LieTransformer: Equivariant Self-Attention for Lie Groups

Michael Hutchinson [*1] **Charline Le Lan** [*1] **Sheheryar Zaidi** [*1]
Emilien Dupont [1] **Yee Whye Teh** [1 2] **Hyunjik Kim** [2]

Abstract

Group equivariant neural networks are used as building blocks of group invariant neural networks, which have been shown to improve generalisation performance and data efficiency through principled parameter sharing. Such works have mostly focused on group equivariant convolutions, building on the result that group equivariant linear maps are necessarily convolutions. In this work, we extend the scope of the literature to *self-attention*, that is emerging as a prominent building block of deep learning models. We propose the `LieTransformer`, an architecture composed of `LieSelfAttention` layers that are equivariant to arbitrary Lie groups and their discrete subgroups. We demonstrate the generality of our approach by showing experimental results that are competitive to baseline methods on a wide range of tasks: shape counting on point clouds, molecular property regression and modelling particle trajectories under Hamiltonian dynamics.

1. Introduction

Group equivariant neural networks are useful architectures for problems with symmetries that can be described in terms of a group (in the mathematical sense). Convolutional neural networks (CNNs) are a special case that deal with translational symmetry, in that when the input to a convolutional layer is translated, the output is also translated. This property is known as *translation equivariance*, and offers a useful inductive bias for perception tasks which usually have translational symmetry. Constraining a linear layer to obey this symmetry, resulting in a covolutional layer, greatly reduces the number of parameters and computational cost. This has led to the success of CNNs in multiple domains such as computer vision (Krizhevsky et al., 2012) and audio (Graves & Jaitly, 2014). Following on from this success, there has been a growing literature on the study of group equivariant CNNs (G-CNNs) that generalise CNNs to deal with other types of symmetries beyond translations, such as rotations and reflections.

Most works on group equivariant NNs deal with CNNs i.e. linear maps with shared weights composed with pointwise non-linearities, building on the result that group equivariant linear maps (with mild assumptions) are necessarily convolutions (Kondor & Trivedi, 2018; Cohen et al., 2019; Bekkers, 2020). However there has been little work on non-linear group equivariant building blocks. In this paper we extend group equivariance to self-attention (Vaswani et al., 2017), a non-trivial non-linear map, that has become a prominent building block of deep learning models in various data modalities, such as natural-language processing (Vaswani et al., 2017; Brown et al., 2020), computer vision (Zhang et al., 2019; Parmar et al., 2019b), reinforcement learning (Parisotto et al., 2020), and audio generation (Huang et al., 2019).

We thus propose `LieTransformer`, a group invariant Transformer built from group equivariant `LieSelfAttention` layers. It uses a lifting based approach, that relaxes constraints on the attention module compared to approaches without lifting. Our method is applicable to Lie groups and their discrete subgroups (e.g. cyclic groups C_n and dihedral groups D_n) acting on homogeneous spaces. Our work is very much in the spirit of Finzi et al. (2020), our main baseline, but for group equivariant self-attention instead of convolutions. Among works that deal with equivariant self-attention, we are the first to propose a methodology for general groups and domains (unspecified to 2D images (Romero et al., 2020; Romero & Cordonnier, 2021) or 3D point clouds (Fuchs et al., 2020)). We demonstrate the generality of our approach through strong performance on a wide variety of tasks, namely shape counting on point clouds, molecular property regression and modelling particle trajectories under Hamiltonian dynamics.

---

[*] Equal contribution, with alphabetical ordering. See Appendix A for detailed contributions. Please cite as: [Hutchinson, Le Lan, Zaidi et al. 2020]. [1] Department of Statistics, University of Oxford, UK [2] DeepMind, UK. Correspondence to: Michael Hutchinson <hutchinson.michael.john@gmail.com>, Charline Le Lan <charline.lelan@stats.ox.ac.uk>, Sheheryar Zaidi <sheh_zaidi96@hotmail.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

2. Background

2.1. Group Equivariance

This section lays down some of the necessary definitions and notations in group theory and representation theory in an informal and intuitive manner. For a more formal presentation of definitions, see Appendix B.

