

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 7.

Table 7: 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
<i>Equivariant Methods</i>					
SEGNN	2021	✓	✓	✓	✓✓
SE(3)-Transformer	2020	✓✓	✓	✓	✓✓
SE(3)-Hyena*	2025	—	✓	✓	✓✓
Equiformer-V2	2023	✓	—	✓	✓✓
GATr	2025	✓	—	—	✓✓
<i>Non-Equivariant Methods</i>					
DGCNN	2018	—	✓✓	✓	✓✓
PointNet++	2017	—	✓✓	—	—

*Our implementation

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.

B.0.1 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.0.2 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.

B.0.3 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.

C ADDING PERMUTATION EQUIVARIANCE

One limitation of [Moskalev et al. \(2024\)](#) 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 $u_i \rightarrow u_{\sigma(i)}$ the queries, keys and values are transformed via the permutation

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

so that the attention matrix transforms as

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

Transformers without positional embeddings are naturally permutation invariant, and permuting inputs results in a different permutation of outputs. Methods like deep set transformer [Lee et al. \(2019\)](#) enforce permutation invariance by adding additional weight tying constraints. The method proposed in [Moskalev et al. \(2024\)](#) is not permutation invariant as it relies on Hyena attention [Poli et al. \(2023\)](#), 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 appendix 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 appendix E.

C.0.1 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 $U = [u_1, \dots, u_n]$ is an orthonormal matrix of eigenvectors and $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ 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 $L \rightarrow U_\sigma L U_\sigma^{-1}$ where U_σ denotes the matrix that is a permutation of inputs. Thus, the diagonalization of L transforms as

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

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

$$x \star_G g = U ((U^\top g) \odot (U^\top x)) \rightarrow U_\sigma U ((U^\top g) \odot (U^\top x)) = U_\sigma (x \star_G g)$$

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

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.

D.1 NON-EQUIVARIANT MODELS

For the non-equivariant models, we use DGCNN Wang et al. (2019) and PointNet++ Qi et al. (2017).

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.

D.2 EQUIVARIANT MODELS

For the equivariant models, we benchmark against Tensor Field Networks Thomas et al. (2018), SE3-transformer Fuchs et al. (2020), SEGNN Brandstetter et al. (2022) and SE3-Hyena Moskalev et al. (2024).

TFN Tensor Field Networks (TFNs) Thomas et al. (2018) 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 Fuchs et al. (2020) 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 Brandstetter et al. (2022) 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 Brandstetter et al. (2022) was to formalize the connection between message passing methods and transformers.

SE(3)-Hyena SE(3)-Hyena Moskalev et al. (2024) 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.

D.3 TENSOR PRODUCT ABLATION

We explain in more depth the tensor product computation.

D.3.1 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 [Liao and Smidt](#) 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 [Xie et al. \(2024\)](#), 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 8: 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 Driscoll and Healy (1994)	1995	$\mathcal{O}(L^2 \log^2(L))$	no
Fourier Transform Suda and Takami (2002)	2002	$\mathcal{O}(L^2 \log L)$	no

Table 9: 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.

D.3.2 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 $0, 1, 2, \dots, L_{out}$, 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

$$\begin{aligned}
 h^0 : \quad & \sum_{ij=1, |i-j|=1}^L O(1ij) = \mathcal{O}(L^3) \\
 h^1 : \quad & \sum_{ij=1, |i-j|=2}^L O(3ij) = \mathcal{O}(L^3) \\
 & \vdots \\
 h^L : \quad & \sum_{ij=1, |i-j|=L}^L O((2L+1)ij) = \mathcal{O}(L^4)
 \end{aligned}$$

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

D.3.3 SPARSITY

However, the fact that the matrix $C_{\ell\ell'}^J$ 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

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

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

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

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

D.3.4 FOURIER TRANSFORM METHOD

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

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

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

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

and reindexing this sum,

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

Therefore, the Fourier coefficients of the product are given by

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

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) [Gorski et al. \(2005\)](#) 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 [McEwen and Wiaux \(2011\)](#) 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 [McEwen and Wiaux \(2011\)](#) uses a novel sampling scheme on the sphere to compute the Fourier transform. The computational complexity of this method scales as $\mathcal{O}(L^3)$.

s2fft s2fft [Price and McEwen \(2024\)](#) 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 computational 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.

D.4 TENSOR PRODUCT ABLATIONS

We compared three methods for computing the tensor product, the naive method, the gaunt tensor product method, the Fourier transform method with healpix, the Fourier transform with s2ft [Price and McEwen \(2024\)](#) and ssht [McEwen and Wiaux \(2011\)](#) and the sparse method.

