Clebsch-Gordan Transformer: Fast and Global Equivariant Attention

Owen Lewis Howell,[1,2] Linfeng Zhao,[1] Xupeng Zhu,[1] Yaoyao Qian,[1] Haojie Huang,[1] Lingfeng Sun,[2] Wil Thomason,[2] Robert Platt,[1,2] and Robin Walters[1,2]

[1]*Northeastern University: 360 Huntington Ave, Boston, MA 02115*
[2]*The Boston Dynamics AI Institute: 145 Broadway, Cambridge, MA 02142*

The global attention mechanism is one of the keys to the success of transformer architecture, but it incurs quadratic computational costs in relation to the number of tokens. On the other hand, equivariant models, which leverage the underlying geometric structures of problem instance, often achieve superior accuracy in physical, biochemical, computer vision, and robotic tasks, at the cost of additional compute requirements. As a result, existing equivariant transformers only support low-order equivariant features and local context windows, limiting their expressiveness and performance. This work proposes Clebsch-Gordan Transformer, achieving efficient global attention by a novel Clebsch-Gordon Convolution on SO(3) irreducible representations. Our method enables equivariant modeling of features at all orders while achieving $O(N \log N)$ input token complexity. Additionally, the proposed method scales well with high-order irreducible features, by exploiting the sparsity of the Clebsch-Gordon matrix. Lastly, we also incorporate optional token permutation equivariance through either weight sharing or data augmentation. We benchmark our method on a diverse set of benchmarks including n-body simulation, QM9, ModelNet point cloud classification and a robotic grasping dataset, showing clear gains over existing equivariant transformers in GPU memory size, speed, and accuracy.

I. INTRODUCTION

Transformer-based models have demonstrated effectiveness beyond language processing, showing strong performance in geometry-aware tasks such as robotics, structural biochemistry, and materials science [1–5]. For instance, 3D robotic perception tasks ranging from segmentation to object matching process point clouds and LiDAR data using attention mechanisms. These tasks heavily rely on token-based representations, and their performance is often constrained by the number of tokens the model can effectively handle. AlphaFold [6], for example, employs equivariant transformers to predict protein structures with unprecedented accuracy by explicitly leveraging $\mathrm{SE}(3)$ symmetries such as rotations and translations. However, implementing an equivariant neural network structure typically incurs significant computational overhead and increased inference time. As a result, most current approaches are limited to small symmetry groups or low-order representations [7–12]. Enabling fast, low-memory overhead equivariant operations over large context windows is essential to scaling robust and sample-efficient learning in geometry-aware domains.

Unfortunately, maintaining equivariance while modeling a global geometric context is challenging due to the computational demands of processing high dimensional data at scale. There are essentially two components that contribute to the computational complexity of $E(3)$-equivariant transformers: the time and memory scaling of the transformer with the number of tokens, N, and the time and memory complexity on the maximum harmonic degree, ℓ. Naively, a global equivariant attention mechanism will have $O(N^2)$ token complexity and $O(\ell^6)$ harmonic complexity [13]. By assuming only local attention, the token complexity can be reduced to $O(dN)$, where d is the local context window, at the cost of discarding information about long range correlations. In addition, various approximation techniques have been used to reduce the harmonic complexity to $O(\ell^3)$ [14]. Recently, SE(3)-Hyena achieved $O(N \log N)$ computational complexity using long convolution [15, 16] in the Fourier domain. However, it only supports up to first-order irreducible representations of SO(3) (i.e., scalar and vectors), making it difficult to capture higher-degree angular dependencies and limiting its ability to represent more complex, structured geometric patterns. Moreover, SE(3)-Hyena is not permutation invariant, making it most suitable for point clouds with a natural ordering.

This raises the question: can we design a method with global equivariant attention and $\mathcal{O}(N \log N)$ token complexity, support for arbitrary orders of spherical harmonics with $\mathcal{O}(\ell^3)$ complexity, and permutation invariance? This work addresses this challenge by introducing Clebsch-Gordon Convolution on SO(3) irreducible representations of arbitrary degree. We also enforce or encourage permutation invariance through either weight sharing or data augmentation. Our contributions can be summarized as follows:

- We generalize the SE(3)-Hyena method of [9] to include equivariant features of all types. By exploiting the sparsity of the Clebsch-Gordon matrix,our method achieves $O(L^3)$ harmonic scaling.
- By applying our proposed attention in the graph spectral domain, we achieve permutation-equivariant global at-

tention in $O(N \log N)$ time.

- We benchmark our method on a diverse array of tasks, including robotics, computer vision, and molecular biochemistry. Our method outperforms current state of the art methods on all tasks, with considerable reduction in memory usage.

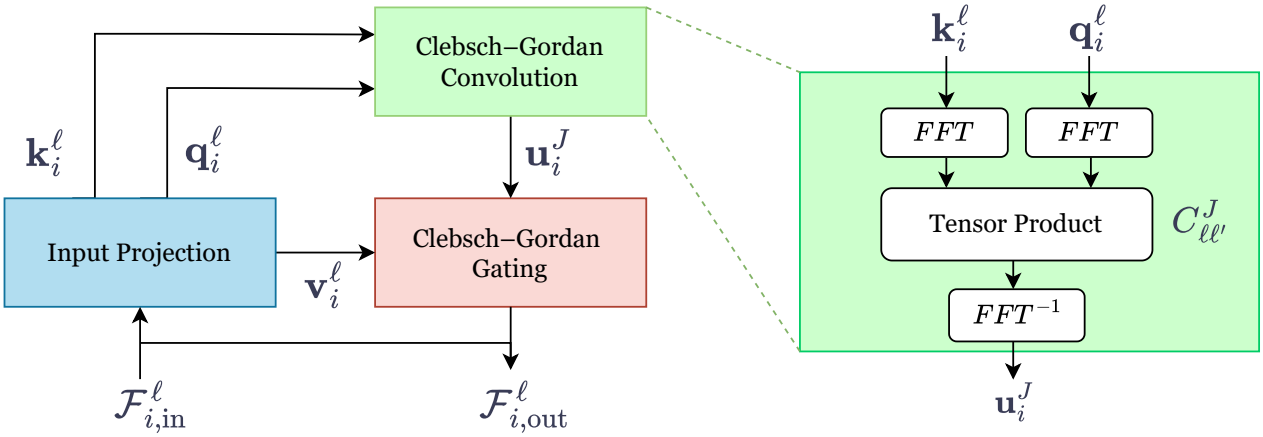

FIG. 1: Schematic of Clebsch-Gordon Convolution. Left: Inputs f_i^ℓ are projected into queries q_i^ℓ, keys k_i^ℓ, and values v_i^ℓ using an $SE(3)$-equivariant projection layer. Queries and keys are passed into a Clebsch-Gordon Convolution which outputs u_i^ℓ with which a tensor product of values is computed. Outputs are added with a residual connection. Right: Queries q_i^ℓ and keys k_i^ℓ are processed using Clebsch-Gordon Convolution. Queries and keys are first fast Fourier transformed, then a tensor product is applied. The output is fast Fourier transformed back.

II. RELATED WORK

a. SE(3)-Equivariance: SE(3)-equivariant neural networks have three main classes: (1) those based on group convolution [17], which discretizes SE(3), transforms a convolutional filter according to each group element in the discretization, and lastly performs cross-correlation using the transformed convolutional filter [18–20]; (2) those based on irreducible (spherical Fourier) representations, which provide a compact representation of SO(3) signals at each point in the point cloud [7, 8, 10, 21–23], and (3) those based on scalar, vector, and multivector representations [9, 24, 25]. Compared with group convolution (class (1)), the irreducible spherical Fourier representation (class (2)) is more compact and avoids discretization errors—among these works, Tensor Field Network (TFN) [7] leverages the tensor product to propagate information between points, SE(3)-Transformer [8] extends TFN by using attention in the Fourier domain for local information passing, and ESCN [22] proposes approximating the tensor product in SO(3) by that in $SO(2)$, significantly reducing computational complexity. Vector representation (class (3)) has limited expressiveness, while the irreducible representation improves expressiveness as its order increases [10]. This work achieves efficient SE(3)-equivariance with high-order irreducible representations by introducing a linear-time attention mechanism based upon the vector long convolution introduced by [9].

b. Subquadratic Attention: Several linear-time attention mechanisms have been proposed to overcome the quadratic time and memory complexity of standard Transformer architectures. Reformer [26] utilizes locality-sensitive hashing (LSH) to approximate self-attention. [27] replaces the standard softmax attention with a kernel-based approximation called FAVOR+ (Fast Attention Via positive Orthogonal Random features), achieving linear time and space complexity. Nyströmformer [28], leverages the Nyström method to approximate the self-attention matrix using a set of landmark points. Linformer [29] addresses the quadratic memory and computation bottleneck of standard Transformers by approximating self-attention with low-rank projections. These methods enable LLM transformers to process much longer sequences than previously feasible.

