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

Fabian B. Fuchs [*1 2 3] Daniel E. Worrall [*4 5] Volker Fischer [1] Max Welling [6]

Abstract

We introduce the SE(3)-Transformer, a variant of the self-attention module for 3D point clouds, which is *equivariant* under continuous 3D roto-translations. Equivariance is important to ensure stable and predictable performance in the presence of nuisance transformations of the data input. A positive corollary of equivariance is increased weight-tying within the model, leading to fewer trainable parameters and thus decreased sample complexity (i.e. we need less training data). The SE(3)-Transformer leverages the benefits of self-attention to operate on large point clouds with varying number of points, while guaranteeing SE(3)-equivariance for robustness. We achieve competitive performance on two real-world datasets, ScanObjectNN and QM9.

1. Introduction

Self-attention mechanisms [23] have enjoyed a sharp rise in popularity in the last few years. Their relative implementational simplicity coupled with high efficacy on a wide range of tasks such as language modeling [23], image recognition [14], or graph-based problems [24], make them an attractive component to use. However, their generality of application means that for specific tasks, knowledge of existing underlying structure is unused. In this paper, we propose the *SE(3)-Transformer* shown in Fig. 1, a self-attention mechanism specifically for 3D point cloud data, which adheres to *equivariance constraints*, improving robustness to nuisance transformations and general performance.

Point cloud data is ubiquitous across many fields, presenting itself in diverse forms such as 3D object scans [21], molecular structures [17], or particle simulations [10]. Finding neural structures which can adapt to varying number of points while respecting the irregular sampling of point positions, is challenging. Furthermore, an important property is that these structures should be invariant to global changes in overall input pose; that is, 3D translations and rotations of the input point cloud should not affect the output. In this paper, we find that the explicit imposition of equivariance constraints on the self-attention mechanism addresses these challenges. The SE(3)-Transformer uses the self-attention mechanism as a data-dependent filter particularly suited for sparse, non-voxelised point cloud data, while respecting and leveraging the symmetries of the task at hand.

Self-attention itself is a pseudo-linear map between sets of points. It consists two components: input-dependent *attention weights* and an embedding of the input, called a *value embedding*. In Fig. 1, we show an example of a molecular graph with a value embedding vector attached to every atom and where the attention weights are represented as edges, with width corresponding to the attention weight magnitude. In the SE(3)-Transformer, we explicitly design the attention weights to be invariant to global pose. Furthermore, we design the value embedding to be equivariant to global pose. Equivariance generalises the translational weight-tying of convolutions. It ensures that transformations of a layer's input manifest as equivalent transformations of the output. SE(3)-equivariance in particular is the generalisation of translational weight-tying in 2D known from conventional convolutions to roto-translations in 3D. This restricts the space of learnable functions to a subspace which adheres to the symmetries of the task and thus reduces the number of learnable parameters. Meanwhile, it provides us with a richer form of invariance, since relative positional information between features in the input is preserved.

Applicability to Covid19 Research As our results on the QM9 dataset show, the SE(3)-Transformer is inherently suited for classification and regression problems on molecules lending itself to application in drug research. We are currently investigating using the algorithm for early-stage suitability classification of molecules for inhibiting the reproductive cycle of the coronavirus. While research of this sort always requires intensive testing in wet labs, computer algorithms are being used to filter out promising compounds from large databases of millions of molecules.

2. Background And Related Work

We are concerned with point cloud based machine learning tasks, such as object classification. We are given a point cloud as input, represented as a collection of n coordinate vectors $\mathbf{x}_i \in \mathbb{R}^3$ with optional per-point features $\mathbf{f}_i \in \mathbb{R}^d$.

2.1. The Attention Mechanism

The standard *attention mechanism* [23] can be thought of as consisting of three terms: a set of query vectors $\mathbf{q}_i$, a set of

[*]Equal contribution [1]Bosch Center for Artificial Intelligence (BCAI) [2]A2I Lab, Oxford University [3]Work done while at BCAI [4]AMLab, University of Amsterdam [5]Philips Lab, University of Amsterdam [6]UvA-Bosch Delta Lab, University of Amsterdam. Correspondence to: Fabian Fuchs <fabian@robots.ox.ac.uk>.

ICML 2020 Workshop on Graph Representation Learning and Beyond, Vienna, Austria, 2020. Copyright 2020 by the author(s).

