SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks

Fabian Fuchs et al. NeurIPS 2020

Presenter: Beibei Xiong, Arman Lotfalikhani, Siddharth Rout

Presentation *Roadmap*

- Motivation
- Related Concepts
- Methods
- Experiments and Results
- Takeaway and Discussion

Motivation

Motivation: Learning on 3D Geometric Data

- Many learning tasks involve **point cloud data**
  - 3D object scans, 3D molecular structures, N-body particle simulations…
- The **global coordinate system is arbitrary**
  - 3D rotations and translations of the input point cloud should **not** affect the output
- Standard neural networks are **not symmetry-aware**
- Desired property: **models that respect geometric symmetries by design**

Rotation of the input should not change the meaning of the prediction
-> SE(3) transformer

SE(3) Transformer

Motivation:

Rotation of the input should not change the meaning of the prediction

SE(3)-Transformer = Self-Attention + SE(3) Equivariance + Graph Neural Networks

SE(3) = Special Euclidean Group (3) = 3 Rotations + 3 Translations

Self-Attention Mechanism

What is Self-Attention?

- Self-attention allows each element to **dynamically aggregate information from other elements**
- Each element decides **who to attend to** and **how much**
- Widely used in Transformers for sequences, sets, graphs, and point clouds

Self-attention = *data-driven, content-based weighted aggregation*

How Self-Attention Works?

In a self-attention mechanism, each input element is projected into three representations:

- **Query (Q):** represents what this element is looking for
- **Key (K):** represents what this element offers for matching
- **Value (V):** represents the information content carried by this element

They are computed as linear projections:

$$Q = XW_Q,\ K = XW_K,\ V = XW_V$$

$$\text{Attention}(q_i) = \sum_j \alpha_{ij}\, v_j$$

Query and Key are used to **compute relevance**.

Value is the **content being aggregated**.

$$\alpha_{ij} = \text{softmax}_j\left(q_i^\top k_j\right)$$

Self-attention = Attention weights + Value aggregation

Powerful and Problematic in 3D

Powerful:

- Captures long-range interactions
- Adaptive and input-dependent
- No fixed neighborhood or ordering assumptions (permutation equivariant)

Problematic:

- Standard self-attention does not respect geometric symmetries
- In 3D data, attention weights can change arbitrarily under rotations

  (not rotation-equivariant)

Self-Attention -> SE(3) Transformers

Permutation equivariance → suitable for sets and graphs
SE(3)-equivariance → suitable for 3D geometry

SE(3)-Transformer preserves both:

- permutation equivariance over points
- rotation / translation equivariance over coordinates

SE(3)-Transformer key idea:

combine the **flexibility of self-attention** with **strict 3D equivariance**

- design **rotation-invariant attention weights** and **SE(3)-equivariant value messages**, so that the entire layer remains equivariant

Graph Neural Networks (GNNs)

Graph Neural Networks and Message Passing

- Graph Neural Networks (GNNs) operate on **nodes and edges**
- Node features are updated by **aggregating messages from neighbors**
  - ➢ **reduces complexity from $O(N^2)$ to $O(Nk)$**
- Aggregation uses **permutation-invariant** operations (e.g. sum/mean/max)
- Self-attention can be viewed as **message passing with learned weights**
- Provides a natural framework for extending attention to structured data

Graph Neural Networks and Message Passing

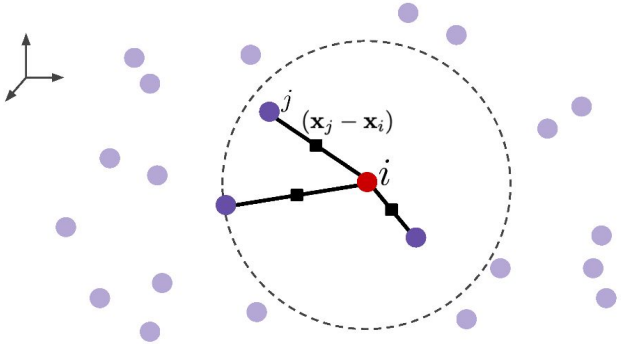
- Self-attention tells us **how to weight messages adaptively**
- GNNs tell us **how to aggregate messages in a permutation-equivariant way**
- TFNs tell us **how to build equivariant messages**

SE(3)-Transformer = TFN-style equivariant messages + attention-based weighting

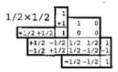
Equivariance + Tensor Fields Network

Implementation Pipeline

Step 1: Get nearest neighbours and relative positions



Step 2: Get SO(3)-equivariant weight matrices



Clebsch-Gordon Coeff.