III. BACKGROUND

Attention. Attention is a data-dependent linear map [30] describing the pairwise interaction between tokens in a transformer's input context. Let q_i, k_i and v_i be linear projections of input. Attention is defined $\text{Attn}(q, k, v) =$

Model	Global Attn.	Perm. Equiv.	Token Complex.	Harmonic Complex.
SEGNN	✗	✓	$\mathcal{O}(dN)$	$\mathcal{O}(L^6)$
SE(3)-Transformer	✗	✓	$\mathcal{O}(dN)$	$\mathcal{O}(L^6)$
Equiformer-v2	✗	✓	$\mathcal{O}(dN)$	$\mathcal{O}(L^3)$
SE(3)-Hyena	✓	✗	$\mathcal{O}(N \log N)$	Type 0 & Type 1 only
Ours	✓	✓/learned	$\mathcal{O}(N \log N)$	$\mathcal{O}(L^3)$
Theoretical Ideal	✓	✓	$\mathcal{O}(N \log N)$	$\mathcal{O}(L^2 \log L)$

TABLE I: Key properties of equivariant attention mechanisms. N: number of tokens, d: average graph degree, L: maximum harmonic degree.

softmax$[\alpha(q,k)]v$ where $\alpha(q,k)_{ij} = q_i^T k_j$ is the attention matrix. The computation of α scales quadratically in the input size, which is the main bottleneck in the transformer architecture in large models. Numerous methods attempt to compute $\alpha(q,k)$ faster using some numerical approximation.

Equivariant Attention and Message Passing. Numerous equivariant methods attempt to generalize attention to process equivarient features. The two most common forms are equivariant attention, or equivariant message passing [21]. In the standard setup, the i-th graph node is located at position $x_i \in \mathbb{R}^3$ has features f_i^ℓ where the index ℓ specifies feature type. Attention and message passing then attempt to process information via update rules

$$\text{Attention: } f_{out,i}^\ell = W_V^{\ell\ell} f_{in,i}^\ell + \sum_{k=0}^{L} \sum_{j \in \mathcal{N}_i} \alpha_{ij} W_V^{\ell k}(x_i - x_j) f_{in,j}^k \tag{1}$$

$$\text{Message Passing: } f_{out,i}^\ell = \phi(f_{in,i}^\ell, \sum_{j \in \mathcal{N}_i} m_{ij}), \;\; m_{ij} = \psi(f_{in,i}^\ell, f_{in,j}^{\ell'}, ||x_i - x_j||) \tag{2}$$

where $\mathcal{N}_i$ is some neighborhood of points around point j. Memory constraints force methods like [7, 8, 11, 13, 21] to restrict to small neighborhoods $\mathcal{N}_i$ of size less than 50. We show in Sec. V that decreasing the size of the local context window can lead to significant changes in performance. A recent method [9] showed that for invariant and vector convolutions adding global context can improve model performance. We extend this work to equivariant features of all types.

IV. METHOD

Let $F^n = \{f_i^\ell\}_{i=1}^n$ be a set of n input features to an SO(3)-equivariant transformer, where SO(3) acts upon each feature $f_i^\ell \in \mathbb{R}^{(2\ell+1)\times m_\ell}$ via its ℓ^{th} irreducible representation. We denote the multiplicity (i.e., channel dimension) of the input of type ℓ as m_ℓ. From the features in F^n, we encode queries, Q_{F^n}, keys, K_{F^n}, and values, V_{F^n}, as:

$$q_i^\ell = W_Q^\ell(f_i^{\ell'}), \quad k_i^\ell = \sum_{\ell'} W_K^\ell(f_i^{\ell'}), \quad v_i^\ell = W_V^\ell(f_i^{\ell'})$$

where W_Q^ℓ, W_K^ℓ, and W_V^ℓ are learnable equivariant mappings converting type ℓ' features of multiplicity m'_ℓ into type ℓ features of multiplicity m_ℓ. Our proposed method is agnostic to the particular equivariant encoders used; see section E 2 for more details. We seek to compute self-attention over Q_{F^n}, K_{F^n}, and V_{F^n} in linear time, scaling to global context, and—unlike earlier work [9]— remaining compatible with equivariant features of any type. Our proposed method extends the core idea of Poli et al. [16], building upon the vector long convolution introduced by Moskalev et al. [9].

A. Clebsch-Gordon Convolution

We structure our attention mechanism as follows: first, inspired by Moskalev et al. [9], we define the following operation, where $C^J_{\ell\ell'}$ is the Clebsch-Gordan matrix projecting from features of type $\ell \otimes \ell'$ onto features of type J:

$$(q^\ell \star k^{\ell'})^J_i = C^J_{\ell\ell'} \sum_{j=1}^{N} q^\ell_j \otimes k^{\ell'}_{i-j} \tag{3}$$

which takes as input features of types ℓ and ℓ' and outputs features of type J. If $q^\ell_i \in \mathbb{R}^{(2\ell+1)m_\ell}$ and $k^\ell_i \in \mathbb{R}^{(2\ell+1)m_\ell}$ the resultant tensor product has dimension $(q^\ell \star k^{\ell'})^J_i \in \mathbb{R}^{(2J+1)m_\ell m_{\ell'}}$. In practice, we found that using multiple heads (which do not interact during the tensor product) led to better performance; see section E for further discussion. Operations of the form $C^J_{\ell\ell} q^\ell \otimes k^{\ell'}$ are ubiquitous in machine learning, making their fast computation a subject of great research interest. For the special case when the keys are spherical harmonic outputs, i.e., $k^\ell = Y^\ell(\hat{n})$, Passaro and Zitnick [13] used a group theoretic decomposition to reduce SO(3) operations into SO(2) operations. Luo et al. [14] generalized this idea and used the Gaunt tensor product coefficients to reduce the tensor product computation to a highly tractable two dimensional Fourier transformation for general input features. We compute the tensor product in eq. (3) using a slight modification of the methods proposed in [14]; see section A for details. The operation in eq. (3), picks the type J output out of the tensor product. By definition of the Clebsch-Gordon matrix $C^J_{\ell\ell'}$, the tensor product of type ℓ and type ℓ' features decomposes as

$$(q^\ell_i \otimes k^{\ell'}_{i-j}) = \bigoplus_J C^J_{\ell\ell'} (q^\ell_i \otimes k^{\ell'}_{i-j})^J \tag{4}$$

where $\bigoplus$ is the direct sum of vector spaces. To allow all query and key types to interact, we want to compute, for each J, $\hat{u}^J_i = \sum_{\ell\ell'} (q^\ell_i \star k^{\ell'}_i)^J$. Following the notation of Luo et al. [14], let $\tilde{q}_i = [q^0_i, q^1_i, ..., q^L_i]$ be the stack of all query vectors containing irreducibles of up to degree L and let $\tilde{k}_i = [k^0_i, k^1_i, ..., k^L_i]$ be the stack of all keys containing irreducibles of up to degree L. The full tensor product of these features for output type J is given by

$$u^J_i = (\tilde{q}_i \star \tilde{k}_i)^J = \sum_{\ell=1}^{L} \sum_{\ell'=1}^{L} (q^\ell_i \star k^{\ell'}_i)^J$$

Naively retaining all output-irreducible types of the Clebsch-Gordan tensor product up to type L requires $O(L^3)$ 3D matrix multiplications, for a total complexity of $O(L^6)$. We then define the full convolution as $\tilde{u}^J_i = (\tilde{q}_i \star \tilde{k}_i)$. This convolution computation can be simplified in two ways. The Fourier transform $\hat{u}^\ell_q$ of u^ℓ_i over the spatial index can be written as $\hat{u}^J_i = C^J_{\ell\ell'} \hat{q}^\ell_i \otimes \hat{k}^{\ell'}_i$ which is a matrix multiplication in Fourier space. Using the Fast Fourier Transform, the computation of $\hat{q}^\ell$ and $\hat{k}^\ell$ can be done in time $O(N \log N)$. We further consider both intra-channel and inter-channel tensor products; see section E for additional ablation studies.

1. *Invariant Gating*

A key aspect of the transform proposed in [9] is its non-linear data dependent gating. Accordingly, after obtaining $\hat{u}^J_q$, we compute a set of invariant features $I^\ell = \gamma^\ell(\hat{u}^0, \hat{u}^1, \hat{u}^2, ...)$ with one I^ℓ for each irreducible type ℓ. We evaluated a variety of encoder types for γ^ℓ; see section E 4 for ablation studies. The gating I^J is of dimension $N \times m_J$. We then apply softmax gating $u^J_i \to \sigma(I^\ell) u^J_i$ and combine the resultant gated features u^J_i with the values via another tensor product and Clebsch-Gordon matrix projection

$$f^J_{i,out} = \sum_{\ell\ell} C^J_{\ell\ell'} u^\ell_i \otimes v^{\ell'}_i \tag{5}$$

Note that the second multiplication, eq. (5) is done in real space rather than Fourier space. The idea of switching between real and Fourier space when computing attention is a key idea developed by Poli et al. [16], Stachenfeld et al. [31], allowing the model to fuse both global and local information. Lastly, we apply an equivariant MLP to reduce the multiplicity dimension back down to that of the input $\hat{f}^\ell_i \to \text{MLP}(\hat{f}^\ell_i)$. We then add the attention features to the input features as a residual via

$$f^\ell_{i,out} = f^\ell_{i,in} + \text{MLP}(f^\ell_{i,in})$$

By subtracting off and re-adding the mean of inputs f^ℓ_i, the outputs $\hat{f}^\ell_{i,out}$ are be fully SE(3)-equivariant.