Loosely speaking, a **group** G is a set of symmetries, with each group element g corresponding to a symmetry transformation. These group elements ($g, g' \in G$) can be composed (gg') or inverted (g^{-1}), just like transformations. An example of a **discrete group** is C_n, the set of rotational symmetries of a regular n-gon. The group consists of n such rotations, including the identity. An example of a continuous (infinite) group is $SO(2)$, the set of all 2D rotations about the origin. C_n is a subset of $SO(2)$, hence we call C_n a **subgroup** of $SO(2)$. Note that $SO(2) = \{g_\theta : \theta \in [0, 2\pi)\}$ can be parameterised by the angle of rotation θ. Such groups that can be continuously parameterised by real values are called **Lie groups**.

A symmetry transformation of group element $g \in G$ on object $v \in V$ is referred to as the **group action** of G on V. If this action is linear on a vector space V, then we can represent the action as a linear map $\rho(g)$. We call ρ a **representation** of G, and $\rho(g)$ often takes the form of a matrix. For $SO(2)$, the standard rotation matrix is an example of a representation that acts on $V = \mathbb{R}^2$:

$$\rho(g_\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

Note that this is only one of many possible representations of $SO(2)$ acting on $\mathbb{R}^2$ (e.g. replacing θ with $n\theta$ yields another valid representation), and $SO(2)$ can act on spaces other than $\mathbb{R}^2$, e.g. $\mathbb{R}^d$ for arbitrary $d \geq 2$.

In the context of group equivariant neural networks, V is commonly defined to be the space of scalar-valued functions on some set S, so that $V = \{f \mid f : S \to \mathbb{R}\}$. This set could be a Euclidean input space e.g. a grey-scale image can be expressed as a feature map $f : \mathbb{R}^2 \to \mathbb{R}$ from pixel coordinate x_i to pixel intensity $\mathtt{f}_i$, supported on the grid of pixel coordinates. We may express the rotation of the image as a representation of $SO(2)$ by extending the action ρ on the pixel coordinates to a representation π that acts on the space of feature maps:

$$[\pi(g_\theta)(f)](x) \triangleq f(\rho(g_\theta^{-1})x). \tag{2}$$

Note that this is equivalent to the mapping $(x_i, \mathtt{f}_i)_{i=1}^n \mapsto (\rho(g_\theta)x_i, \mathtt{f}_i)_{i=1}^n$. As a special case, we can define $V = \{f | f : G \to \mathbb{R}\}$ to be the space of scalar-valued functions on the group G, for which we can define a representation π acting on V via the **regular representation**:

$$[\pi(g_\theta)(f)](g_\phi) \triangleq f(g_\theta^{-1} g_\phi). \tag{3}$$

Here the action ρ is replaced by the action of the group on itself. If we wish to handle multiple channels of data, e.g. RGB images, we can stack these feature maps together, transforming in a similar manner.

Now let us define the notion of G-**equivariance**.

Definition 1. *We say that a map $\Phi : V_1 \to V_2$ is G-**equivariant** with respect to actions ρ_1, ρ_2 of G acting on V_1, V_2 respectively if: $\Phi[\rho_1(g)f] = \rho_2(g)\Phi[f]$ for any $g \in G, f \in V_1$.*

In the above example of rotating RGB images, we have $G = SO(2)$ and $\rho_1 = \rho_2 = \pi$. Hence the equivariance of Φ with respect to $SO(2)$ means that rotating an input image and then applying Φ yields the same result as first applying Φ to the original input image and then rotating the output, i.e. Φ *commutes* with the representation π.

The end goal for group equivariant neural networks is to design a neural network that obeys certain symmetries in the data. For example, we may want an image classifier to output the same classification when the input image is rotated. So in fact we want a G-**invariant** neural network, where the output is invariant to group actions on the input space. Note that G-invariance is a special case of G-equivariance, where ρ_2 is the **trivial representation** i.e. $\rho_2(g)$ is the identity map for any $g \in G$. Invariant maps are easy to design, by discarding information, e.g. pooling over spatial dimensions is invariant to rotations and translations. However, such maps are not expressive as they fail to extract high-level semantic features from the data. This is where equivariant neural networks become relevant; the standard recipe for constructing an expressive invariant neural network is to compose multiple equivariant layers with a final invariant layer. It is a standard result that such maps are invariant (e.g. Bloem-Reddy & Teh (2020)) and a proof is given in Appendix C for completeness.