Radial Neural Network

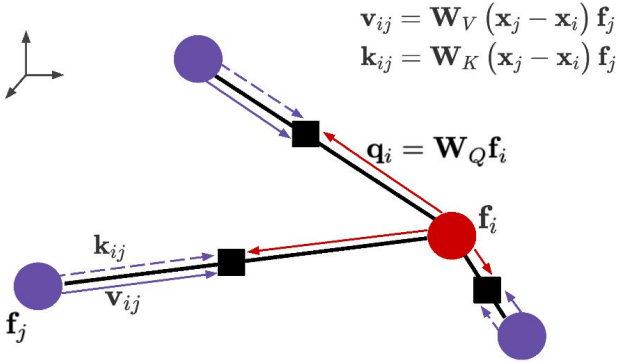
Spherical Harmonics

$$\underbrace{\mathbf{Q}_{Jm}^{\ell k} \quad \varphi_J^{\ell k}(||x||) \quad Y_{Jm}\left(\frac{x}{||x||}\right)}$$

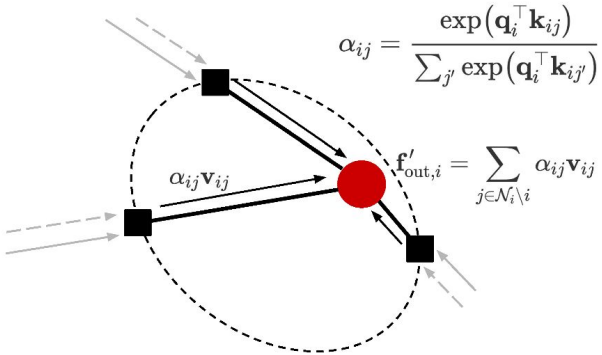
Matrix W consists of blocks mapping between degrees

$$\mathbf{W}(x) = \mathbf{W}\left(\left\{\mathbf{Q}_{Jm}^{\ell k}, \varphi_J^{\ell k}(||x||), Y_{Jm}\left(\frac{x}{||x||}\right)\right\}_{J,m,\ell,k}\right)$$

Step 3: Propagate queries, keys, and values to edges



Step 4: Compute attention and aggregate



Group Theory Background

A group consists of a set $\mathcal{G}$ and a binary operator. The set and the operator satisfy:

1. Closure under product
2. Associativity
3. Existence of Identity element
4. Existence of inverse element

Equivariance definition:

$$S_g[f(\mathbf{x})] = f(T_g[\mathbf{x}]) \qquad \text{for all } g \in G, \mathbf{x} \in \mathcal{X}.$$

Group representation

1. Group representation: Map from Group G to invertible $N \times N$ matrices
2. Homomorphism: $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$
3. Similarity: $\rho'(g) = \mathbf{Q}^{-1}\rho(g)\mathbf{Q}, \quad \text{for all } g \in G.$
4. Reducible and irreducible

$$\rho(g) = \mathbf{Q}^{-1}(\rho_1(g) \oplus \rho_2(g))\mathbf{Q} = \mathbf{Q}^{-1}\begin{bmatrix}\rho_1(g) & \\ & \rho_2(g)\end{bmatrix}\mathbf{Q},$$

Spherical Harmonics and Wigner-D matrices

- The rotation operator on a function can be characterized with the operator

$$R(\alpha,\beta,\gamma) = e^{-i\alpha L_z} e^{-i\beta L_y} e^{-i\gamma L_z}$$

- R commutes with spherical part of the Laplacian (or L^2operator)
- Eigenfunctions of :

$$R(\alpha,\beta,\gamma)Y_{Jm} = \sum_{-m}^{m} \alpha_{m'} Y_{Jm'} \Rightarrow < Y^*_{Jm'} R(\alpha,\beta,\gamma) Y_{Jm} >= D^J_{m,m'}$$

- Spherical harmonics: complete orthonormal basis on a sphere

$$\mathbf{f}(\mathbf{x}) = \sum_{J\geq 0} \sum_{m=-J}^{J} f_{Jm} Y_{Jm}(\mathbf{x}) = \sum_{J\geq 0} \mathbf{f}_\mathbf{J}^T \mathbf{Y}_J(\mathbf{x})$$

Spherical Harmonics and Wigner-D matrices

- Compute action of R to f using the expansion:

$$\mathbf{f}(R^{-1}\mathbf{x}) = \sum_{J\geq 0}\sum_{m=-J}^{J} f_{Jm}Y_{Jm}(R^{-1}\mathbf{x}) = \sum_{J\geq 0}\sum_{m=-J}^{J} f_{Jm}R^{-1}Y_{Jm}(R^{-1}\mathbf{x}) = \sum_{J\geq 0}\mathbf{f}_{\mathbf{J}}^{T}\mathbf{D}_{J}^{*}\mathbf{Y}_{J}(\mathbf{x})$$

- Each f_J is called a type-J feature.