Algorithm 1: Clebsch-Gordon Convolution

Input : Input signal $f^{\ell}_{i,in}$
Output: Output Attention $f^{J}_{i,out}$

Encode: $q^{\ell}_i = W^{\ell}_Q(f^{\ell'}_i), k^{\ell}_i = W^{\ell}_K(f^{\ell'}_i), v^{\ell}_i = W^{\ell}_V(f^{\ell'}_i)$
FFT: $q^{\ell}_i, k^{\ell}_i \to \hat{q}^{\ell}_i, \hat{k}^{\ell}_i$
Tensor Product: $u^J_i = C^J_{\ell\ell'}\hat{q}^{\ell}_i \otimes \hat{k}^{\ell'}_i$
Inverse FFT: $\hat{u}^{\ell}_i \to u^{\ell}_i$
Tensor Product: $f^J_{i,out} = C^J_{\ell\ell'} u^{\ell}_i \otimes v^{\ell'}_i$
return $f^J_{i,out}$

In practice, we found that using a head dimension of 4 or 8 with a maximum harmonic of $L = 5$ of $L = 6$ and a channel dimension of 8 or 16 was optimal. See section E 3 for ablations on model parameters. In general, we found that using a fixed local context window allowed for much greater performance. Specifically, out full output features are

$$f^{\ell}_{i,out} = \mathcal{F}^{CG}(f^{\ell}_{i,in}) + \mathcal{F}^{SGNN}(f^{\ell}_{i,in})$$

where $\mathcal{F}^{CG}(f^{\ell}_{i,in})$ is the output of the Clebsch-Gordon convolution and $\mathcal{F}^{SGNN}(f^{\ell}_{i,in})$ is a standard equiformer layer [13], with a fixed local context window. We conjecture using both global and local attention that this allows the model to better process both local and global information. This idea is inspired by Han et al. [32], where it is conjectured that state space models combined with some small local attention can capture both global and local features.

B. Architecture

Figure 1 shows the full flow of our proposed attention block. Specific choices for architecture for experiments is discussed more in E.

C. Sparsity of the Clebsch-Gordon Matrix

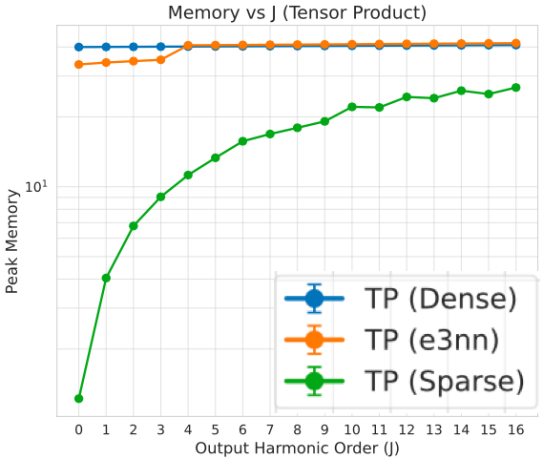

FIG. 2: Memory usage versus the number of irreducible representations J for our tensor product attention mechanism. We compare with dense matrix multiplication and e3nn[33].

We now turn to improving the performance of our proposed attention mechanism by exploiting sparsity properties of the Clebsch-Gordon matrix. Consider the operation defined by$u^J = \sum_{\ell\ell'} C^J_{\ell\ell'} q^{\ell} \otimes k^{\ell'}$. The naive cost of this operation is $\mathcal{O}((2J+1)(2\ell+1)(2\ell'+1))$. Thus, the total computation of u^{ℓ} for each ℓ is naively $O(J\ell\ell') = JL^4$ where L is the maximum harmonic used. Fortunately, this neglects the sparsity of the matrix $C^J_{\ell\ell'}$. Specifically, $C^J_{\ell\ell'}$ is a tensor of size $(2J+1) \times (2\ell+1) \times (2\ell'+1)$ with most elements equal to zero. To see this, let $|\ell m\rangle$ be the basis for the ℓ representation. Then, the Clebsch-Gordon coefficients $C^{JM}_{\ell m \ell' m'}$ satisfy

$$|JM\rangle = \sum_{mm'} C^{JM}_{\ell m \ell' m'} |jm\rangle \otimes |j'm'\rangle$$

because J^2 and J_z can be simultaneously diagonalized, it is always possible to find a basis (e.g., the z-basis is standard convention in physics) where $J_z|jm\rangle = m|jm\rangle$ applying this relation to the definition of the Clebsch-Gordon coefficients, we have that

$$J_z|JM\rangle = M|JM\rangle \implies M|JM\rangle = \sum_{mm'} C^{JM}_{\ell m \ell' m'}(m+m')|\ell m\rangle \otimes |\ell' m'\rangle$$

Ergo, the matrix elements $C^{JM}_{\ell m \ell' m'}$ are non-zero only when $M = m + m'$. In physicists language, the sparsity of $C^{JM}_{\ell m \ell' m'}$ is a *selection rule* that follows from conservation of the z-component of angular momentum. Thus, the total number of non-zero elements in $C^J_{\ell\ell'}$ is $(2\ell+1)(2\ell'+1)$—much smaller than the naive estimate of $(2J+1) \times (2\ell+1) \times (2\ell'+1)$. Exploiting this fact, we can compute the tensor product in time dependent only on the input sizes. Parity symmetry further requires that any $C^{JM}_{\ell m, \ell' m'}$ where $\ell + \ell' + J$ is odd is identically zero; it also constrains some elements to be redundant. We compare the sparse method with the Gaunt tensor product method of Luo et al. [14] and the e3nn implementation [33] in section E.

D. Special Case: Vector Long Convolution

We show that our method recovers the vector long convolution method proposed in [9] as a special case. This vector long convolution is a fast $O(N \log N)$ attention mechanism for queries, keys and values of type 1. Let q_i and k_i be

type 1 queries and keys. The vector convolution from [9] is defined as

$$(q \star k)_i = \sum_j q_j \times k_{i-j} \tag{6}$$

where $\times$ denotes the standard vector cross product. Elementwise, the vector convolution has $(q \star k)_{ia} = \epsilon_{abc} \sum_j q_{jb} k_{(i-j)c}$. The vector convolution $q \star k$ transforms in the vector representation of $SO(3)$. Equation (6) is equal to the expression $(q \star k)_i = C_{11}^1 \sum_j q_j \otimes k_{(i-j)}$. To see, this note that the tensor C_{11}^1, which is of dimension $(3 \times 3 \times 3)$ can be written out as a (3×9) matrix with elements given by

$$C_{11}^1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note the for two 3-vectors q and k, this can be written as $C_{11}^1(q \otimes k) = \frac{1}{\sqrt{2}}(q \times k)$. Thus, for any three vectors $C_{11}^1(q \otimes k) = \frac{1}{\sqrt{2}}(q \times k)$ is proportional to the standard cross product. Thus, we have that

$$(q \star k)_{ia} = \epsilon_{abc} \sum_j q_{jb} k_{(i-j)c} = \sqrt{2} C_{11}^1 \sum_j q_j \otimes k_{(i-j)}$$

which reduces to our method for input types $(1, 1)$ and output type 1. Thus, the method proposed in [9] can be seen as a special case of our method when tensor product input features are restricted to be pairs of invariant or vector features.

V. EXPERIMENTS

A. Baselines

We compare our method with state of the art baselines for point cloud processing. The SE(3)-transformer [8] is an equivariant attention mechanism based on Tensor Field Networks [7]. Equiformer v2 [23] uses a convolutional trick from Passaro and Zitnick [13] to reduce the harmonic complexity to $O(L^3)$ from $O(L^6)$. Fused SE(3)-transformer implements the method of Fuchs et al. [8] using fused kernels for decreased computational overhead. SE(3)-Hyena [9] uses a modification of the Hyena architecture to do global linear time attention on invariant (type 0) and vector (type 1) features. The use of only invariant and vector features significantly limits model expressivity.

We benchmark our method on the ModelNet40 [34] classification task, Nbody trajectory simulation [8], QM9 [35], and a custom robotic grasping dataset. See section A for all experiments.