Table 10: 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 (s2ft)	0.10	49	31	26	0.31	0.35

Table 11: 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 [Xie et al. \(2024\)](#). The Sparse method has $\mathcal{O}(L^3)$ scaling. All other methods achieve $\mathcal{O}(L^3)$ scaling.

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.

E.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.

E.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 [Geiger and Smidt \(2022\)](#), with custom tensor product (see ??). We checked that each layer was equivariant using escnn [Cesa et al. \(2022a\)](#). The original $SE(3)$ -Hyena paper used layers constructed from Clifford-MLP [Ruhe et al. \(2023\)](#). We ablate by testing the use of Clifford-MLP [Ruhe et al. \(2023\)](#). We found no substantial difference between Clifford-MLP layers and e3nn layers, indicating that the attention mechanism is the key for downstream model performance.

Model	N=5			N=10			N=20			N=40		
	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	9.6×10^{-6}	0.012	0.019	9.1×10^{-6}	0.026	0.031	6.1×10^{-5}	0.031	0.049	4.9×10^{-5}
Clifford MLP layers	0.0043	0.0068	1.9×10^{-3}	0.011	0.023	4.5×10^{-5}	0.026	0.033	8.1×10^{-5}	0.035	0.045	4.2×10^{-4}

Table 12: Comparison of e3nn layers vs Clifford MLP [Ruhe et al. \(2023\)](#). Performance comparison across different N values.

E.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. These 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 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 13: 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 14: Model Ablation: Comparison of performance on QM9 as a function of the maximum harmonic L used. Performance monotonically increases with the maximum harmonic L .

E.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 may be 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 15: 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 16: 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.

REFERENCES

- Johannes Brandstetter, Rob Hesselink, Elise van der Pol, Erik J Bekkers, and Max Welling. Geometric and physical quantities improve e(3) equivariant message passing, 2022. URL <https://arxiv.org/abs/2110.02905>.
- Johann Brehmer, Pim de Haan, Sönke Behrends, and Taco Cohen. Geometric algebra transformer, 2023. URL <https://arxiv.org/abs/2305.18415>.
- Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build E(N)-equivariant steerable CNNs. In *International Conference on Learning Representations (ICLR)*, 2022a. URL <https://openreview.net/forum?id=WE4qe9xlnQw>.
- Gabriele Cesa, Leon Lang, and Maurice Weiler. A program to build e(n)-equivariant steerable cnns. In *International conference on learning representations*, 2022b.
- Haiwei Chen, Shichen Liu, Weikai Chen, Hao Li, and Randall Hill. Equivariant point network for 3d point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14514–14523, 2021.
- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations (ICLR)*, 2021. URL <https://arxiv.org/abs/2009.14794>.

- Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.
- Congyue Deng, Or Litany, Yueqi Duan, Adrien Poulénard, Andrea Tagliasacchi, and Leonidas J Guibas. Vector neurons: A general framework for so(3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12200–12209, 2021.
- James Driscoll and Dennis Healy. Computing fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15, 06 1994. doi: 10.1006/aama.1994.1008.
- Fabian B. Fuchs, Daniel E. Worrall, Volker Fischer, and Max Welling. Se(3)-transformers: 3d roto-translation equivariant attention networks, 2020. URL <https://arxiv.org/abs/2006.10503>.
- Mario Geiger and Tess Smidt. e3nn: Euclidean neural networks, 2022. URL <https://arxiv.org/abs/2207.09453>.
- K. M. Gorski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759–771, April 2005. ISSN 1538-4357. doi: 10.1086/427976. URL <http://dx.doi.org/10.1086/427976>.
- Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710. PMLR, 2023.
- Hongcheng Guo, Jian Yang, Jiaheng Liu, Jiaqi Bai, Boyang Wang, Zhoujun Li, Tiejiao Zheng, Bo Zhang, Junran Peng, and Qi Tian. Logformer: A pre-train and tuning pipeline for log anomaly detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 135–143, 2024. URL <https://arxiv.org/abs/2401.04749>.
- Dongchen Han, Ziyi Wang, Zhuofan Xia, Yizeng Han, Yifan Pu, Chunjiang Ge, Jun Song, Shiji Song, Bo Zheng, and Gao Huang. Demystify mamba in vision: A linear attention perspective, 2024. URL <https://arxiv.org/abs/2405.16605>.
- John Jumper, Richard Evans, Alexander Pritzel, Tom Green, Michael Figurnov, Olaf Ronneberger, Robert Bates, Adam Zidek, Anton Potapenko, Peer Bork, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.
- Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations (ICLR)*, 2020. URL <https://arxiv.org/abs/2001.04451>.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks, 2019. URL <https://arxiv.org/abs/1810.00825>.
- Yi-Lun Liao and Tess Smidt. Equiformer: Equivariant graph attention transformer for 3d atomistic graphs. In *The Eleventh International Conference on Learning Representations*.
- Yi-Lun Liao, Brandon Wood, Abhishek Das, and Tess Smidt. Equiformerv2: Improved equivariant transformer for scaling to higher-degree representations, 2024. URL <https://arxiv.org/abs/2306.12059>.
- Shengjie Luo, Tianlang Chen, and Aditi S. Krishnapriyan. Enabling efficient equivariant operations in the fourier basis via gaunt tensor products, 2024. URL <https://arxiv.org/abs/2401.10216>.
- J. D. McEwen and Y. Wiaux. A novel sampling theorem on the sphere. *IEEE Trans. Sig. Proc.*, 59(12):5876–5887, 2011. doi: 10.1109/TSP.2011.2166394.
- Artem Moskalev, Mangal Prakash, Rui Liao, and Tommaso Mansi. Se(3)-hyena operator for scalable equivariant learning, 2024. URL <https://arxiv.org/abs/2407.01049>.

- Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7463–7472, 2021.
- Saro Passaro and C Lawrence Zitnick. Reducing so (3) convolutions to so (2) for efficient equivariant gnns. In *International conference on machine learning*, pages 27420–27438. PMLR, 2023a.
- Saro Passaro and C. Lawrence Zitnick. Reducing so(3) convolutions to so(2) for efficient equivariant gnns, 2023b. URL <https://arxiv.org/abs/2302.03655>.
- Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models, 2023. URL <https://arxiv.org/abs/2302.10866>.
- Matthew A. Price and Jason D. McEwen. Differentiable and accelerated spherical harmonic and wigner transforms. *Journal of Computational Physics*, 510:113109, August 2024. ISSN 0021-9991. doi: 10.1016/j.jcp.2024.113109. URL <http://dx.doi.org/10.1016/j.jcp.2024.113109>.
- Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space, 2017. URL <https://arxiv.org/abs/1706.02413>.
- Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole von Lilienfeld. Quantum chemistry datasets for benchmarking molecular machine learning. *The Journal of Chemical Physics*, 140(13):134101, 2014.
- Benjamin Rhodes, Sander Vandenhaute, Vaidotas Šimkus, James Gin, Jonathan Godwin, Tim Duignan, and Mark Neumann. Orb-v3: atomistic simulation at scale, 2025. URL <https://arxiv.org/abs/2504.06231>.
- David W Romero, Anna Kuzina, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogenboom. Ckconv: Continuous kernel convolution for sequential data. *arXiv preprint arXiv:2102.02611*, 2021.
- David Ruhe, Johannes Brandstetter, and Patrick Forré. Clifford group equivariant neural networks, 2023. URL <https://arxiv.org/abs/2305.11141>.
- Victor Garcia Satorras, Emiel Hoogetboom, and Max Welling. E(n) equivariant graph neural networks, 2022. URL <https://arxiv.org/abs/2102.09844>.
- Kimberly Stachenfeld, Jonathan Godwin, and Peter Battaglia. Graph networks with spectral message passing, 2020. URL <https://arxiv.org/abs/2101.00079>.
- Reiji Suda and Masayasu Takami. A fast spherical harmonics transform algorithm. *Math. Comput.*, 71:703–715, 2002. URL <https://api.semanticscholar.org/CorpusID:15275128>.
- Nathaniel Thomas, Tess 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, 2018. URL <https://arxiv.org/abs/1802.08219>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL <https://arxiv.org/abs/1706.03762>.
- Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2019. URL <https://arxiv.org/abs/1801.07829>.
- Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger, 2024. URL <https://arxiv.org/abs/2312.10035>.

- Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.
- YuQing Xie, Ameeya Daigavane, Mit Kotak, and Tess Smidt. The price of freedom: Exploring tradeoffs between expressivity and computational efficiency in equivariant tensor products. In *ICML 2024 Workshop on Geometric Representations and Modeling (GRaM)*, 2024. URL <https://openreview.net/forum?id=0HHidbjwcf>. Extended abstract.
- Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021. URL <https://arxiv.org/abs/2102.03902>.
- Claudio Zeni, Robert Pinsler, Daniel Zügner, Andrew Fowler, Matthew Horton, Xiang Fu, Sasha Shysheya, Jonathan Crabbé, Lixin Sun, Jake Smith, Bichlien Nguyen, Hannes Schulz, Sarah Lewis, Chin-Wei Huang, Ziheng Lu, Yichi Zhou, Han Yang, Hongxia Hao, Jielan Li, Ryota Tomioka, and Tian Xie. Mattergen: a generative model for inorganic materials design, 2024. URL <https://arxiv.org/abs/2312.03687>.
- Minghan Zhu, Maani Ghaffari, William A Clark, and Huei Peng. E2pn: Efficient se (3)-equivariant point network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1223–1232, 2023.