Tensor Field Network Derivation

- A TFN layer computes the convolution of a learnable kernel $\mathbf{W}_{\ell k}$
- Start by taking taking convolution over type-k features:

$$
\begin{aligned}
\mathbf{f}_{\text{out,i}}^{\ell} &= [\mathbf{W}^{\ell k} * \mathbf{f}_{\text{in}}^{k}](\mathbf{x}) \\
&= \int_{\mathbb{R}^3} \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\mathbf{f}_{\text{in}}^{k}(\mathbf{x}')\,\mathrm{d}\mathbf{x}' \\
&= \int_{\mathbb{R}^3} \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\sum_{j=1}^{N}\mathbf{f}_{\text{in},j}^{k}\delta(\mathbf{x}' - \mathbf{x}_j)\,\mathrm{d}\mathbf{x}' \\
&= \sum_{j=1}^{N}\mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}.
\end{aligned}
$$

Tensor Field Network Derivation

- Apply the equivariance condition

$$\mathbf{f}_{\text{out,i}}^{\ell} = \sum_{j=1}^{N} \mathbf{D}_{\ell}(g)^{-1} \mathbf{W}^{\ell k} \left(\mathbf{R}_{g}^{-1} \left(\mathbf{x}_{j} - \mathbf{x}_{i} \right) \right) \mathbf{D}_{k}(g) \mathbf{f}_{\text{in},j}^{k}$$

- Simplified in vector form

$$\text{vec}(\mathbf{AXB}) = \left(\mathbf{B}^{\top} \otimes \mathbf{A} \right) \text{vec}(\mathbf{X})$$

$$\text{vec}(\mathbf{W}^{\ell k}(\mathbf{R}_{g}^{-1}\mathbf{x})) = (\mathbf{D}_{k}(g) \otimes \mathbf{D}_{\ell}(g)) \text{vec}(\mathbf{W}^{\ell k}(\mathbf{x}))$$

Tensor Field Network Derivation

- Clebsh-Gordan Coefficients to reduce the kronecker product:

$$\mathbf{D}_k(g) \otimes \mathbf{D}_\ell(g) = \mathbf{Q}^{\ell k \top} \left[\bigoplus_{J=|k-\ell|}^{k+\ell} \mathbf{D}_J(g) \right] \mathbf{Q}^{\ell k}$$

- Final irreducible from:

$$\mathrm{vec}(\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})) = \mathbf{Q}^{\ell k \top} \left[\bigoplus_{J=|k-\ell|}^{k+\ell} \mathbf{D}_J(g) \right] \mathbf{Q}^{\ell k} \mathrm{vec}(\mathbf{W}^{\ell k}(\mathbf{R}_g^{-1}\mathbf{x})).$$

Tensor Field Network Derivation

- Simplest solution to above equations

$$\text{vec}\left(\mathbf{W}^{\ell k}(\mathbf{x})\right) = \mathbf{Q}^{\ell k\top} \bigoplus_{J=|k-\ell|}^{k+\ell} \mathbf{Y}_J(\mathbf{x}).$$

- However, equivariance does not change by adding radial functions

$$\mathbf{W}^{\ell k}(\mathbf{x}) = \sum_{J=|k-\ell|}^{k+\ell} \varphi_J^{\ell k}(\|\mathbf{x}\|)\mathbf{W}_J^{\ell k}(\mathbf{x}), \qquad \text{where } \mathbf{W}_J^{\ell k}(\mathbf{x}) = \sum_{m=-J}^{J} Y_{Jm}(\mathbf{x}/\|\mathbf{x}\|)\mathbf{Q}_{Jm}^{\ell k}$$

- Note: $\mathbf{W}^{\ell k}(0) \neq 0 \Rightarrow \ell = k \Rightarrow \mathbf{W}^{\ell\ell} = \mathbf{I}$