2.2. Equivariant Maps on Homogeneous Input Spaces

Here we introduce the framework for G-equivariant maps, and provide group equivariant convolutions as an example. Suppose we have data in the form of a set of input pairs $(x_i, \mathtt{f}_i)_{i=1}^n$ where $x_i \in \mathcal{X}$ are spatial coordinates and $\mathtt{f}_i \in \mathcal{F}$ are feature values. The data can be described as a single feature map $f_\mathcal{X} : x_i \mapsto \mathtt{f}_i$. We assume that a group G acts on the x-space $\mathcal{X}$ via action ρ, and that the action is **transitive** (also referred to as $\mathcal{X}$ being **homogeneous**). This means that all elements of $\mathcal{X}$ are connected by the action: $\forall x, x' \in \mathcal{X}$, $\exists g \in G : \rho(g)x = x'$. We often write gx instead of $\rho(g)x$ to reduce clutter. For example, the group of 2D translations $T(2)$ acts transitively on $\mathbb{R}^2$ since there is a translation connecting any two points in $\mathbb{R}^2$. On the other hand, the group of 2D rotations about the origin $SO(2)$ does not act transitively on $\mathbb{R}^2$, since points that have different

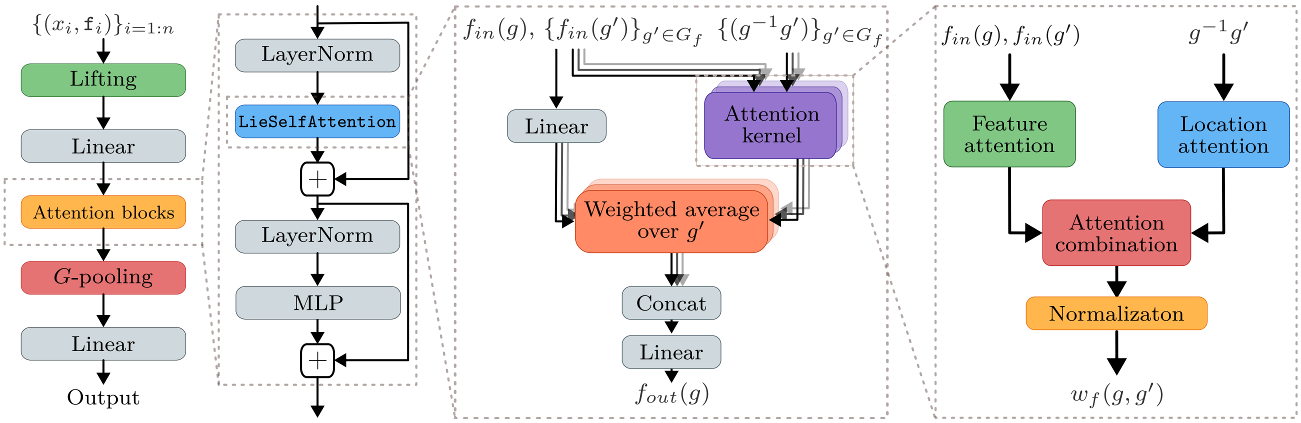

Figure 1. Architecture of the LieTransformer.

distances to the origin cannot be mapped onto each other by rotations. However the group of 2D roto-translations $SE(2)$, whose elements can be written as a composition tR of $t \in T(2)$ and $R \in SO(2)$, acts transitively on $\mathbb{R}^2$ since $SE(2)$ contains $T(2)$.