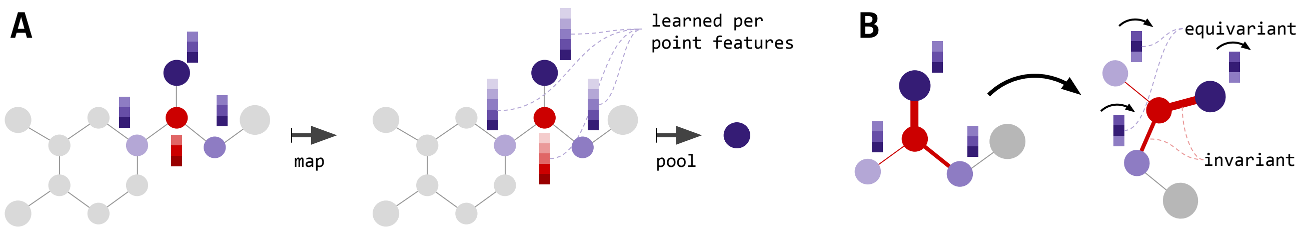

FIGURE 1: **A**) Each layer of the SE(3)-Transformer maps from a point cloud to a point cloud while guaranteeing euqivariance. For classification, this is followed by an invariant pooling layer and an MLP. **B**) In each layer, for each node, attention is performed. Here, the red node attends to its neighbours. Attention weights (indicated by line thickness) are invariant w.r.t. rotation of the input.

key vectors $\mathbf{k}_j$, and a set of value vectors $\mathbf{v}_j$. We interpret key $\mathbf{k}_j$, value $\mathbf{v}_j$ and query $\mathbf{q}_i$ as being 'attached' to the points j and i respectively.

$$\mathrm{Attn}\left(\mathbf{q}_i, \{\mathbf{k}_j, \mathbf{v}_j\}\right) = \sum_{j=1}^{n} \alpha_{ij} \mathbf{v}_j, \quad \alpha_{ij} = \frac{e^{\mathbf{q}_i^\top \mathbf{k}_j}}{\sum_{j'=1}^{n} e^{\mathbf{q}_i^\top \mathbf{k}_{j'}}} \tag{1}$$

In the case of *self-attention* the query, key, and value vectors are embeddings of the input features: $\mathbf{q}=h_Q(\mathbf{f})$, $\mathbf{k}=h_K(\mathbf{f})$, $\mathbf{v}=h_V(\mathbf{f})$ where $\{h_Q, h_K, h_V\}$ are, in the most general case, neural networks [22]. For us, query $\mathbf{q}_i$ is associated with a point i in the input, which has a geometric location $\mathbf{x}_i$. Thus, for n points, we have n possible queries. For query $\mathbf{q}_i$, we say that node i *attends* to all other nodes $j \neq i$.

2.2. Equivariance

Given a set of transformations $T_g{:}\mathcal{V}{\to}\mathcal{V}$ for $g{\in}G$, where G is an abstract group, a function $\phi{:}\mathcal{V}{\to}\mathcal{Y}$ is called equivariant if for every g there exists a $S_g{:}\mathcal{Y}{\to}\mathcal{Y}$ such that

$$S_g[\phi(v)] = \phi(T_g[v]) \qquad \text{for all } g \in G, v \in \mathcal{V}. \tag{2}$$

The indices g can be considered as parameters describing the transformation. Given a pair (T_g, S_g), we can solve for the family of equivariant functions ϕ satisfying Eq. (2). If (T_g, S_g) are linear and the map ϕ is also linear, then a very rich and developed theory already exists for finding ϕ [5]. In the equivariance literature, deep networks are built from interleaved linear maps ϕ and equivariant nonlinearities. For 3D roto-translations, a suitable structure for ϕ is a *Tensor Field Network* [20], explained below.

Group Representations In general, the transformations (T_g, S_g) are called *group representations*. Formally, a group representation $\rho : G \to GL(N)$ is a map from a group G to the set of $N \times N$ invertible matrices $GL(N)$. Critically ρ is a *group homomorphism*; that is, it satisfies the following property $\rho(g_1 g_2) = \rho(g_1)\rho(g_2)$ for all $g_1, g_2 \in G$. Specifically for 3D rotations $G = SO(3)$, it is: 1) its representations are orthogonal matrices, 2) all representations can be decomposed as

$$\rho(g) = \mathbf{Q}^\top \left[\bigoplus_\ell \mathbf{D}_\ell(g)\right] \mathbf{Q}, \tag{3}$$

where $\mathbf{Q}$ is an orthogonal, $N{\times}N$, change-of-basis matrix [4]; each $\mathbf{D}_\ell$ for $\ell = 0, 1, 2, ...$ is a $(2\ell{+}1) \times (2\ell{+}1)$ matrix known as a Wigner-D matrix and the $\bigoplus$ is the *direct sum* or concatenation of matrices along the diagonal. The Wigner-D matrices are *irreducible representations* of SO(3). They are the 'smallest' representations possible. Vectors transforming according to $\mathbf{D}_\ell$ (i.e. we set $\mathbf{Q}{=}\mathbf{I}$, $i{=}\ell$), are called *type-ℓ* vectors. Type-0 vectors are invariant under rotations and type-1 vectors rotate according to 3D rotation matrices. Type-ℓ vectors have length $2\ell{+}1$. They can be stacked, forming a vector $\mathbf{f}$ which transforms via Eq. (3).