Full method pipeline

- TFN structure for values, keys and queries in attention mechanism

$$\mathbf{f}_{\text{out},i}^{\ell} = \underbrace{\mathbf{W}_V^{\ell\ell}\mathbf{f}_{\text{in},i}^{\ell}}_{\text{③ self-interaction}} + \sum_{k\geq 0}\sum_{j\in\mathcal{N}_i\setminus i} \underbrace{\alpha_{ij}}_{\text{① attention}} \underbrace{\mathbf{W}_V^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}}_{\text{② value message}}$$

$$\mathbf{q}_i = \bigoplus_{\ell\geq 0}\sum_{k\geq 0}\mathbf{W}_Q^{\ell k}\mathbf{f}_{\text{in},i}^{k}, \quad \mathbf{k}_{ij} = \bigoplus_{\ell\geq 0}\sum_{k\geq 0}\mathbf{W}_K^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}$$

$$\alpha_{ij} = \frac{\exp(\mathbf{q}_i^{\top}\mathbf{k}_{ij})}{\sum_{j'\in\mathcal{N}_i\setminus i}\exp(\mathbf{q}_i^{\top}\mathbf{k}_{ij'})}$$

Full method pipeline

- TFN structure for values, keys and queries in attention mechanism

$$\mathbf{f}_{\text{out},i}^{\ell} = \underbrace{\mathbf{W}_V^{\ell\ell}\mathbf{f}_{\text{in},i}^{\ell}}_{\text{③ self-interaction}} + \sum_{k\geq 0}\sum_{j\in\mathcal{N}_i\setminus i} \underbrace{\alpha_{ij}}_{\text{① attention}} \underbrace{\mathbf{W}_V^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}}_{\text{② value message}}$$

- Each node aggregates information from its neighbors:
- Two components:
    - **Self-interaction term**
    - **Attention-weighted equivariant message passing**

Full method pipeline

- TFN structure for values, keys and queries in attention mechanism

$$\mathbf{f}_{\text{out},i}^{\ell} = \underbrace{\mathbf{W}_V^{\ell\ell}\mathbf{f}_{\text{in},i}^{\ell}}_{\text{③ self-interaction}} + \sum_{k\geq 0}\sum_{j\in\mathcal{N}_i\setminus i} \underbrace{\alpha_{ij}}_{\text{① attention}} \underbrace{\mathbf{W}_V^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}}_{\text{② value message}}$$

- Additional proposal: Use MLP layers for self-interaction

$$w_{i,c'c}^{\ell\ell} = \text{MLP}\left(\bigoplus_{c,c'} \mathbf{f}_{\text{in},i,c'}^{\ell\top}\mathbf{f}_{\text{in},i,c}^{\ell}\right)$$

Fast Spherical Harmonics

- Explicit formula for Spherical Harmonics
- Converted complex exponentials to sine and cosine
- Recursion to calculate Legendre polynomials

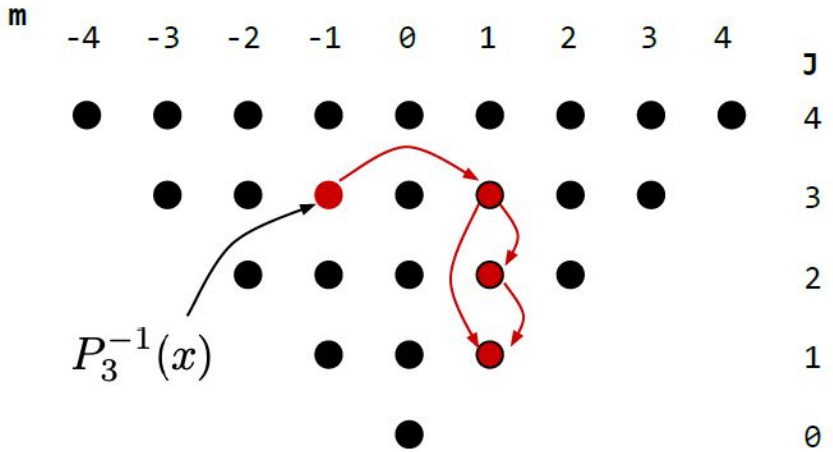
$$Y_{Jm}(\theta,\phi) = \sqrt{\frac{2J+1}{4\pi}\frac{(J-m)!}{(J+m)!}} P_J^{|m|}(\cos\theta) \cdot \begin{cases} \sin(|m|\phi) & m < 0, \\ 1 & m = 0, \\ \cos(m\phi) & m > 0, \end{cases}$$

