing (NeurIPS)*, 2021.

[46] Cheng-I Lai, Yang Zhang, Alexander H. Liu, Shiyu Chang, Yi-Lun Liao, Yung-Sung Chuang, Kaizhi Qian, Sameer Khurana, David Daniel Cox, and James R. Glass. PARP: Prune, adjust and re-prune for self-supervised speech recognition. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Conference on Neural Information Processing (NeurIPS)*, 2021.

[47] Tuan Le, Frank Noé, and Djork-Arné Clevert. Equivariant graph attention networks for molecular property prediction. *arXiv preprint arXiv:2202.09891*, 2022.

[48] Yi-Lun Liao, Sertac Karaman, and Vivienne Sze. Searching for efficient multi-stage vision transformers. *arxiv preprint arxiv:2109.00642*, 2021.

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

[50] Denghui Lu, Han Wang, Mohan Chen, Lin Lin, Roberto Car, Weinan E, Weile Jia, and Linfeng Zhang. 86 pflops deep potential molecular dynamics simulation of 100 million atoms with ab initio accuracy. *Computer Physics Communications*, 259:107624, 2021.

[51] Benjamin Kurt Miller, Mario Geiger, Tess E. Smidt, and Frank Noé. Relevance of rotationally equivariant convolutions for predicting molecular properties. *arxiv preprint arxiv:2008.08461*, 2020.

[52] Albert Musaelian, Simon Batzner, Anders Johansson, Lixin Sun, Cameron J. Owen, Mordechai Kornbluth, and Boris Kozinsky. Learning local equivariant representations for large-scale atomistic dynamics. *arxiv preprint arxiv:2204.05249*, 2022.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[54] Zhuoran Qiao, Matthew Welborn, Animashree Anandkumar, Frederick R. Manby, and Thomas F. Miller. OrbNet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal of Chemical Physics*, 153(12):124111, sep 2020.

[55] Joshua A. Rackers, Lucas Tecot, Mario Geiger, and Tess E. Smidt. Cracking the quantum scaling limit with machine learned electron densities. *arxiv preprint arxiv:2201.03726*, 2022.

[56] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[57] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

[58] Bharath Ramsundar, Peter Eastman, Patrick Walters, Vijay Pande, Karl Leswing, and Zhenqin Wu. *Deep Learning for the Life Sciences*. O'Reilly Media, 2019. `https://www.amazon.com/Deep-Learning-Life-Sciences-Microscopy/dp/1492039837`.

[59] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *Conference on Neural Information Processing (NeurIPS)*, 2020.

[60] Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, 2012. PMID: 23088335.

[61] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[62] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning (ICML)*, 2020.

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

[64] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *Conference on Neural Information Processing (NeurIPS)*, 2017.

[65] Kristof T. Schütt, Oliver T. Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning (ICML)*, 2021.

[66] Yu Shi, Shuxin Zheng, Guolin Ke, Yifei Shen, Jiacheng You, Jiyan He, Shengjie Luo, Chang Liu, Di He, and Tie-Yan Liu. Benchmarking graphormer on large-scale molecular modeling datasets. *arxiv preprint arxiv:2203.04810*, 2022.

[67] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arxiv preprint arxiv:2009.03509*, 2020.

[68] Muhammed Shuaibi, Adeesh Kolluru, Abhishek Das, Aditya Grover, Anuroop Sriram, Zachary Ulissi, and C. Lawrence Zitnick. Rotation invariant graph neural networks using spin convolutions. *arxiv preprint arxiv:2106.09575*, 2021.

[69] Tess E. Smidt, Mario Geiger, and Benjamin Kurt Miller. Finding symmetry breaking order parameters with euclidean neural networks. *Physical Review Research*, 3(1), jan 2021.

[70] Anuroop Sriram, Abhishek Das, Brandon M Wood, and C. Lawrence Zitnick. Towards training billion parameter graph neural networks for atomic simulations. In *International Conference on Learning Representations*, 2022.

[71] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[72] Philipp Thölke and Gianni De Fabritiis. Equivariant transformers for neural network based molecular potentials. In *International Conference on Learning Representations (ICLR)*, 2022.

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

[74] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[75] Raphael J. L. Townshend, Brent Townshend, Stephan Eismann, and Ron O. Dror. Geometric prediction: Moving beyond scalars. *arXiv preprint arXiv:2006.14163*, 2020.

[76] Raphael John Lamarre Townshend, Martin Vögele, Patricia Adriana Suriana, Alexander Derry, Alexander Powers, Yianni Laloudakis, Sidhika Balachandar, Bowen Jing, Brandon M. Anderson, Stephan Eismann, Risi Kondor, Russ Altman, and Ron O. Dror. ATOM3d: Tasks on molecules in three dimensions. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[77] Oliver T. Unke and Markus Meuwly. PhysNet: A neural network for predicting energies, forces, dipole moments, and partial charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, may 2019.

[78] Oliver Thorsten Unke, Mihail Bogojeski, Michael Gastegger, Mario Geiger, Tess Smidt, and Klaus Robert Muller. SE(3)-equivariant prediction of molecular wavefunctions and electronic densities. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Conference on Neural Information Processing (NeurIPS)*, 2021.

[79] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Conference on Neural Information Processing (NeurIPS)*, 2017.

[80] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[81] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. In *Advances in Neural Information Processing Systems 32*, pages 10402–10413, 2018.

[82] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019.

[83] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. *arxiv preprint arxiv:1612.04642*, 2016.

[84] Tian Xie and Jeffrey C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, 120(14), apr 2018.

[85] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[86] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Conference on Neural Information Processing (NeurIPS)*, 2021.

[87] A. Zee. *Group Theory in a Nutshell for Physicists*. Princeton University Press, USA, 2016.

[88] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 339–349, 2018.

[89] Linfeng Zhang, Jiequn Han, Han Wang, Roberto Car, and Weinan E. Deep potential molecular dynamics: A scalable model with the accuracy of quantum mechanics. *Phys. Rev. Lett.*, 120:143001, Apr 2018.