Tensor Field Networks (TFN) [20] are neural networks, which map point clouds to point clouds under the constraint of SE(3)-equivariance. For point clouds, the input is a vector field $\mathbf{f}{:}\mathbb{R}^3{\to}\mathbb{R}^d$ of the form $\mathbf{f}(\mathbf{x}){=}\sum_{j=1}^N \mathbf{f}_j \delta(\mathbf{x}{-}\mathbf{x}_j)$, where δ is the Dirac delta function, $\{\mathbf{x}_j\}$ are the 3D point coordinates and $\{\mathbf{f}_j\}$ are point features, representing such quantities as atomic number or point identity. For equivariance to be satisfied, the features of a TFN transform under Eq. (3), where $\mathbf{Q}{=}\mathbf{I}$. Each $\mathbf{f}_j$ is a concatenation of vectors of different *types*, where a subvector of type-ℓ is written $\mathbf{f}_j^\ell$. A TFN layer computes the convolution of a continuous-in-space, learnable weight kernel $\mathbf{W}^{\ell k}{:}\mathbb{R}^3{\to}\mathbb{R}^{(2\ell+1)\times(2k+1)}$ from type-k features to type-ℓ features. The type-ℓ output of the TFN layer at position $\mathbf{x}_i$ is

$$\begin{aligned} \mathbf{f}^\ell_{\text{out},i} &= \sum_{k\geq 0} \underbrace{\int \mathbf{W}^{\ell k}(\mathbf{x}' - \mathbf{x}_i)\mathbf{f}^k_{\text{in}}(\mathbf{x}')\, \mathrm{d}\mathbf{x}'}_{k\to\ell \text{ convolution}} \\ &= \sum_{k\geq 0} \sum_{j=1}^{n} \underbrace{\mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}^k_{\text{in},j}}_{\text{node } j \,\to\, \text{node } i \text{ message}}, \end{aligned} \tag{4}$$

We can also include a sum over input channels, but we omit it here. Weiler et al. [26], Thomas et al. [20] and Kondor [11] showed that the kernel $\mathbf{W}^{\ell k}$ lies in the span of an equivariant basis $\{\mathbf{W}^{\ell k}_J\}_{J=|k-\ell|}^{k+\ell}$. The kernel is a linear combination of these basis kernels, where the J^{th} coefficient is a learnable function $\varphi^{\ell k}_J : \mathbb{R}_{\geq 0} \to \mathbb{R}$ of the radius $\|\mathbf{x}\|$:

$$\mathbf{W}^{\ell k}(\mathbf{x}) = \sum_{J=|k-\ell|}^{k+\ell} \varphi^{\ell k}_J(\|\mathbf{x}\|)\mathbf{W}^{\ell k}_J(\mathbf{x}), \tag{5}$$

where $\mathbf{W}^{\ell k}_J(\mathbf{x}) = \sum_{m=-J}^{J} Y_{Jm}(\mathbf{x}/\|\mathbf{x}\|)\mathbf{Q}^{\ell k}_{Jm}$. Each basis kernel $\mathbf{W}^{\ell k}_J : \mathbb{R}^3 \to \mathbb{R}^{(2\ell+1)\times(2k+1)}$ is formed by taking

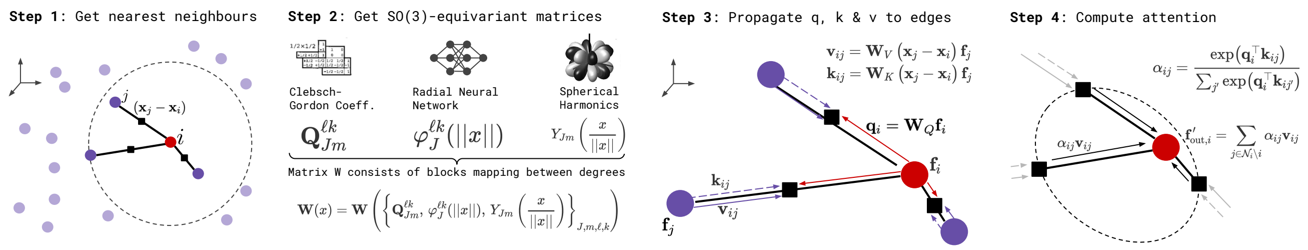
Step 1: Get nearest neighbours
j
$(\mathbf{x}_j - \mathbf{x}_i)$
i
Step 2: Get SO(3)-equivariant matrices
Clebsch-Gordon Coeff.
Radial Neural Network
Spherical Harmonics
$\mathbf{Q}_{Jm}^{\ell k}$
$\varphi_J^{\ell k}(||x||)$
$Y_{Jm}\left(\frac{x}{||x||}\right)$
Matrix W consists of blocks mapping between 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 q, k & v to edges
$\mathbf{v}_{ij} = \mathbf{W}_V\left(\mathbf{x}_j - \mathbf{x}_i\right)\mathbf{f}_j$
$\mathbf{k}_{ij} = \mathbf{W}_K\left(\mathbf{x}_j - \mathbf{x}_i\right)\mathbf{f}_j$
$\mathbf{q}_i = \mathbf{W}_Q \mathbf{f}_i$
$\mathbf{f}_i$
$\mathbf{k}_{ij}$
$\mathbf{v}_{ij}$
$\mathbf{f}_j$
Step 4: Compute attention
$\alpha_{ij} = \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_{ij})}{\sum_{j'}\exp(\mathbf{q}_i^\top \mathbf{k}_{ij'})}$
$\alpha_{ij}\mathbf{v}_{ij}$
$\mathbf{f}'_{\text{out},i} = \sum_{j\in\mathcal{N}_i\setminus i} \alpha_{ij}\mathbf{v}_{ij}$