$$P_J^{|J|}(x) = (-1)^{|J|} \cdot (1-x^2)^{|J|/2} \cdot (2|J|-1)!!$$

$$P_J^{-m}(x) = (-1)^J \frac{(\ell-m)!}{(\ell+m)!} P_J^m(x)$$

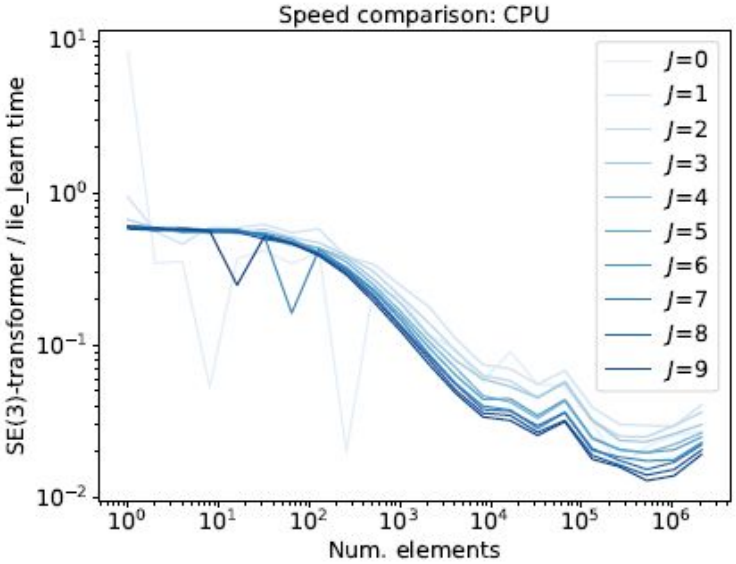
$$P_J^{|m|}(x) = \frac{2J-1}{J-|m|} x P_{J-1}^m(x) + \mathbb{I}[J-|m|>1] \frac{J+|m|-1}{J-|m|} P_{J-2}^m(x)$$

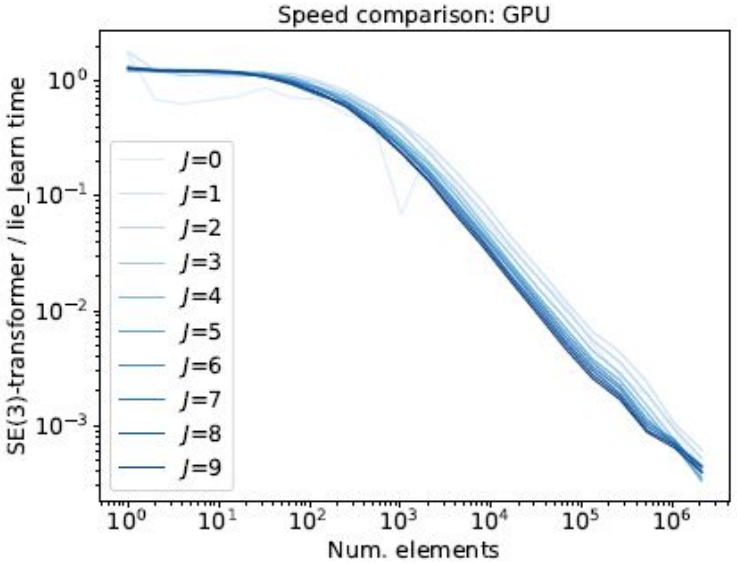
Fast Spherical Harmonics



Subproblem graph for the computation of the associated Legendre polynomial $P_3^{-1}(x)$

Accelerated Computation of Spherical Harmonics





Spherical harmonics computation of fast implementation vs to the lie-learn library.

Experiments and Results

Experiments

1. **N-body Simulations (an equivariant task)**
   Rotation of the input should result in rotated predictions of locations and velocities of the particles.
2. **Real-world object classification task (on ScanObjectNN)**
   The network is confronted with large point clouds of noisy data with symmetry only around the gravitational axis.
3. **Molecular property regression task**
   Show ability to incorporate rich graph structures.

They apply uniformly sampled SO(3) transformations to input/output, and compare "rotate-then-predict" vs "predict-then-rotate."
The normalized distance is their **equivariance error:**

$$\Delta_{EQ} = \|L_s\Phi(f) - \Phi L_s(f)\|_2 \,/\, \|L_s\Phi(f)\|_2$$

N-body Simulations

- Synthetic physical system with **5 interacting particles**
- Each particle has:
  - position, velocity, and charge
- Particles attract or repel based on charge
- **Task**: predict relative positions and velocities **500 steps into the future**
- This is an **SE(3)-equivariant task**, not invariant

N-Body Results: Equivariance and Accuracy

- **Compared models:**
  - Set Transformer (non-equivariant, attention)
  - Tensor Field Network (equivariant, no attention)
  - SE(3)-Transformer (equivariant + attention)