For such homogeneous spaces $\mathcal{X}$, it can be shown that there is a natural partition of G into disjoint subsets such that there is a one-to-one correspondence between $\mathcal{X}$ and these subsets. Namely each $x \in \mathcal{X}$ corresponds to the **coset** $s(x)H = \{s(x)h | h \in H\}$, where the subgroup $H = \{g \in G | gx_0 = x_0\}$ is called the **stabiliser** of origin x_0, and $s(x) \in G$ is a group element that maps x_0 to x. It can be shown that the coset $s(x)H$ does not depend on the choice of $s(x)$, and that $s(x)H$ and $s(x')H$ are disjoint for $x \neq x'$. For $T(2)$ acting on $\mathbb{R}^2$, we have $H = \{e\}$, the identity, and $s(x) = t_x$, the group element describing the translation from x_0 to x, and so each x corresponds to $\{t_x\}$. For $SE(2)$ acting on $\mathbb{R}^2$, we have $H = SO(2)$ and $s(x) = t_x$, so each x corresponds to $\{t_x R | R \in SO(2)\}$. This correspondence is often written as an isomorphism $X \simeq G/H$, where G/H is the set of cosets of H.

Using this isomorphism, we can map each point in $\mathcal{X}$ to a set of group elements in G, i.e. mapping each data pair $(x_i, \mathtt{f}_i)$ to (possibly multiple) pairs $\{(g, \mathtt{f}_i) | g \in s(x_i)H\}$. This can be thought of as **lifting** the feature map $f_{\mathcal{X}} : x_i \mapsto \mathtt{f}_i$ defined on $\mathcal{X}$ to a feature map $\mathcal{L}[f_{\mathcal{X}}] : g \mapsto \mathtt{f}_i$ defined on G (Kondor & Trivedi, 2018). Let $\mathcal{I}_U$ denote the space of such feature maps from G to $\mathcal{F}$. Subsequently, we may define group equivariant maps as functions from $\mathcal{I}_U$ to itself, which turns out to be a simpler task than defining equivariant maps directly on $\mathcal{X}$.

The **group equivariant convolution** (Cohen & Welling, 2016; Cohen et al., 2018; Finzi et al., 2020; Romero et al., 2020) is an example of such a group equivariant map that has been studied extensively. Specifically, the group equivariant convolution $\Psi : \mathcal{I}_U \to \mathcal{I}_U$ is defined as:

$$[\Psi f](g) \triangleq \int_G \psi(g'^{-1}g) f(g') dg' \tag{4}$$

where $\psi : G \to \mathbb{R}$ is the convolutional filter and the integral is defined with respect to the left Haar measure of G. Note that for discrete groups the integral amounts to a sum over the group. Hence the integral can be computed exactly for discrete groups (Cohen & Welling, 2016), and for Lie groups it can be approximated using Fast Fourier Transforms (Cohen et al., 2018) or Monte Carlo (MC) estimation (Finzi et al., 2020). Given the regular representation π of G acting on $\mathcal{I}_U$ as in Equation 3, we can easily verify that Ψ is equivariant with respect to π (c.f. Appendix C).

3. LieTransformer

We first outline the problem setting before describing our model, the `LieTransformer`. We tackle the problem of regression/classification for predicting a scalar/vector-valued target y given a set of input pairs $(x_i, \mathtt{f}_i)_{i=1}^n$ where $x_i \in \mathbb{R}^{d_x}$ are spatial locations and $\mathtt{f}_i \in \mathbb{R}^{d_f}$ are feature values at the spatial location. Hence the training data of size N is a set of tuples $((x_i, \mathtt{f}_i)_{i=1}^{n_j}, y_j)_{j=1}^N$. In some tasks such as point cloud classification, the feature values $\mathtt{f}_i$ may not be given. In this case the $\mathtt{f}_i$ can set to be a fixed constant or a (G-invariant) function of $(x_i)_{i=1}^n$.

`LieTransformer` is composed of a **lifting** layer followed by residual blocks of `LieSelfAttention` layers, `LayerNorm` and pointwise `MLP`s, all of which are equivariant with respect to the regular representation, followed by a final invariant `G-pooling` layer (c.f. Appendix H for more details on these layers). We summarise the architecture in Figure 1 and describe its key components below.