FIGURE 2: Updating node features in four steps. Steps 3 and 4 visualise a graph network perspective: features are passed from nodes to edges to compute keys, queries and values, which depend both on features and relative positions in a rotation-equivariant manner.

a linear combination of Clebsch-Gordan matrices $\mathbf{Q}_{Jm}^{\ell k}$ of shape $(2\ell+1)\times(2k+1)$, where the J, m^{th} linear combination coefficient is the m^{th} dimension of the J^{th} spherical harmonic $Y_J : \mathbb{R}^3 \to \mathbb{R}^{2J+1}$. Each basis kernel $\mathbf{W}_J^{\ell k}$ completely constrains the kernel in the angular direction, leaving the only learnable degree of freedom in the radial direction. Note that $\mathbf{W}_J^{\ell k}(\mathbf{0}) \neq \mathbf{0}$ only when $k=\ell$ and $J=0$, which reduces the kernel to a scalar w multiplied by the identity, $\mathbf{W}^{\ell\ell} = w^{\ell\ell}\mathbf{I}$, referred to as *self-interaction* [20]. As such we can rewrite the TFN layer as

$$\mathbf{f}_{\text{out},i}^{\ell} = \underbrace{w^{\ell\ell}\mathbf{f}_{\text{in},i}^{\ell}}_{\text{self-interaction}} + \sum_{k\geq 0}\sum_{j\neq i}^{n} \mathbf{W}^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}, \qquad (6)$$

Eq. (4) and Eq. (6) present the convolution in message-passing form, where messages are aggregated from all nodes and feature types. They are a form of nonlocal graph operation, where the weights are functions on edges and $\{\mathbf{f}_i\}$ are node features. We show how our attention layer unifies aspects of convolutions and graph neural networks.

3. Method

Here, we present the *SE(3)-Transformer*. The layer can be broken down into a procedure of steps as shown in Fig. 2, which we describe in the following section. These are the construction of a graph from a point cloud, the construction of equivariant edge functions on the graph, how to propagate SE(3)-equivariant messages on the graph, and how to aggregate them. We also introduce an alternative for the self-interaction layer, which we call *attentive self-interaction*.

Neighbourhoods (see Step 1 of Fig. 2) Given a point cloud $\{(\mathbf{x}_i, \mathbf{f}_i)\}$, we first introduce a collection of neighbourhoods $\mathcal{N}_i \subseteq \{1, ..., N\}$, one centered on each point i. These neighbourhoods are computed either via the nearest-neighbours methods or may already be defined. For instance, molecular structures have neighbourhoods defined by their bonding structure. Neighbourhoods reduce the computational complexity of the attention mechanism from quadratic in the number of points to linear. The introduction of neighbourhoods converts our point cloud into a graph.

The SE(3)-Transformer itself consists of three components. These are 1) edge-wise attention weights α_{ij}, constructed to be SE(3)-invariant on each edge ij, 2) edge-wise SE(3)-equivariant value messages, propagating information between nodes, as found in the TFN convolution of Eq. (4), and 3) a linear/attentive self-interaction layer. Attention is performed on a per-neighbourhood basis as follows:

$$\mathbf{f}_{\text{out},i}^{\ell} = \underbrace{\mathbf{W}_V^{\ell\ell}\mathbf{f}_{\text{in},i}^{\ell}}_{③\ \text{self-interact.}} + \sum_{k\geq 0}\sum_{j\in\mathcal{N}_i\setminus i} \underbrace{\alpha_{ij}}_{①\ \text{att.}} \underbrace{\mathbf{W}_V^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}}_{②\ \text{value message}} \qquad (7)$$