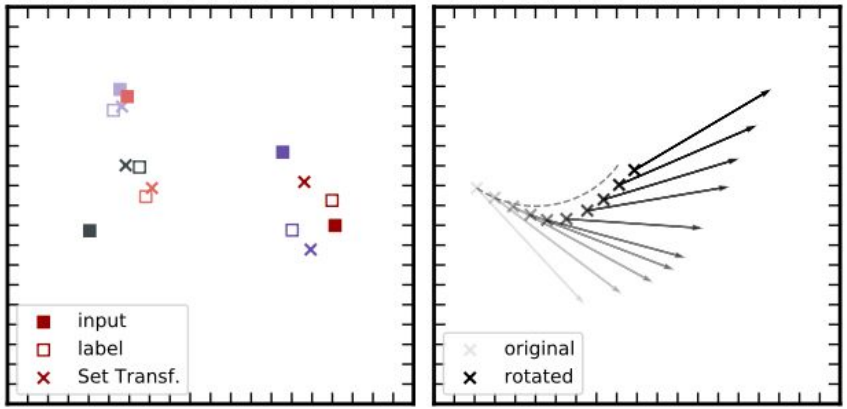
- Figure 3: rotation consistency test

- Table 1: mean squared error (MSE) and equivariance error

rotation consistency test

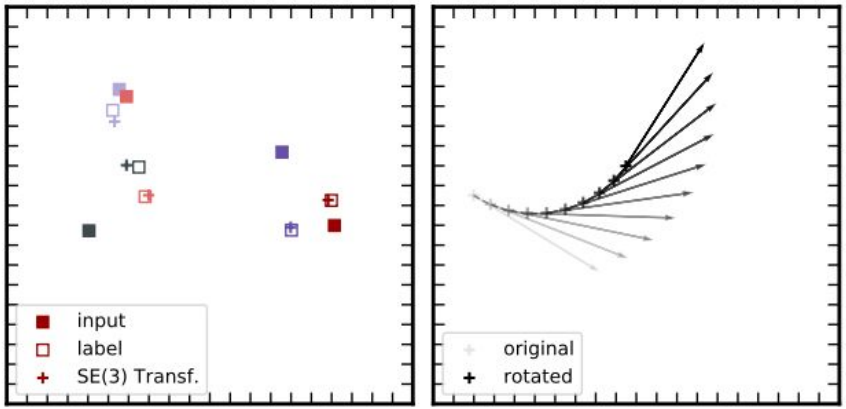
Left plot: input at t=0, ground truth at t=500, and prediction
Then they rotate the input and check what happens to the prediction

Right plot: rotate the input in steps of 10 degrees and plot predicted trajectories/velocities
The dashed curve indicates the prediction of a perfectly equivariant model



(a) Set Transformer

attention-based, but not rotation-equivariant



(b) SE(3)-Transformer

MSE and equivariance error

- **Set Transformer:** good at attention but not equivariant
  → worse equivariance error
- **Tensor Field Network**: equivariant but no attention
  → worse MSE than SE(3)-Transformer
- **SE(3)-Transformer**: **both good accuracy + correct equivariance**

Table 1: Predicting future locations and velocities in an electron-proton simulation.

		Linear	DeepSet [46]	Tensor Field [28]	Set Transformer [16]	**SE(3)-Transformer**
Position	MSE x	0.0691	0.0639	0.0151	0.0139	**0.0076**
	std	-	0.0086	0.0011	0.0004	0.0002
	Δ_{EQ}	-	0.038	$1.9 \cdot 10^{-7}$	0.167	$3.2 \cdot 10^{-7}$
Velocity	MSE v	0.261	0.246	0.125	0.101	**0.075**
	std	-	0.017	0.002	0.004	0.001
	Δ_{EQ}	-	1.11	$5.0 \cdot 10^{-7}$	0.37	$6.3 \cdot 10^{-7}$

Real-world object classification task (on ScanObjectNN)

- Real-world **3D point cloud classification** benchmark
- 2902 objects across **15 categories**
- Input: point coordinates only
- Output: object category
- Dataset is **not fully rotation-invariant**
    - Objects are aligned with a gravity (“up”) direction