B. Nbody Simulation

Model	**Linear**	**Ours**	**Ours w/o local**	**SE(3)-Hyena**	**Set Trans.**	**SE(3)-Trans.**	**SEGNN**	**EGNN**	**TFN**
MSE x	0.0691	**0.0041**±0.0003	0.0050±0.0003	0.0071 (0.0018)	0.0139	0.0076 (0.0076)	0.0048 (0.0056)	0.0070	0.0150
ΔEQ	—	0.0000096	0.000010	0.00011	0.167	0.00000032	NR	NR	NR
MSE v	0.261	**0.0065**±0.0002	0.0075±0.0003	0.0071±0.0007	0.101	0.075 (0.075)	NR	NR	NR
ΔEQ	—	0.00000048	0.00000052	0.0000012	0.370	0.00000063	NR	NR	NR

TABLE II: N-body simulation results ($N = 5$ particles). Mean squared error for position (x) and velocity (v) prediction. Literature values in parentheses. NR = not reported.

We first test on method on the Nbody simulation task, described in Fuchs et al. [8]. In the simulation in Fuchs et al. [8], five particles each carry either a positive or a negative charge and exert repulsive or attractive forces on each other. The network input is the position of a particle in a specific time step, its velocity, and its charge. The algorithm must predict the relative location and velocity 500 time steps into the future.

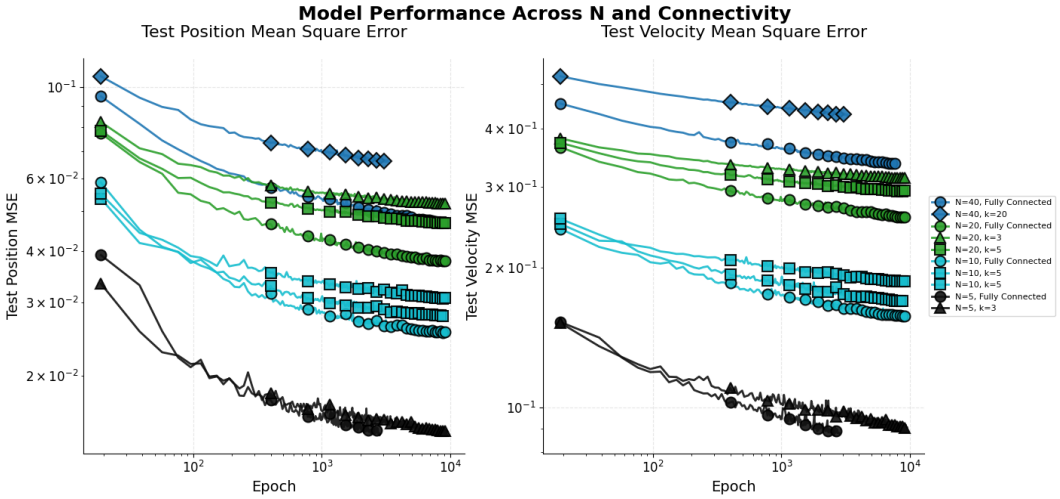

FIG. 3: Performance of $SE(3)$-transformer on the Nbody dataset for different number of points N and nearest neighbors k. We use the exact model parameters at that of Fuchs et al. [8]. Note that on the $N = 20$ curves decreasing $k = 20$ (fully connected graph) to $k = 5$ leads to over a twenty percent drop in performance. Left shows MSE of position, right shows MSE of velocity. Local context window based methods fail to capture accuracy as tasks get more difficult; global context is needed.

For this test, our model consists of a equivariant graph convolution [21], followed by 4 equivariant Clebsch-Gordan attention layers. We use irriducibles types up to six, each with channel dimension of 8 and head dimension of four. Models were trained for 500 epochs, using cosine annealing scheduler with an initial learning rate of $1e^{-3}$ and gradient clipping. Each run was run on a single NVIDA V100 GPU.

As motivation, we show that the model performance can vary with the size of the message passing window. Specifically, in the nbody simulations of Fuchs et al. [8], Satorras et al. [11], Brandstetter et al. [21] an all to all connection is used. This will scale quadratically in the number of particles, which quickly becomes computationally untractable. We 3 perform an ablation study on Fuchs et al. [8] and Brandstetter et al. [21] where we use a k-nearest neighbors, as opposed to fully connected graphs.

Method	**N = 5**		**N = 10**			**N = 20**				**N = 40**				
	k=3	fc	k=3	k=5	fc	k=3	k=5	k=10	fc	k=3	k=5	k=10	k=20	fc
SE(3)-Transformer	0.013	0.013	0.031	0.028	0.025	0.057	0.052	0.050	0.044	0.061	0.056	0.052	0.049	OOM
SEGNN	0.040	0.048	0.023	0.018	0.013	0.042	0.039	0.033	0.029	0.052	0.480	0.450	0.038	OOM
Ours w/o local		**0.003**			0.012				0.026					0.031
Ours		**0.003**			**0.010**				**0.023**					**0.030**

TABLE III: Performance comparison for varying numbers of particles N and nearest neighbors k (fc = fully connected graph). Our method uses fixed $k = 3$ local attention. Averaged over 2 seeds.

As is shown in table III, our method achieves state of the art performance on this task. Furthermore, our model is highly scalable to larger N. Although this is a toy task, we believe these results illustrates both the scalability of our method and the danger of using methods with only local context. Additional experiments and ablations are in section A.

Model	Mean Absolute Error					
	α (Bohr^3)	$\Delta\epsilon$ (meV)	ϵ_{HOMO} (meV)	ϵ_{LUMO} (meV)	μ (D)	C_v (cal/mol·K)
SE(3)-Transformer	0.142	53.0	35.0	33.0	0.510	0.054
EGNN	0.071	48.0	29.0	25.0	0.290	0.031
SEGNN	0.060	42.0	24.0	21.0	0.023	0.310
Ours w/o local	0.100	49.0	31.0	26.0	0.310	0.350
Ours	0.100	**39.0**	**26.0**	**19.0**	**0.210**	**0.030**

TABLE IV: Mean absolute error (MAE) on QM9 dataset. Lower values indicate better performance.

C. QM9

For molecular chemistry, we benchmark on the QM9 dataset [35], a widely-used collection of 134k small organic molecules with up to 9 heavy atoms (C, O, N, F). Each molecule is annotated with 19 regression targets, including atomization energies, dipole moments, and HOMO-LUMO gaps, calculated using DFT. Following standard practice, we predict one target at a time using the provided 110k/10k/10k training/validation/test split, and report mean absolute error (MAE) in units consistent with prior work.

In its current form, our model does not use edge features. For that reason, we first apply an SEGNN layer [21] with skip connection, followed by four of our attention blocks. Output invariant features are then fed into an MLP for classification. For this test, our model consists of an equivariant graph convolution [21], followed by 8 equivariant Clebsch-Gordan attention layers. We use irreducibles up to $\ell = 6$, each with 8 channels and 4 heads. Models were trained for 500 epochs, using cosine annealing scheduler with an initial learning rate of $1e^{-4}$ and gradient clipping. Each run was run on eight NVIDA V100 GPUs. Additional experiments and ablations are shown in section A.

D. ModelNet40 Classification

The ModelNet40 classification task is a widely used benchmark for evaluating 3D shape recognition methods. Introduced by Wu et al. [34], the ModelNet40 dataset consists of $12,311$ 3D CAD models from 40 object categories. The dataset is split into $9,843$ training examples and $2,468$ test examples. The task is to classify 3D objects based on geometric structure. We compare our model with DGCNN [36] and PointNet++ [37] which are state of the art non-equivariant methods.

Model	**Year**	**MN10 Acc.(%)**	**MN40 Acc.(%)**
DGCNN	2019	95.1	92.9
PointNet++	2020	97.4	91.3
SEGNN	2021	94.2	90.5
SE(3)-Trans.	2020	93.2	88.1
Ours w/o local	2025	95.5	89.3
Ours	2025	90.1	85.9

TABLE V: Classification accuracy on ModelNet10 and ModelNet40. All models trained with scale, rotation, and permutation augmentation.

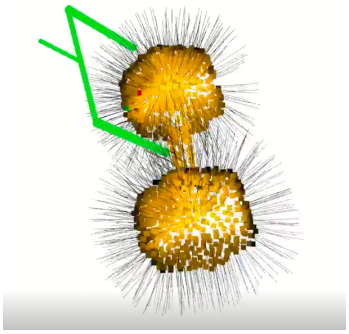

FIG. 4: Robotic object dataset example showing an object with surface normals and optimal grasp locations.

E. Object Grasping Dataset

We also consider a bespoke robotic grasping dataset. Robotic grasping fundamentally depends on object geometry, and high precision robotic grasping is difficult because it requires both fine angular resolution and large context window. Our dataset consists of 400 samples, each of which consists of a point cloud, a set of surface normal vectors,

an optimal grasp orientation (represented as a 3×3 matrix), a optimal grasp depth (which is a single positive real number) and an optimal grasp location. Each point cloud has resolutions of 512, 1024, 2048, or 4096 points. We consider three tasks for the object grasping dataset; namely, surface normal prediction and grasp prediction. We detail these tasks in B.