These components are visualised in Fig. 2. If we remove the attention weights then we have a tensor field convolution, and if we instead remove the dependence of $\mathbf{W}_V$ on $(\mathbf{x}_j - \mathbf{x}_i)$, we have a conventional attention mechanism. Provided that the attention weights α_{ij} are invariant, Eq. (7) is equivariant to SE(3)-transformations. This is because it is just a linear combination of equivariant value messages. Invariant attention weights can be achieved with a dot-product attention structure shown in Eq. (8). This mechanism consists of a normalised inner product between a query vector $\mathbf{q}_i$ at node i and a set of key vectors $\{\mathbf{k}_{ij}\}_{j\in\mathcal{N}_i}$ along each edge ij in the neighbourhood $\mathcal{N}_i$ where

$$\alpha_{ij} = \frac{e^{\mathbf{q}_i^\top \mathbf{k}_{ij}}}{\sum_{j'\in\mathcal{N}_i\setminus i} e^{\mathbf{q}_i^\top \mathbf{k}_{ij'}}}, \quad \mathbf{q}_i = \bigoplus_{\ell\geq 0}\sum_{k\geq 0}\mathbf{W}_Q^{\ell k}\mathbf{f}_{\text{in},i}^{k}, \qquad (8)$$

$$\mathbf{k}_{ij} = \bigoplus_{\ell\geq 0}\sum_{k\geq 0}\mathbf{W}_K^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)\mathbf{f}_{\text{in},j}^{k}. \qquad (9)$$

$\bigoplus$ is the direct sum, i.e. vector concatenation in this instance. The linear embedding matrices $\mathbf{W}_Q^{\ell k}$ and $\mathbf{W}_K^{\ell k}(\mathbf{x}_j - \mathbf{x}_i)$ are of TFN type (c.f. Eq. (5)). The attention weights α_{ij} are invariant for the following reason. If the input features $\{\mathbf{f}_{\text{in},j}\}$ are SO(3)-equivariant, then the query $\mathbf{q}_i$ and key vectors $\{\mathbf{k}_{ij}\}$ are also SE(3)-equivariant, since the linear embedding matrices are of TFN type. The inner product of SO(3)-equivariant vectors, transforming under the same representation $\mathbf{S}_g$ is invariant, since if $\mathbf{q} \mapsto \mathbf{S}_g\mathbf{q}$ and $\mathbf{k} \mapsto \mathbf{S}_g\mathbf{k}$, then $\mathbf{q}^\top\mathbf{S}_g^\top\mathbf{S}_g\mathbf{k} = \mathbf{q}^\top\mathbf{k}$, because of the orthonormality of representations of SO(3), mentioned in the background section. We chose a softmax nonlinearity to normalise the attention weights following, e.g., [23, 12].

Angular Modulation The attention weights add extra degrees of freedom to the TFN kernel in the angular direction. This is seen when Eq. (7) is viewed as a convolution with a data-dependent kernel $\alpha_{ij}\mathbf{W}_V^{\ell k}(\mathbf{x})$. In the literature, SO(3) equivariant kernels are decomposed as a sum of products of learnable radial functions $\varphi_J^{\ell k}(\|\mathbf{x}\|)$ and non-learnable angular kernels $\mathbf{W}_J^{\ell k}(\mathbf{x}/\|\mathbf{x}\|)$ (c.f. Eq. (5)). The fixed angular

TABLE 1: Classification on the 'object only' category of the ScanObjectNN dataset. Std of SE(3)-Transformer over 5 runs is 0.7%.

	DeepSet[30]	3DmFV[3]	Set Transformer[12]	PointNet[15]	SpiderCNN[29]	Tensor Field +z[20]	PointNet++[16]	SE(3)-Transf.+z	PointCNN[13]	DGCNN[25]	PointGLR[18]
No. Points	1024	1024	1024	1024	1024	128	1024	**128**	1024	1024	1024
Accuracy	71.4%	73.8%	74.1%	79.2%	79.5%	81.0%	84.3%	**85.0%**	85.5%	86.2%	87.2%

dependence of $\mathbf{W}_J^{\ell k}(\mathbf{x}/\|\mathbf{x}\|)$ is an artifact of the equivariance condition in noncommutative algebras and while necessary to guarantee equivariance, it is seen as overconstraining the expressiveness of the kernels. The attention weights α_{ij} introduce a means to modulate the angular profile of $\mathbf{W}_J^{\ell k}(\mathbf{x}/\|\mathbf{x}\|)$, while maintaining equivariance.