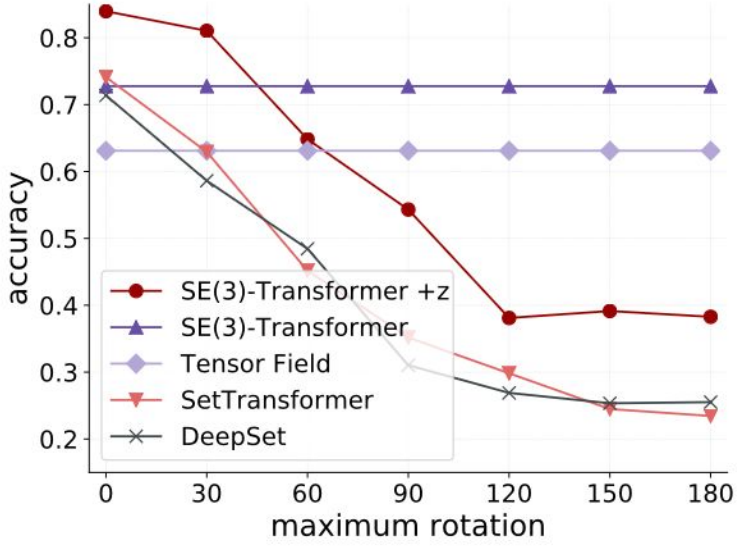
ScanObjectNN Results: Symmetry vs Performance

- Two model variants:
  - SE(3)-Transformer
  - SE(3)-Transformer +z (explicit gravity cue)

- Figure 4: accuracy vs test-time rotation
- Table 2: classification accuracy comparison

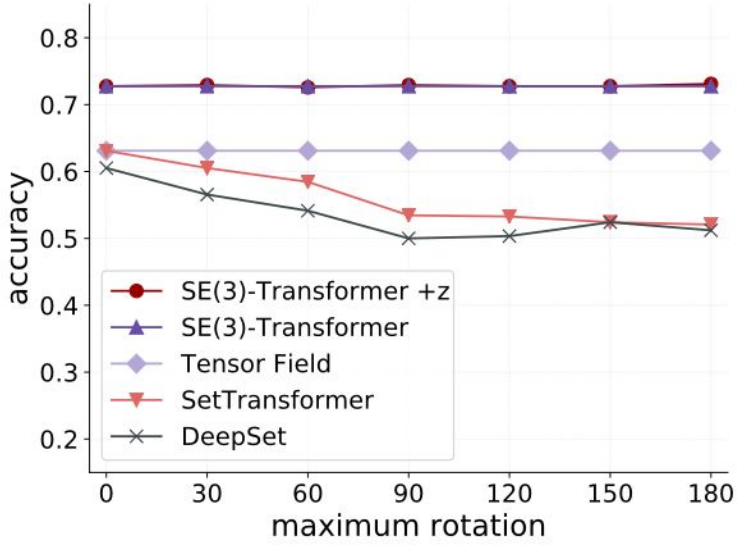
accuracy vs test-time rotation

- **Without augmentation:**
  The model can exploit dataset biases (e.g. “chairs are usually upright”)
- **With augmentation:**
  Orientation becomes unreliable
  The only consistent signal is *rotation-invariant structure*



(a) Training *without* data augmentation.

Can the model exploit dataset orientation bias?



(b) Training *with* data augmentation.

Can the model ignore orientation when needed?

classification accuracy comparison

*"Despite the dataset not playing to the strengths of our model (full SE(3)-invariance) and a much lower number of input points, the performance is **competitive** with models specifically designed for object classification"*

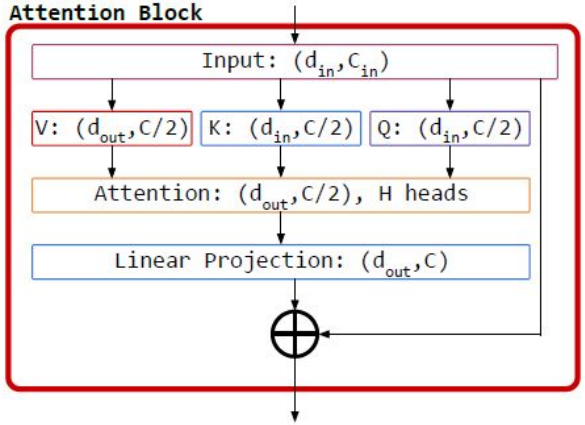
Table 2: Classification accuracy on the 'object only' category of the ScanObjectNN dataset[4]. The performance of the SE(3)-Transformer is averaged over 5 runs (standard deviation 0.7%).