Model	**Year**	**Rotation Error**				**Distance Error**				**Depth Error**				**Normal Error**			
		512	1024	2048	4096	512	1024	2048	4096	512	1024	2048	4096	512	1024	2048	4096
DGCNN	2018	0.015	0.019	0.031	0.120	3.01	5.34	8.34	10.30	0.08	0.08	0.09	0.08	0.013	0.017	0.023	0.051
PointNet++	2017	0.015	0.021	0.028	0.080	2.51	4.88	5.57	9.54	0.08	0.08	0.07	0.09	0.013	0.018	0.021	0.048
SEGNN	2021	0.018	0.024	0.031	0.100	4.02	5.26	8.37	10.54	0.08	0.09	0.08	0.09	0.015	0.023	0.025	0.052
SE(3)-Trans.	2020	0.025	0.028	OOM	OOM	5.03	7.91	OOM	OOM	0.08	0.09	OOM	OOM	0.025	0.035	OOM	OOM
Ours w/o local	2025	0.019	0.025	0.030	0.090	4.02	5.10	8.30	9.85	0.08	0.08	0.08	0.08	0.013	0.017	0.020	0.041
Ours	2025	**0.013**	**0.017**	0.025	**0.080**	**2.44**	**3.51**	**5.31**	**9.39**	0.08	0.08	**0.07**	0.08	**0.011**	**0.015**	**0.019**	**0.039**

TABLE VI: Performance comparison on robotic grasping dataset across different point cloud resolutions. OOM = Out of Memory. All models trained with rotation, permutation, and scaling augmentation.

We benchmarked each of the baselines methods using the same model parameters as model net classification. Harmonics and nearest neighbors were chosen to be the max amount that fit on memory. Each training run was done on 8 NVIDIA v100 GPUs for 500 epochs.

VI. CONCLUSION

Conclusions. This work tackles two key challenges in equivariant transformers: achieving scalability to global geometric context and efficiently computing high-order irreducible representations for greater expressiveness. By extending vector long convolution to Clebsch–Gordon convolution, we propose the first architecture that achieves global token attention in $\mathcal{O}(N \log N)$ time and supports arbitrary orders of irreducible representations. In addition, our method allows for permutation equivariance, which is essential for point cloud and atomic systems. We also provide comprehensive theoretical analyses to prove both equivariance and time complexity. Finally, we benchmark our method on various dataset, demonstrating clear gains in memory, speed, and accuracy across robotics, physics, and chemistry, outperforming existing state-of-the-art equivariant transformers.

Limitations. The $\mathcal{O}(L^3)$ complexity of our method is still suboptimal and may be prohibitive for cases requiring very high angular resolution.

Future work. Equivariant transformers can draw significant inspiration from computational astrophysics methods designed to efficiently handle N-body interactions.

[1] X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, *Point transformer v3: Simpler, faster, stronger* (2024), 2312.10035, URL `https://arxiv.org/abs/2312.10035`.

[2] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, in *Conference on Robot Learning* (PMLR, 2023), pp. 694–710.

[3] X. Pan, Z. Xia, S. Song, L. E. Li, and G. Huang, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 7463–7472.

[4] C. Zeni, R. Pinsler, D. Zügner, A. Fowler, M. Horton, X. Fu, S. Shysheya, J. Crabbé, L. Sun, J. Smith, et al., *Mattergen: a generative model for inorganic materials design* (2024), 2312.03687, URL `https://arxiv.org/abs/2312.03687`.

[5] B. Rhodes, S. Vandenhaute, V. Šimkus, J. Gin, J. Godwin, T. Duignan, and M. Neumann, *Orb-v3: atomistic simulation at scale* (2025), 2504.06231, URL `https://arxiv.org/abs/2504.06231`.

[6] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, R. Bates, A. Zídek, A. Potapenko, P. Bork, et al., Nature **596**, 583 (2021).

[7] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley, *Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds* (2018), 1802.08219, URL `https://arxiv.org/abs/1802.08219`.

[8] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling, *Se(3)-transformers: 3d roto-translation equivariant attention networks* (2020), 2006.10503, URL `https://arxiv.org/abs/2006.10503`.

[9] A. Moskalev, M. Prakash, R. Liao, and T. Mansi, *Se(3)-hyena operator for scalable equivariant learning* (2024), 2407.01049, URL https://arxiv.org/abs/2407.01049.

[10] Y.-L. Liao and T. Smidt, in *The Eleventh International Conference on Learning Representations* (????).

[11] V. G. Satorras, E. Hoogeboom, and M. Welling, *E(n) equivariant graph neural networks* (2022), 2102.09844, URL https://arxiv.org/abs/2102.09844.

[12] R. Kondor, Z. Lin, and S. Trivedi, *Clebsch-gordan nets: a fully fourier space spherical convolutional neural network* (2018), 1806.09231, URL https://arxiv.org/abs/1806.09231.

[13] S. Passaro and C. L. Zitnick, *Reducing so(3) convolutions to so(2) for efficient equivariant gnns* (2023), 2302.03655, URL https://arxiv.org/abs/2302.03655.

[14] S. Luo, T. Chen, and A. S. Krishnapriyan, *Enabling efficient equivariant operations in the fourier basis via gaunt tensor products* (2024), 2401.10216, URL https://arxiv.org/abs/2401.10216.

[15] D. W. Romero, A. Kuzina, E. J. Bekkers, J. M. Tomczak, and M. Hoogendoorn, arXiv preprint arXiv:2102.02611 (2021).

[16] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon, and C. Ré, *Hyena hierarchy: Towards larger convolutional language models* (2023), 2302.10866, URL https://arxiv.org/abs/2302.10866.

[17] T. Cohen and M. Welling, in *International conference on machine learning* (PMLR, 2016), pp. 2990–2999.

[18] G. Cesa, L. Lang, and M. Weiler, in *International conference on learning representations* (2022).

[19] H. Chen, S. Liu, W. Chen, H. Li, and R. Hill, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 14514–14523.

[20] M. Zhu, M. Ghaffari, W. A. Clark, and H. Peng, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 1223–1232.

[21] J. Brandstetter, R. Hesselink, E. van der Pol, E. J. Bekkers, and M. Welling, *Geometric and physical quantities improve e(3) equivariant message passing* (2022), 2110.02905, URL https://arxiv.org/abs/2110.02905.

[22] S. Passaro and C. L. Zitnick, in *International conference on machine learning* (PMLR, 2023), pp. 27420–27438.

[23] Y.-L. Liao, B. Wood, A. Das, and T. Smidt, *Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations* (2024), 2306.12059, URL https://arxiv.org/abs/2306.12059.

[24] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas, in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 12200–12209.

[25] J. Brehmer, P. de Haan, S. Behrends, and T. Cohen, *Geometric algebra transformer* (2023), 2305.18415, URL https://arxiv.org/abs/2305.18415.

[26] N. Kitaev, L. Kaiser, and A. Levskaya, in *International Conference on Learning Representations (ICLR)* (2020), URL https://arxiv.org/abs/2001.04451.

[27] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, et al., in *International Conference on Learning Representations (ICLR)* (2021), URL https://arxiv.org/abs/2009.14794.

[28] Y. Xiong, Z. Zeng, R. Chakraborty, M. Tan, G. Fung, Y. Li, and V. Singh, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2021), vol. 35, pp. 14138–14148, URL https://arxiv.org/abs/2102.03902.

[29] H. Guo, J. Yang, J. Liu, J. Bai, B. Wang, Z. Li, T. Zheng, B. Zhang, J. Peng, and Q. Tian, in *Proceedings of the AAAI Conference on Artificial Intelligence* (2024), vol. 38, pp. 135–143, URL https://arxiv.org/abs/2401.04749.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention is all you need* (2023), 1706.03762, URL https://arxiv.org/abs/1706.03762.

[31] K. Stachenfeld, J. Godwin, and P. Battaglia, *Graph networks with spectral message passing* (2020), 2101.00079, URL https://arxiv.org/abs/2101.00079.

[32] D. Han, Z. Wang, Z. Xia, Y. Han, Y. Pu, C. Ge, J. Song, S. Song, B. Zheng, and G. Huang, *Demystify mamba in vision: A linear attention perspective* (2024), 2405.16605, URL https://arxiv.org/abs/2405.16605.

[33] M. Geiger and T. Smidt, *e3nn: Euclidean neural networks* (2022), 2207.09453, URL https://arxiv.org/abs/2207.09453.

[34] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1912–1920.

[35] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, The Journal of Chemical Physics **140**, 134101 (2014).

[36] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, *Dynamic graph cnn for learning on point clouds* (2019), 1801.07829, URL https://arxiv.org/abs/1801.07829.

[37] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space* (2017), 1706.02413, URL https://arxiv.org/abs/1706.02413.