Channels, Self-interaction Layers, and Non-Linearities Analogous to conventional neural networks, the SE(3)-Transformer can straightforwardly be extended to multiple channels per representation degree ℓ, so far omitted for brevity. This sets the stage for self-interaction layers. The attention layer (c.f. Fig. 2 and circles 1 and 2 of Eq. (7)) aggregates information over nodes and input representation degrees k. In contrast, the self-interaction layer (c.f. circle 3 of Eq. (7)) exchanges information solely between features of the same degree and within one node—much akin to 1x1 convolutions in CNNs. In our experiments, we use two different types of self-interaction layer: (1) linear and (2) attentive, both of the form

$$\mathbf{f}_{\text{out},i,c'}^{\ell} = \sum_{c} w_{i,c'c}^{\ell\ell} \mathbf{f}_{\text{in},i,c}^{\ell}. \tag{10}$$

Linear: Following Schütt et al. [19], output channels are a learned linear combination of input channels using one set of weights $w_{i,c'c}^{\ell\ell} = w_{c'c}^{\ell\ell}$ per representation degree, shared across all points. As proposed in Thomas et al. [20], this is followed by a norm-based non-linearity.

Attentive: We propose an extension of linear self-interaction, *attentive self-interaction*, combining self-interaction and nonlinearity. We replace the learned scalar weights $w_{c'c}^{\ell\ell}$ with attention weights output from an MLP, shown in Eq. (11) ($\bigoplus$ means concatenation.). These weights are SE(3)-invariant due to the invariance of inner products of features, transforming under the same representation.

$$w_{i,c'c}^{\ell\ell} = \text{MLP}\left(\bigoplus_{c,c'} \mathbf{f}_{\text{in},i,c'}^{\ell\top} \mathbf{f}_{\text{in},i,c}^{\ell}\right) \tag{11}$$

Node and Edge Features Point cloud data often has information attached to points (node-features) and connections between points (edge-features), which we would both like to pass as inputs into the first layer of the network. Node information can directly be incorporated via the tensors $\mathbf{f}_j$ in Section 2.2 and eq. (7). For incorporating edge information, note that $\mathbf{f}_j$ is part of multiple neighbourhoods. One can replace $\mathbf{f}_j$ with $\mathbf{f}_{ij}$ in Eq. (7). Now, $\mathbf{f}_{ij}$ can carry different information depending on which neighbourhood $\mathcal{N}_i$ we are currently performing attention over. In other words, $\mathbf{f}_{ij}$ can carry information both about node j but also about edge ij. Alternatively, if the edge information is scalar, it can be incorporated into the weight matrices $\mathbf{W}_V$ and $\mathbf{W}_K$ as an input to the radial network (see step 2 in Fig. 2).

4. Experiments

Real-World Object Classification ScanObjectNN is a recently introduced dataset for real-world object classification. The benchmark provides point clouds of 2902 objects across 15 different categories. We only use the coordinates of the points as input and object categories as training labels. We train an SE(3)-Transformer with 4 equivariant layers with linear self-interaction followed by max-pooling and an MLP. Interestingly, the task is not fully rotation invariant, in a statistical sense, as the objects are aligned w.r.t. the gravity axis. This results in a performance loss when deploying an SO(3) invariant model. We create an SO(2) invariant version, *SE(3)-Transformer +z*, by additionally feeding the z-component as an type-0 field and the x, y position as an additional type-1 field (see Appendix). In Table 1, we compare our model to the current state-of-the-art in object classification. 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.

QM9 The QM9 regression dataset [17] is a publicly available chemical property prediction task. There are 134k molecules with up to 29 atoms per molecule. Atoms are represented as a 5 dimensional one-hot node embeddings in a molecular graph connected by 4 different chemical bond types (more details in Appendix). 'Positions' of each atom are provided. We show results on the test set of Anderson et al. [1] for 6 regression tasks in Table 2. Lower is better. The table is split into non-equivariant (top) and equivariant models (bottom). Our nearest models are Cormorant and TFN (own). We see that while not state-of-the-art, we offer competitive performance, especially against Cormorant and TFN, which transform under irreducible representations of SE(3) (like us), unlike LieConv(T3), using a left-regular representation of SE(3), which may explain its success.

TABLE 2: QM9 Mean Absolute Error. Top: Non-equivariant models. Bottom: Equivariant models

TASK	α	$\Delta\varepsilon$	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	μ	C_ν
UNITS	bohr^3	meV	meV	meV	D	cal/mol K
WaveScatt [8]	.160	118	85	76	.340	.049
NMP [7]	.092	69	43	38	.030	.040
SchNet [19]	.235	63	41	34	.033	.033
Cormorant [1]	.085	61	34	38	.038	.026
LieConv(T3) [6]	.084	49	30	25	.032	.038
TFN [20]	.223	58	40	38	.064	.101
Us	.148	53	36	33	.053	.057

Acknowledgements

We would like to express our gratitude to the Bosch Center for Artificial Intelligence and Konincklijke Philips N.V. for their support and contribution to open research in publishing our paper.

References

[1] Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2019.

[2] Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL http://arxiv.org/abs/1607.06450.

[3] Ben-Shabat, Y., Lindenbaum, M., and Fischer., A. 3dmfv: Three-dimensional point cloud classification in realtime using convolutional neural networks. *IEEE Robotics and Automation Letters*, 2018.