3.1. Lifting

Recall from Section 2.2 that the **lifting** $\mathcal{L}$ maps $f_{\mathcal{X}}$ (supported on $\bigcup_{i=1}^n \{x_i\} \subset \mathcal{X}$) to $\mathcal{L}[f_{\mathcal{X}}]$ (supported on

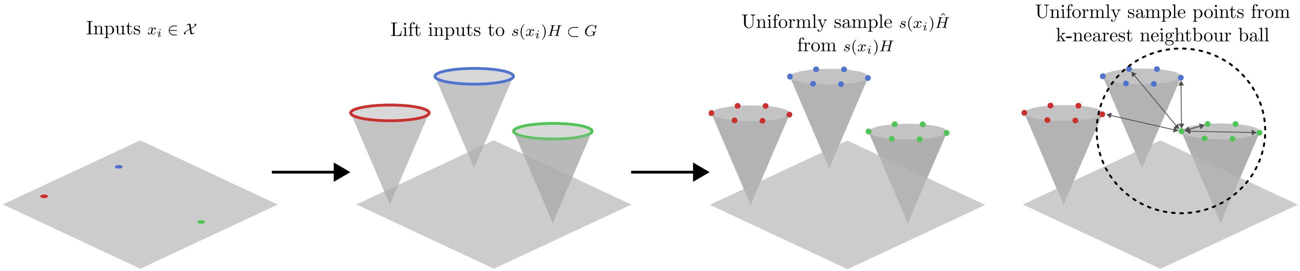

Figure 2. Visualisation of lifting, sampling $\hat{H}$, and subsampling in the local neighbourhood for $SE(2)$ acting on $\mathbb{R}^2$. Self-attention is performed on this subsampled neighbourhood.

$\bigcup_{i=1}^{n} s(x_i)H \subset G$) such that:

$$\mathcal{L}[f_{\mathcal{X}}](g) \triangleq \mathtt{f}_i \quad \text{for} \quad g \in s(x_i)H. \tag{5}$$

This can be thought of as extending the domain of $f_{\mathcal{X}}$ from $\mathcal{X}$ to G while preserving the feature values $\mathtt{f_i}$, mapping $(x_i, \mathtt{f}_i) \mapsto (g, \mathtt{f}_i)$ for $g \in s(x_i)H$ (c.f. Figure 2). Subsequently we may design G-equivariant maps on the space of functions on G, which is a simpler task than desgining G-equivariant maps directly on $\mathcal{X}$ (e.g. Cohen et al. (2018)).

As in Equations 2 and 3, we define the representation π on $f_{\mathcal{X}}$ and $\mathcal{L}[f_{\mathcal{X}}]$ as:

$$[\pi(u)f_{\mathcal{X}}](x) = f_{\mathcal{X}}(u^{-1}x)$$
$$[\pi(u)\mathcal{L}[f_{\mathcal{X}}]](g) = \mathcal{L}[f_{\mathcal{X}}](u^{-1}g)$$

where $u \in G$. Note that the actions correspond to mappings $(x_i, \mathtt{f}_i) \mapsto (ux_i, \mathtt{f}_i)$ and $(g, \mathtt{f}_i) \mapsto (ug, \mathtt{f}_i)$ respectively.

We need to ensure that lifting preserves equivariance, which is why we need the space to be homogeneous with respect to the action of G on $\mathcal{X}$.

Proposition 1. *The lifting layer $\mathcal{L}$ is equivariant with respect to the representation π.*

Intuition for proof. When x_i is shifted by $u \in G$, the lifted coset $s(x_i)H$ is also shifted by u, i.e. $x_i \mapsto ux_i \Rightarrow s(x_i)H \mapsto us(x_i)H$. See Appendix C for full proof.

3.2. LieSelfAttention

Let $f \triangleq \mathcal{L}[f_{\mathcal{X}}]$, hence f is defined on the set $G_f = \cup_{i=1}^{n} s(x_i)H$. We define the `LieSelfAttention` layer in Algorithm 1, where self-attention (see Appendix D for the original formulation) is defined across the elements of G_f. There are various choices for functions content-based attention k_c , location-based attention k_l , F that determines how to combine the two to form unnormalised weights, and the choice of normalisation of weights. See Appendix E for a non-exhaustive list of choices of the above, and also for the details of the multi-head generalisation of `LieSelfAttention`.