	Tensor Field	DeepSet	**SE(3)-Transf.**	3DmFV	Set Transformer	PointNet	SpiderCNN	Tensor Field +z	PointNet++	**SE(3)-Transf.+z**	PointCNN	DGCNN	PointGLR
No. Points	128	1024	**128**	1024	1024	1024	1024	128	1024	**128**	1024	1024	1024
Accuracy	63.1%	71.4%	**72.8** %	73.8%	74.1%	79.2%	79.5%	81.0%	84.3%	**85.0%**	85.5%	86.2%	87.2%

Experiments with Qm9 Dataset

- Architecture: Seven multihead attention with norms, TFN layer, max pooling and linear layers

No. repeats	Layer Type	d_{out}	C
1x	Input	1	6
1x	Attention: 8 heads	4	16
	Norm Nonlinearity	4	16
6x	Attention: 8 heads	4	16
	Norm Nonlinearity	4	16
1x	TFN layer	1	128
1x	Max pool	1	128
1x	Linear	1	128
1x	ReLU	1	128
1x	Linear	1	1



Experiments with Qm9 Dataset

- Normalization layer

$$\texttt{Norm ReLU}(\mathbf{f}^\ell) = \texttt{ReLU}(\texttt{LN}\left(\|\mathbf{f}^\ell\|\right)) \cdot \frac{\mathbf{f}^\ell}{\|\mathbf{f}^\ell\|}$$

- Radial function structure

LAYER TYPE	C
Input	6
Linear	32
Layer Norm	32
ReLU	32
Linear	32
Layer Norm	32
ReLU	32
Linear	$d_{\text{out}} * C_{\text{in}} * C_{\text{out}}$

Experiments with Qm9 Dataset

- Chemical property prediction task; 134k drug-like molecules, 29 atoms/molecule maximum.
- Five atomic species (Hydrogen, Carbon, Oxygen, Nitrogen, and Fluorine) and four types of chemical bonds (single, double, triple, and aromatic)

TASK UNITS	α bohr^3	$\Delta\varepsilon$ meV	$\varepsilon_{\text{HOMO}}$ meV	$\varepsilon_{\text{LUMO}}$ meV	μ D	C_ν cal/mol K
WaveScatt [11]	.160	118	85	76	.340	.049
NMP [10]	.092	69	43	38	.030	.040
SchNet [25]	.235	63	41	34	.033	.033
Cormorant [1]	.085	61	34	38	.038	.026
LieConv(T3) [8]	.084	49	30	25	.032	.038
TFN [28]	.223	58	40	38	.064	.101
SE(3)-Transformer	.142±.002	53.0±0.3	35.0±.9	33.0±.7	.051±.001	.054±.002

QM9 Mean Absolute Error. Top: Non-equivariant models. Bottom: Equivariant models.

Takeaways

Takeaways

- Attention-based equivariant structure for point cloud data
- Equivariance removes need for rotation augmentation
- Neighborhood attention and fast spherical harmonics for scalability and speed
- Competitive performance with state-of-the-art on experiments

THANK YOU!

Key Design Principle of SE(3)-Transformer

- Attention layer decomposed into two parts:
    - **Attention weights** (scalars)
    - **Value messages** (features)
- If attention weights are **SE(3)-invariant**
- And value messages are **SE(3)-equivariant**
- Then the entire attention layer is **SE(3)-equivariant**

Key Insight:

Equivariance is preserved under weighted sums with invariant weights

SE(3)-Equivariant Attention Layer

- Each node aggregates information from its neighbors:

$$f_{out,i}^{\ell} = W_V^{\ell\ell} f_{in,i}^{\ell} + \sum_{j \in \mathcal{N}_i} \alpha_{ij} W_V^{\ell k}(x_j - x_i) f_{in,j}^{k}$$

- Two components:
  - **Self-interaction term**
  - **Attention-weighted equivariant message passing**

Guarantee:

The layer is provably SE(3)-equivariant

SE(3)-Transformer Layer: Implementation Pipeline

- Construct a local neighborhood graph (e.g. k-NN or molecular bonds)
- Compute relative positions x_j - x_i
- Generate **equivariant kernels** based on relative geometry
- Compute Query, Key, and Value features
- Apply attention and aggregate messages

Complexity:

- Local attention reduces cost from O(N^2)to O(Nk)

Equivariant Kernel Construction

- Kernels depend on **relative position**, not absolute coordinates
- Decomposed into:
    - **Angular component**: spherical harmonics (direction)
    - **Radial component**: learnable functions of distance
- Ensures correct transformation under 3D rotations

Key idea:
Geometry is handled analytically, learning focuses on radial interactions

Why Attention Improves Expressiveness

- Traditional equivariant convolutions use **fixed kernels**
- SE(3)-Transformer introduces **data-dependent weighting**
- Attention modulates interactions based on context
- Preserves equivariance while improving flexibility

Interpretation:
Attention acts as a **learned importance filter** over geometric messages