[4] Chirikjian, G. S., Kyatkin, A. B., and Buckingham, A. Engineering applications of noncommutative harmonic analysis: with emphasis on rotation and motion groups. *Appl. Mech. Rev.*, 54(6):B97–B98, 2001.

[5] Cohen, T. S. and Welling, M. Steerable cnns. *International Conference on Learning Representations (ICLR)*, 2017.

[6] Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. 2020.

[7] Gilmer, J., Schoenholz, S. S., Riley, P. F., and Dahl, O. V. G. E. Neural message passing for quantum chemistry. 2020.

[8] Hirn, M. J., Mallat, S., and Poilvert, N. Wavelet scattering regression of quantum chemical energies. *Multiscale Model. Simul.*, 15(2):827–863, 2017.

[9] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

[10] Kipf, T. N., Fetaya, E., Wang, K., Welling, M., and Zemel, R. S. Neural relational inference for interacting systems. In *Proceedings of the International Conference on Machine Learning, ICML*, 2018.

[11] Kondor, R. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint*, 2018.

[12] Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the International Conference on Machine Learning, ICML*, 2019.

[13] Li, Y., Bu, R., Sun, M., and Chen, B. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[14] Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. Stand-alone self-attention in vision models. In *Advances in Neural Information Processing System (NeurIPS)*, 2019.

[15] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[16] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[17] Ramakrishnan, R., Dral, P., Rupp, M., and von Lilienfeld, A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 08 2014.

[18] Rao, Y., Lu, J., and Zhou, J. Global-local bidirectional reasoning for unsupervised representation learning of 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] Schütt, K. T., Kindermans, P.-J., Sauceda, H. E., Chmiela1, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[20] Thomas, N., Smidt, T., Kearnes, S. M., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *ArXiv Preprint*, 2018.

[21] Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, D. T., and Yeung, S.-K. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

[22] van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *International Conference on Learning Representations (ICLR)*, 2018.

[23] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[24] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. *International Conference on Learning Representations (ICLR)*, 2018.

[25] Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[26] Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[27] Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance, 2016.

[28] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[29] Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *European Conference on Computer Vision (ECCV)*, 2018.

[30] Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. Deep Sets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

A. Object Classification Experiments

A particularity of object classification from point clouds is the large number of points the algorithm needs to handle. We use up to 200 points out of the available 2024 points per sample and create neighbourhoods using up to 40 nearest neighbours. It is worth pointing out that especially in this setting, adding self-attention (i.e. when comparing the SE(3) Transformer to Tensor Field Networks) significantly increased the stability. As a result, when we swapped out the attention mechanism for a convolution to retrieve the Tensor Field network baseline, we had to decrease the model size to obtain stable training. However, we would like to stress that the Tensor Field network we trained was significantly bigger than in the original paper [20], mostly enabled by a faster computation of the spherical harmonics with our optimised pytorch implementation. We will make this code available to the public.

For the quantitative comparison to the start-of-the-art in Table 1, we used 128 input points and neighbourhood size 10 for both the Tensor Field network baseline and the Table 1. We used farthest point sampling with a random starting point to retrieve the 128 points from the overall point cloud. We used degrees up to 3 and 5 channels per degree, which we had to reduce to 3 channels for the Tensor Field network to obtain stable training. We used a norm based non-linearity for the Tensor Field network (as in [20]) and no extra non-linearity (beyond the *softmax* in the self-attention algorithm) for the SE(3) Transformer.

The final layer of the equivariant encoder maps to 64 channels of degree 0 representations. This yields a 64-dimensional SE(3) *invariant* representation per point. Next, we pool over the point dimension followed by an MLP with one hidden layer of dimension 64, a ReLU and a 15 dimensional output with a cross entropy loss. We trained for 60000 steps with batch size 10. We used the Adam optimizer [9] with a start learning of 0.01 and a reduction of the learning rate by 70% every 15000 steps.

The input to the Tensorfield network and the Se(3) Transformer are relative x-y-z positions of each point w.r.t. their neighbours. To guarantee equivariance, these inputs are provided as fields of degree 1. For the '+z' versions, however, we deliberately break the SE(3) equivariance by providing additional and relative z-position as two additional scalar fields (i.e. degree 0), as well as relative x-y positions as a degree 1 field (where the z-component is set to 0).

DeepSet Baseline We originally replicated the implementation proposed in [30] for their object classification experiment on ModelNet40 [28]. However, most likely due to the relatively small number of objects in the ScanObjectNN dataset, we found that reducing model size helped the performance significantly. The reported model had 128 units per hidden layer (instead of 256) and no dropout but the same number of layers and type of non-linearity as in [30].

Set Transformer Baseline We used the same architecture as [12] in their object classification experiment on ModelNet40 [28] with an ISAB (induced set attention block)-based encoder followed by PMA (pooling by multihead attention) and an MLP.