[38] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, *Set transformer: A framework for attention-based permutation-invariant neural networks* (2019), 1810.00825, URL https://arxiv.org/abs/1810.00825.

[39] Y. Xie, A. Daigavane, M. Kotak, and T. Smidt, in *ICML 2024 Workshop on Geometric Representations and Modeling (GRaM)* (2024), extended abstract, URL https://openreview.net/forum?id=0HHidbjwcf.

[40] J. Driscoll and D. Healy, Advances in Applied Mathematics **15** (1994).

[41] R. Suda and M. Takami, Math. Comput. **71**, 703 (2002), URL https://api.semanticscholar.org/CorpusID:15275128.

[42] K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, The Astrophysical Journal **622**, 759–771 (2005), ISSN 1538-4357, URL http://dx.doi.org/10.1086/427976.

[43] J. D. McEwen and Y. Wiaux, IEEE Trans. Sig. Proc. **59**, 5876 (2011), arXiv:1110.6298.

[44] M. A. Price and J. D. McEwen, Journal of Computational Physics **510**, 113109 (2024), ISSN 0021-9991, URL http://dx.doi.org/10.1016/j.jcp.2024.113109.

[45] O. J. Cobb, C. G. R. Wallis, A. N. Mavor-Parker, A. Marignier, M. A. Price, M. d'Avezac, and J. D. McEwen, *Efficient generalized spherical cnns* (2021), 2010.11661, URL https://arxiv.org/abs/2010.11661.

[46] G. Cesa, L. Lang, and M. Weiler, in *International Conference on Learning Representations (ICLR)* (2022), URL https://openreview.net/forum?id=WE4qe9xlnQw.

[47] D. Ruhe, J. Brandstetter, and P. Forré, *Clifford group equivariant neural networks* (2023), 2305.11141, URL https://arxiv.org/abs/2305.11141.

Appendix A: Additional Experiments

The goal of this paper is to develop a model architecture that is optimal in harmonic and number of tokens and can learn good representations across a variety of tasks. We summarize our benchmarking results in VII.

TABLE VII: Summary of benchmarking results across datasets. Black checkmarks = our reproduced results, blue checkmarks = results from original papers.

Method	**Year**	**QM9**	**ModelNet40 Classification**	**Robotic Grasp**	**N-body**
Equivariant Methods					
SEGNN	2021	✓	✓	✓	✓✓
SE(3)-Transformer	2020	✓✓	✓	✓	✓✓
SE(3)-Hyena*	2025	—	✓	✓	✓✓
Equiformer-V2	2023	✓	—	✓	✓
GATr	2025	✓	—	—	✓✓
Non-Equivariant Methods					
DGCNN	2018	—	✓✓	✓	✓✓
PointNet++	2017	—	✓✓	—	—

*Our implementation

Appendix B: Object Grasping Dataset

As mentioned in the main text, we benchmark on a custom robotic grasping dataset. Robotic grasping fundamentally depends on object geometry, and high precision robotic grasping is a difficult problem because it requires both fine angular resolution and large context window. Our dataset consists of 400 samples, each of which consists of a point cloud, a set of surface normal vectors, an optimal grasp orientation (represented as a 3×3 matrix), a optimal grasp depth (which is a single positive real number) and an optimal grasp location (i.e. the index of one point in the point cloud). Each group of 100 samples has point cloud resolutions of 512, 1024, 2048, and 4096 points, respectively. Our dataset will be made publicly available.

.

a. Surface Normal Prediction

In the first task, we the model is given the point cloud and must predict the surface normals. This task has inherent $SO(3)$-symmetry as the normal vectors are rotated if the point cloud is rotated. We use a $80-10$ split for train and test data. Each model is given N input vectors (type 1 features) and must output N vectors (type 1) features. We benchmarked each of the baselines methods using the same model parameters as model net classification. Harmonics and nearest neighbors were chosen to be the max amount that fit on memory. Each training run was done on 8 NVIDIA v100 GPUs. Each model was trained for 500 epochs.

b. Grasping

To better simulate real-world settings, we begin by sampling an object and dropping it onto a table. We then randomly select n camera views to capture depth images. These depth images are projected into 3D space using the corresponding camera matrices to generate point clouds. We segment out the table from each raw point cloud and apply the Farthest Point Sampling (FPS) algorithm to downsample the remaining points to the desired resolution, ensuring no duplicate points. To obtain the optimal grasp pose, we first sample an approach point, an orientation, and a grasp depth. We then execute the grasp and evaluate its success by checking whether the object can be lifted and remains secure during a sequence of gripper shaking motions.

c. Grasp Prediction

In the grasp prediction task, the model is given the point cloud and the surface normals and must predict a distribution for optimal grasp location, an optimal grasp orientation and an optimal grasp depth. It should be noted that many of these objects have symmetries, and can have multiple optimal grasp locations. We define the weighted distance error as

$$D(p_{true}, p_{model}) = \int dr dr' \; p_{true}(r) d(r, r') p_{model}(r')$$

which measure the optimal transport distance between the true distribution p_{true} and the model output p_{model}. Grasping is an inherently difficult task as finding the optimal grasp location requires processing information about the global properties of the the object. For this reason, we expect that local attention methods will have difficulty with this task.

Appendix C: Adding Permutation Equivariance

One limitation of Moskalev et al. [9] is that the Fourier decomposition breaks permutation equivariance. Specifically, the introduction of the choice of ordering to compute the Fourier basis breaks that natural permutation equivariance of points. We say that a transformer is permutation equivariant if under the input transformation $u_i \to u_{\sigma(i)}$ the queries, keys and values are transformed via the permutation

$$q_i \to q_{\sigma(i)}, \quad k_i \to k_{\sigma(i)}, \quad v_i \to v_{\sigma(i)}$$

so that the attention matrix transforms as

$$\alpha_{ij} \to \alpha_{\sigma(i)\sigma(j)}$$

Transformers without positional embeddings are naturally permutation invariant, and permuting inputs results in a different permutation of outputs. Methods like deep set transformer [38] enforce permutation invariance by adding additional weight tying constraints. The method proposed in [9] is not permutation invariant as it relies on Hyena attention [16], which is itself not permutation equivariant. Transformer permutation equivariance is key for point cloud operations and simulation of atomic systems (which are, by the uncertainty principle, indistinguishable). Models which are not hardcoded to be permutation invariant will not be able to automatically generalize to permutations of the input graph. We tested numerous methods for enforcing permutation equivariance, discussed in section E. In general, we found that data augmentation with regularization or Fourier space attention was optimal. We describe the Fourier space attention in this section, and two other methods, along with ablation study, are discussed in section E.

a. Fourier Space Attention

The Fourier transform on graphs extends classical Fourier analysis to signals defined on the vertices of a graph. Let $G = (V, E)$ be an undirected graph with n nodes, and let $L \in \mathbb{R}^{n \times n}$ denote its (combinatorial) normalized graph Laplacian, defined as $L = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$, where A is the adjacency matrix, D is the degree matrix and I is the identity matrix. The graph Laplacian L is symmetric and positive semi-definite, it admits an eigendecomposition $L = U \Lambda U^\top$ where $U = [u_1, \ldots, u_n]$ is an orthonormal matrix of eigenvectors and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is the diagonal matrix of positive eigenvalues. The eigenvectors of L form the basis for the graph Fourier transform. For a signal $x \in \mathbb{R}^{n \times d}$ defined on the graph's nodes, its GFT is given by $\hat{x} = U^\top x$ with the inverse GFT is $x = U\hat{x}$. Intuitively, the eigenvectors of the Laplacian serve as generalized "Fourier modes" on the graph, and the eigenvalues correspond to their frequencies. This spectral perspective enables convolution and filtering operations on graphs by manipulating the signal in the frequency domain. Let $x \in \mathbb{R}^{n \times d'}$ be a signal on the graph and $g \in \mathbb{R}^{n \times d \times d'}$ be a filter. The graph convolution in frequency space is defined as:

$$x \star_G g = U \left((U^\top g) \odot (U^\top x) \right) = U(\hat{g} \odot \hat{x})$$

where $\hat{x} = U^\top x$, $\hat{g} = U^\top g$ are the signal and filter Fourier transforms, and $\odot$ denotes element-wise multiplication. This operation corresponds to pointwise multiplication in the spectral domain followed by an inverse transform. The convolution in Fourier space is permutation equivariant. To see this, note that under a permutation of inputs, the

Laplacian matrix transforms as $L \to U_\sigma L U_\sigma^{-1}$ where U_σ denotes the matrix that is a permutation of inputs. Thus, the diagonalization of L transforms as

$$L = U\Lambda U^{-1} \to U_\sigma L U_\sigma^{-1} = U_\sigma[U\Lambda U^{-1}]U_\sigma^{-1} = [U_\sigma U]\Lambda[U^{-1}U_\sigma^{-1}]$$

so the matrix U transforms as $U \to U_\sigma U$. Thus, under rank spectral convolution,

$$x \star_G g = U\left((U^\top g) \odot (U^\top x)\right) \to U_\sigma U\left((U^\top g) \odot (U^\top x)\right) = U_\sigma(x \star_G g)$$

so the spectral convolution $x \star_G g$ is permutation equivariant.