Algorithm 1 LieSelfAttention

```
Input: $(g, f(g))_{g \in G_f}$
  for $g \in G_f$
    for $g' \in G_f$ (or $\mathtt{nbhd}_\eta(g)$)
      ▷ Compute content/location attention
      $k_c(f(g), f(g')), k_l(g^{-1}g')$
      ▷ Compute unnormalised weights
      $\alpha_f(g, g') = F(k_c(f(g), f(g')), k_l(g^{-1}g'))$
    ▷ Compute normalised weights and output
    $\{w_f(g, g')\}_{g' \in G_f} = \mathtt{norm}\{\alpha_f(g, g')\}_{g' \in G_f}$
    $f_{out}(g) = \int_{G_f} w_f(g, g') W^V f(g') dg'$
Output: $(g, f_{out}(g))_{g \in G_f}$
```

Proposition 2. *LieSelfAttention is equivariant with respect to the regular representation π.*

Intuition for proof. `LieSelfAttention` can be thought of as a map $\Phi : (g, f(g))_{g \in G_f} \mapsto (g, f_{out}(g))_{g \in G_f}$, and equivariance holds if $\forall u \in G$, Φ maps $(ug, f(g))_{g \in G_f}$ to $(ug, f_{out}(g))_{g \in G_f}$. Now note that Φ is a function of $g \in G_f$ only via $g^{-1}g'$ for $g' \in G_f$, and $g^{-1}g'$ is invariant to the group action $g \mapsto ug, g' \mapsto ug'$. This is enough to show that Φ satisfies the above condition for equivariance. See Appendix C for full proof.

Generalisation to infinite G_f. For Lie Groups, G_f is usually infinite (it need not be if H is discrete e.g. for $T(n)$ acting on $\mathbb{R}^n$, we have $H = \{e\}$ hence G_f is finite). To deal with this case we resort to Monte Carlo (MC) estimation to approximate the integral in `LieSelfAttention`, following the approach of Finzi et al. (2020):

1. Replace $G_f \triangleq \cup_{i=1}^{n} s(x_i)H$ with a finite subset $\hat{G}_f \triangleq \cup_{i=1}^{n} s(x_i)\hat{H}$ where $\hat{H}$ is a finite subset of H sampled uniformly. We refer to $|\hat{H}|$ as the number of *lift samples*.
2. (Optional, for computational efficiency) Further replace $\hat{G}_f$ with uniform samples from the neighbourhood $\mathtt{nbhd}_\eta(g) \triangleq \{g' \in \hat{G}_f : d(g, g') \leq \eta\}$ for some

threshold η where distance is measured by the log map $d(g, g') = ||\nu[\log(g^{-1}g')]||$ (c.f. Appendix F).

See Figure 2 for a visualisation. Due to MC estimation we now have equivariance in expectation as Finzi et al. (2020). For sampling within the neighbourhood, we can show that the resulting `LieSelfAttention` is still equivariant in expectation given that the distance is a function of $g^{-1}g'$ (c.f. Appendix C).

4. Related Work

Equivariant maps with/without lifting Equivariant neural networks can be broadly categorised by whether the input spatial data is *lifted* onto the space of functions on group G or not. Without lifting, the equivariant map is defined between the space of functions/features on the homogeneous input space X, with equivariance imposing a constraint on the parameterisation of the convolutional kernel or attention module (Cohen & Welling, 2017; Worrall et al., 2017; Thomas et al., 2018; Kondor et al., 2018; Weiler et al., 2018b;a; Weiler & Cesa, 2019; Esteves et al., 2020; Fuchs et al., 2020). In the case of convolutions, the kernel is expressed using a basis of equivariant functions such as circular or spherical harmonics. However with lifting, the equivariant map is defined between the space of functions/features on G, and aforementioned constraints on the convolutional kernel or attention module are relaxed at the cost of an increased dimensionality of the input to the neural network (Cohen & Welling, 2016; Cohen et al., 2018; Esteves et al., 2018; Finzi et al., 2020; Bekkers, 2020; Romero & Hoogendoorn, 2020; Romero et al., 2020; Hoogeboom et al., 2018). Our method also uses lifting to define equivariant self-attention.