B. QM9 Experiments

The QM9 regression dataset [17] is a publicly available chemical property prediction task consisting of 134k small drug-like organic molecules with up to 29 atoms per molecule. There are 5 atomic species (Hydrogen, Carbon, Oxygen, Nitrogen, and Flourine) in a molecular graph connected by chemical bonds of 4 types (single, double, triple, and aromatic bonds). 'Positions' of each atom, measured in ångtröms, are provided. We used the exact same train/validation/test splits as Anderson et al. [1] of sizes 100k/18k/13k.

The architecture we used is shown in Table 3. It consists of 7 multihead attention layers interspersed with norm nonlinearities, followed by a TFN layer, max pooling, and two linear layers separated by a ReLU. For each attention layer, we embed the input to half the number of feature channels before applying multiheaded attention [23]. Multiheaded attention is a variation of attention, where we partition the queries, keys, and values into H *attention heads*. So if our embeddings have dimensionality $(4, 16)$ (denoting 4 feature types with 16 channels each) and we use $H = 8$ attention heads, then we partition the embeddings to shape $(4, 2)$. We then combine each of the 8 sets of shape $(4, 2)$ queries, keys, and values individually and then concatenate the results into a single vector of the original shape $(4, 16)$. The keys and queries are edge embeddings, and thus the embedding matrices are of TFN type (c.f. Eq. (5)). For TFN type layers, the radial functions are learnable maps. For these we used neural networks with architecture shown in Table 4.

For the norm nonlinearities [27], we use

$$\texttt{Norm ReLU}(\mathbf{f}^\ell) = \texttt{ReLU}(\texttt{LN}\left(\|\mathbf{f}^\ell\|\right)) \cdot \frac{\mathbf{f}^\ell}{\|\mathbf{f}^\ell\|} \tag{12}$$

$$\|\mathbf{f}^\ell\| = \sqrt{\sum_{m=-\ell}^{\ell} (f_m^\ell)^2}, \tag{13}$$

where LN is layer norm [2] applied across all features within a feature type. For the TFN baseline, we used the exact same architecture but we replaced each of the multiheaded attention blocks with a TFN layer with the same output shape.

The input to the network is a sparse molecular graph, with edges represented by the molecular bonds. The node embeedings are a 6 dimensional vector composed of a 5 dimensional one-hot embedding of the 5 atomic species and a 1 dimension integer node embedding for number of protons per atom. The edges embeddings are a 5 dimensional vector

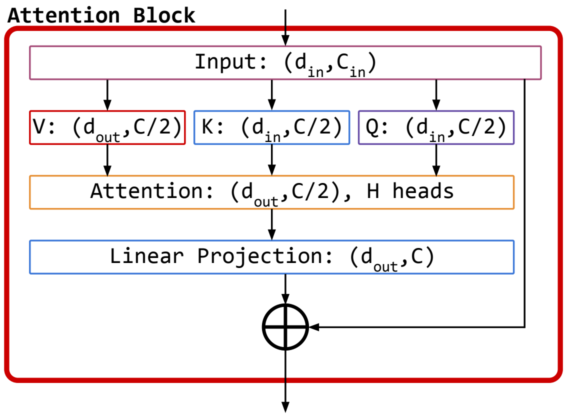

FIGURE 3: Attention block for the QM9 dataset. Each component is listed with a tuple of numbers representing the output feature types and multiplicities, so $(4, 32)$ means feature types $0, 1, 2, 3$ (with dimensionalities $1, 3, 5, 7$), with 32 channels per type.

consisting of a 4 dimensional one-hot embedding of bond type and a positive scalar for the Euclidean distance between the two atoms at the ends of the bond. For each regression target, we normalised the values by mean and dividing by the standard deviation of the training set.

We trained for 50 epochs using Adam [9] at initial learning rate `1e-3` and a single-cycle cosine rate-decay to learning rate `1e-4`. The batch size was 32, but for the TFN baseline we used batch size 16, to fit the model in memory. We show results on the 6 regression tasks not requiring thermochemical energy subtraction in Table 2. As is common practice, we optimised architectures and hyperparameters on $\varepsilon_{\text{HOMO}}$ and retrained each network on the other tasks. Training took about 2.5 days on an NVIDIA GeForce GTX 1080 Ti GPU with 4 CPU cores and 15 GB of RAM.

TABLE 3: QM9 Network architecture: d_{out} is the number of feature types of degrees $0, 1, ..., d_{\text{out}} - 1$ at the output of the corresponding layer and C is the number of multiplicities/channels per feature type. For the norm nonlinearity we use ReLUs, preceded by equivariant layer norm [26] with learnable affine transform.

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

TABLE 4: QM9 Radial Function Architecture. C is the number of output channels at each layer. Layer norm [2] is computed per pair of input and output feature types, which have C_{in} and C_{out} channels each.

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