Appendix D: Model Baselines

In this section, we describe the model baselines in more depth. We split models into two categories, equivariant and non-equivariant. For fairness, both model categories were always trained using data augmentation.

1. Non-Equivariant Models

For the non-equivariant models, we use DGCNN [36] and PointNet++ [37].

DGCNN (Dynamic Graph CNN) constructs local neighborhood graphs in each layer and applies convolution-like operations on edges, allowing the model to capture local geometric structures. DGCNN only utilizes local operations, and may have issues modeling long range dependencies.

PointNet++ extends the original PointNet by applying hierarchical feature learning on nested partitions of the input point set, capturing both local and global context in point clouds. PointNet++ specifically uses layers designed to capture information across multiple length scales, which may explain its excellent performance, despite not being equivariant.

2. Equivariant Models

For the equivariant models, we benchmark against Tensor Field Networks [7], SE3-transformer [8], SEGNN [21] and SE3-Hyena [9].

TFN Tensor Field Networks (TFNs) [7] are 3D rotation and translation equivariant networks that associate each point with features transforming under irreducible representations of $SO(3)$. Using spherical harmonics and learnable radial functions, TFNs perform equivariant convolutions over point clouds or molecules.

SE3-Transformer The SE(3)-Transformer [8] builds on the tensor field network for point clouds with SE(3)-equivariance. It uses self-attention and tensor-based operations to model interactions in 3D space, making it effective for tasks like molecular modeling.

SEGNN SEGNN [21] is a message passing neural network that ensures SE(3)-equivariance by representing features with SO(3) irreducibles. SEGNN is designed to model physical quantities that transform under 3D symmetries. Another contribution of [21] was to formalize the connection between message passing methods and transformers.

SE(3)-Hyena SE(3)-Hyena [9] is the most similar method we compare to. It introduces SE(3)-equivariant Hyena operators that combine hierarchical long-range interactions with invariant and vector representations.

3. Tensor Product Ablation

We explain in more depth the tensor product computation.

a. Related Work

The tensor product non-linearity requires significant memory usage. Numerous methods have been developed to reduce the computational complexity of the tensor product. The Gaunt tensor product method [10] changes the Clebsch-Gordan tensor product to the overlap of three spherical harmonics, resulting in an $\mathcal{O}(L^3)$ complexity. However, as pointed out in [39], the Gaunt tensor product is unable to distinguish parity, which makes it unsuitable

for many molecular data processing applications where distinguishing compounds of different chirality is essential (i.e. thalidomide disaster). A different line of work attempts to speed up the tensor product operation using the fast fourier transform on the sphere. The Fourier transform on groups relates $SO(3)$ features to functions on the sphere. The tensor product of $SO(3)$ features is directly related to their product in real space. However, spherical Fourier transforms often suffer from numerical accuracy issues, and this problem is exacerbated on GPU.

TABLE VIII: Comparison of Tensor Product Methods

Method	**Year**	**Scaling**	**GPU Parallelized?**
Naive Method		$\mathcal{O}(L^6)$	yes
Sparse Method		$\mathcal{O}(L^3)$	yes
Gaunt Tensor Product	2024	$\mathcal{O}(L^3)$	yes
Fourier Transform (HEALPix)	2022	$\mathcal{O}(L^3)$	no
Fourier Transform (ssht)	2011	$\mathcal{O}(L^3)$	yes
Fourier Transform (s2fft)	2024	$\mathcal{O}(L^3)$	yes
Fourier Transform [40]	1995	$\mathcal{O}(L^2 \log^2(L))$	no
Fourier Transform [41]	2002	$\mathcal{O}(L^2 \log L)$	no

TABLE IX: Summary of Tensor product methods. The methods based on Fourier transformation have some issues with numerical accuracy and large constant memory cost, and may be suboptimal.

b. Problem Statement

Let $\{f^\ell\}_{\ell=0}^L$ and $\{g^\ell\}_{\ell=0}^L$ be a set of input features band-limited (of degree ℓ) at L. We wish to compute the full tensor product

$$h^J = \sum_\ell C_{\ell\ell'}^J f^\ell \otimes g^{\ell'}$$

for all J in the range $0, 1, 2, ..., L_{out}$, where L_{out} is the maximum output harmonic. Features have additional head and channel dimensions. Separate heads do not interact. If two input features have channel dimension m and m', the tensor product of these features has channel dimension $m \cdot m'$. Now, using the tensor product decomposition

$$\ell \otimes \ell' = \bigoplus_{k=|\ell-\ell'|}^{\ell+\ell'} k$$

the naive complexity of computing this tensor product is given by computing every term in the expansion of h^J. The total computational complexity of computing the single term $C_{\ell\ell}^J f^\ell \otimes g^{\ell'}$ involves contraction of a tensor of size $(2J+1)(2\ell+1)(2\ell'+1)$ and tensors of size $(2\ell+1)$ and $(2\ell'+1)$ for a total complexity of $\mathcal{O}((2J+1)(2\ell+1)(2\ell'+1))$. Thus, the total complexity of computing each term in the tensor product is

$$h^0 : \sum_{ij=1,|i-j|=1}^{L} O(1ij) = \mathcal{O}(L^3)$$

$$h^1 : \sum_{ij=1,|i-j|=2}^{L} O(3ij) = \mathcal{O}(L^3)$$

$$\vdots$$

$$h^L : \sum_{ij=1,|i-j|=L}^{L} O((2L+1)ij) = \mathcal{O}(L^4)$$

Thus, the total computational complexity of computing all the resultant terms h^J is $O(L^6)$.

c. *Sparsity*

However, the fact that the matrix $C^J_{\ell\ell'}$ is sparse can be used to reduce the computational overhead. Specifically, in the real basis, harmonics are zero only when $M = \pm m + \pm' m'$. Thus, the matrix multiplication

$$h^J = C^J_{\ell\ell} f_\ell \otimes g_{\ell'}$$

can be performed by computing the non-zero elements of $C^J_{\ell\ell'}$. There are $(2\ell+1)(2\ell'+1)$ such non-zero matrix entries. Thus, the computational complexity in computing the full tensor product is

$$\begin{aligned} h &: \sum_{ij=1,|i-j|=1}^{L} O(ij) = \mathcal{O}(L^3) \\ h^1 &: \sum_{ij=1,|i-j|=2}^{L} O(ij) = \mathcal{O}(L^3) \\ &\vdots \\ h^L &: \sum_{ij=1,|i-j|=L}^{L} O(ij) = \mathcal{O}(L^3) \end{aligned}$$

so that the total computational complexity of computing each h^J is $O(L^5)$.

d. *Fourier Transform Method*

The Fourier coefficients specify a function on the sphere. Specifically, any function $F : S^2 \to \mathbb{C}$ can be written as

$$F(\hat{n}) = \sum_{\ell=0}^{\infty} (F^\ell)^T Y^\ell(\hat{n})$$

where the Fourier transform coefficients are given by

$$F^\ell = \int_{\hat{n}\in S^2} d\hat{n}\ \ F(\hat{n}) Y^\ell(\hat{n})$$

The tensor product of the Fourier coefficients corresponds to multiplication in Fourier space. Specifically, let f and g be functions on the sphere with Fourier expansions

$$f(\hat{n}) = \sum_{\ell=0}^{\infty} (f^\ell)^T Y^\ell(\hat{n}), \quad g(\hat{n}) = \sum_{\ell=0}^{\infty} (g^\ell)^T Y^\ell(\hat{n})$$

then, consider the Fourier expansion of the product $fg = f \cdot g$ given by

$$(fg)(\hat{n}) = f(\hat{n}) g(\hat{n}) = \sum [(fg)^\ell]^T Y^\ell(\hat{n})$$

Using the inverse Fourier transform in the sphere

$$(fg)^\ell = \int_{\hat{n}\in S^2} d\hat{n}\ \ (fg)(\hat{n}) Y^\ell(\hat{n})$$

Now, using the Fourier expansions of f and g, we have that

$$(fg)(\hat{n}) = \left(\sum_{\ell_1=0}^{\infty} [f^{\ell_1}]^T Y^{\ell_1}(\hat{n}) \right) \left(\sum_{\ell_2=0}^{\infty} [g^{\ell_2}]^T Y^{\ell_2}(\hat{n}) \right) = \sum_{\ell_1,\ell_2=0}^{\infty} [f^{\ell_1} \otimes g^{\ell_2}]^T [Y^{\ell_1}(\hat{n}) \otimes Y^{\ell_2}(\hat{n})]$$

The tensor product of two spherical harmonics decomposes as

$$Y^{\ell_1}(\hat{n}) \otimes Y^{\ell_2}(\hat{n}) = \sum_{\ell=|\ell_1-\ell_2|}^{\ell_1+\ell_2} C^{\ell}_{\ell_1\ell_2} Y^{\ell}(\hat{n})$$

where $C_{\ell}^{\ell_1,\ell_2}$ are the Clebsch-Gordan coefficients. Thus, we have that

$$(fg)(\hat{n}) = \sum_{\ell_1,\ell_2=0}^{\infty} [f^{\ell_1} \otimes g^{\ell_2}]^T [Y^{\ell_1}(\hat{n}) \otimes Y^{\ell_2}(\hat{n})] = \sum_{\ell_1,\ell_2=0}^{\infty} \sum_{\ell=|\ell_1-\ell_2|}^{\ell_1+\ell_2} [f^{\ell_1} \otimes g^{\ell_2}]^T C^{\ell}_{\ell_1\ell_2} Y^{\ell}(\hat{n})$$

and reindexing this sum,

$$(fg)(\hat{n}) = \sum_{\ell_1,\ell_2=0}^{\infty} \sum_{\ell=|\ell_1-\ell_2|}^{\ell_1+\ell_2} [f^{\ell_1} \otimes g^{\ell_2}]^T C^{\ell}_{\ell_1\ell_2} Y^{\ell}(\hat{n}) = \sum_{\ell=0}^{\infty} [\sum_{\ell_1,\ell_2=0}^{\infty} C_{\ell}^{\ell_1\ell_2} (f^{\ell_1} \otimes g^{\ell_2})]^T Y^{\ell}(\hat{n})$$

Therefore, the Fourier coefficients of the product are given by

$$[(fg)^{\ell}]^T = \sum_{\ell_1,\ell_2} C_{\ell}^{\ell_1,\ell_2} [f^{\ell_1} \otimes g^{\ell_2}]^T$$

Thus, one way to evaluate the tensor product is to compute the inverse spherical harmonic transform (isht) of f^{ℓ} and g^{ℓ}, multiply the signals on the sphere, and then spherical Fourier transform (sht). We consider a few methods for computing the sht and isht.

In principal, the computational complexity of computing the Fourier transform can be as low as $\mathcal{O}(L^2 \log L)$. However, for the regime of interest ($L \approx 7$), methods that have $\mathcal{O}(L^3)$ complexity often have lower prefactor and are more memory efficient.

Healpix HEALPix (Hierarchical Equal Area isoLatitude Pixelization) [42] is a standard method for discretizing functions on the sphere. HEALPix sampling divides the sphere into equal-area pixels arranged along iso-latitude rings. HEALPix has built in support for spherical harmonic transforms. The full cost of spherical harmonic transform on HEALpix grid is $\mathcal{O}(L^3)$.

ssht ssht [43] library is a C++ and Python library for computing spherical harmonic transforms. It is used for Fourier analysis on the sphere, particularly in scientific fields like astrophysics and geophysics. ssht [43] uses a novel sampling scheme on the sphere to compute the Fourier transform. The computationally complexity of this method scales as $\mathcal{O}(L^3)$.

s2fft s2fft [44] is a recursive algorithm for the calculation of Wigner d-functions. s2fft is a highly parallelizable method and is implemented in JAX. s2fft supports HEALpix and equiangular sampling. The computationally complexity of this method scales as $\mathcal{O}(L^3)$, but by utilizing GPU parallelization, the complexity decreases almost linearly in number of GPUs used.

[45] also developed interesting methods for spherical convolutions which may be utilized for fast group theoretic Fourier transformations in the future.

4. Tensor Product Ablations

We compared three methods for computing the tensor product, the naisve method, the gaunt tensor product method, the Fourier transform method with healpix, the Fourier transform with s2fft[44] and ssht [43] and the sparse method.

Appendix E: Model Ablations

We consider a set of ablations on our model. We hope these ablations help the reader navigate the design space of this class of models.

TABLE X: Mean absolute error (MAE) on QM9 dataset using different tensor product methods. Lower is better.

Tensor Product Method	α (Bohr^3)	$\Delta\epsilon$ (meV)	ϵ_{HOMO} (meV)	ϵ_{LUMO} (meV)	μ (D)	C_v (cal/mol K)
Sparse Method	0.08	48	30	25	0.29	0.31
Gaunt Tensor Product	0.15	50	33	28	0.33	0.38
Fourier Transform (HEALPix)	0.13	49	31	26	0.31	0.35
Fourier Transform (s2fft)	0.10	49	31	26	0.31	0.35

TABLE XI: Comparison of tensor product methods in terms of MAE on QM9 dataset. Advanced and GPU-parallelized methods are expected to yield better accuracy. The Gaunt tensor product has an issue with parity, as is explained in [39]. The Sparse method has $\mathcal{O}(L^3)$ scaling. All other methods achieve $\mathcal{O}(L^3)$ scaling.

1. Invariant embeddings

Tasks of interest usually consisted of both invariant features and vector features. We found that using an encoder to transform all features into invariant representations improved model performance. We used Laplacian graph eigenvectors as positional features to provide a global structural signal. These eigenvectors were computed from the normalized graph Laplacian and concatenated with the input features before encoding. When additional information, such as charge in the Nbody experiment, was given, these properties were encoded using a learned one hot encoding. Learned output types are encoded through standard neural network, i.e. the encoder is a non-equivariant neural network.

2. Encoding Ablation

We consider two ablations on the structure equivariant MLP between layers. In the main text of the paper, we used layers constructed from e3nn [33], with custom tensor product. We checked that each layer was equivariant using escnn [46]. The original $SE(3)$-Hyena paper used layers constructed from Clifford-MLP[47]. We ablate by testing the use of Clifford-MLP [47]. We found no substantial difference between Clifford-MLP layers and e3nn layers, indicating that the attention mechanism is the key for downstream model performance.

	N=5			**N=10**			**N=20**			**N=40**		
Model	mse_x	mse_v	δEQ	mse_x	mse_v	δEQ	mse_x	mse_v	δEQ	mse_x	mse_v	δEQ
e3nn layers	0.0041	0.0065	0.0000096	0.012	0.019	0.0000091	0.026	0.031	0.000061	0.031	0.049	0.000049
Clifford MLP layers	0.0043	0.0068	0.0019	0.011	0.023	0.000045	0.026	0.033	0.000081	0.035	0.045	0.00042

TABLE XII: Comparison of e3nn layers vs Clifford MLP [47]. Performance comparison across different N values.

3. Multiple Head Clebsch-Gordan Attention

In practice, we found that using multiple heads in the tensor product lead to better performance. Specifically, query and key features q_i and k_i which are inputs to Clebsch-Gordan convolution are set to have h independent heads. This heads do not interact with each other during the Clebsch-Gordan convolution, but are allowed to interact during linear layers and non-linearities. Thus, there are three architecture parameters to consider in a CG convolution, the number of heads h, the maximum harmonic L and the channel dimension m. Query and key were always constrained to have the same number of heads, maximum harmonic and channel dimension. The total output dimension is has memory scaling $O(h \cdot L^2 \cdot m)$. We found that performance was sensitive to the choice of (h, L, m), setting the head dimension h or the channel dimension m too small led to a drastic decrease in performance. We found that increasing the harmonic number L always improved performance, at the cost of increased memory usage.

TABLE XIII: Mean absolute error (MAE) on QM9 dataset for different maximum spherical harmonic degrees L. Lower is better.

Max Harmonic Degree L	α (Bohr^3)	$\Delta\epsilon$ (meV)	ϵ_{HOMO} (meV)	ϵ_{LUMO} (meV)	μ (D)	C_v (cal/mol K)
$L = 3$	0.53	68	71	59	0.93	0.89
$L = 4$	0.25	59	42	35	0.69	0.58
$L = 5$	0.19	51	33	29	0.50	0.39
$L = 6$	0.10	49	31	26	0.31	0.35

TABLE XIV: Model Ablation: Comparison of performance on QM9 as a function of the maximum harmonic L used. Performance monotonically increases with the maximum harmonic L.

4. Gating Ablation

The gating mechanism is the key component in transformers. In our proposed method, we perform another tensor product to combine queries, keys and values. However, it is not obvious if this is needed. As an alternative, it maybe possible to simply concatenate u_i^ℓ with values v_i^ℓ. We found that this leads to slight decrease in performance (although decreases the forward pass time by a sizable margin).

TABLE XV: Mean absolute error (MAE) on QM9 dataset for different gating types. Lower is better

Gating Type	α (Bohr^3)	$\Delta\epsilon$ (meV)	ϵ_{HOMO} (meV)	ϵ_{LUMO} (meV)	μ (D)	C_v (cal/mol K)
Concatenation gating	0.25	61	46	31	0.63	0.51
CG gating	0.10	49	31	26	0.31	0.35

TABLE XVI: Model Ablation: Comparison of performance on QM9 as a function of the maximum harmonic L used. Models were trained with $L = 6$ and four heads.