Equivariant self-attention Most of the above works use equivariant convolutions as the core building block of their equivariant module, drawing from the result that bounded linear operators are group equivariant if and only if they are convolutions (Kondor & Trivedi, 2018; Cohen et al., 2019; Bekkers, 2020). Such convolutions are used with pointwise non-linearities (applied independently to the features at each spatial location/group element) to form expressive equivariant maps. Exceptions to this are Romero et al. (2020) and Fuchs et al. (2020) that explore equivariant attentive convolutions, reweighing convolutional kernels with attention weights. This gives non-linear equivariant maps with non-linear interactions across spatial locations/group elements. Instead, our work removes convolutions and investigates the use of equivariant self-attention only, inspired by works that use stand-alone self-attention on images to achieve competitive performance to convolutions (Parmar et al., 2019a; Dosovitskiy et al., 2020). Furthermore, Romero et al. (2020) focus on image applications (hence scalability) and discrete groups (p4, p4m), and Fuchs et al. (2020) focus on 3D point cloud applications and the $SE(3)$ group with irreducible representations acting on functions on $\mathcal{X}$. Instead we use regular representations actingon functions on G, and give a general method for Lie groups acting on homogeneous spaces, with a wide range of applications from dealing with point cloud data to modelling Hamiltonian dynamics of particles. This is very much in the spirit of Finzi et al. (2020), except for self-attention instead of convolutions. In concurrent work, Romero & Cordonnier (2021) describe group equivariant self-attention also using lifting and regular representations. Their analogue of location-based attention are group invariant positional encodings. The main difference between the two works is that Romero & Cordonnier (2021) specify methodology for discrete groups applied to image classification only and it is not clear how to extend their approach to Lie groups. In contrast, our method provides a general formula for (unimodular) Lie groups and their discrete subgroups for the aforementioned applications.

5. Experiments

We consider three different tasks that have certain symmetries, highlighting the benefits of the `LieTransformer`: (1) Counting shapes in 2D point cloud of constellations (2) Molecular property regression and (3) Modelling particle trajectories under Hamiltonian dynamics.[1]

5.1. Counting Shapes in 2D Point Clouds

We first consider the toy, synthetic task of counting shapes in a 2D point cloud $\{x_1, x_2, ..., x_K\}$ of constellations (Kosiorek et al., 2019), mainly to check that `LieTransformer` has the correct invariance properties. We use $\mathtt{f}_i = 1$ for all points. Each example consists of points in the plane that form the vertices of a pattern. There are four types of patterns: triangles, squares, pentagons and the 'L' shape, with varying sizes, orientation, and number of instances per pattern (see Figure 3 (right)). The task is to classify the number of instances of each pattern, hence is invariant to 2D roto-translations $SE(2)$.

We first create a fixed training set D_{train} and test set D_{test} of size 10,000 and 1,000 respectively. We then create augmented test sets D_{test}^{T2} and D_{test}^{SE2} that are copies of D_{test} with arbitrary transformations in $T(2)$ and $SE(2)$ respectively. In Table 1, we evaluate the test accuracy of `LieTransformer` at convergence with and without data augmentation during training time – D_{train}^{T2} and D_{train}^{SE2} indicate random $T(2)$ and $SE(2)$ augmentations respectively to each batch of D_{train} at every training iteration. We evaluate the test performance of `LieTransformer-T2` and `LieTransformer-SE2`

---

[1] The code for our experiments is available at: `https://github.com/oxcsml/lie-transformer`

Training data **Test data**	D_{train} D_{test}	D_{train} D^{T2}_{test}	D_{train} D^{SE2}_{test}	D^{T2}_{train} D^{T2}_{test}	D^{T2}_{train} D^{SE2}_{test}	D^{SE2}_{train} D^{SE2}_{test}
SetTransformer	0.58 ± 0.07	0.44 ± 0.02	0.44 ± 0.02	0.61 ± 0.02	0.51 ± 0.01	0.55 ± 0.01
LieTransformer-T2	0.75 ± 0.03	0.75 ± 0.03	0.63 ± 0.06	0.75 ± 0.03	0.63 ± 0.06	0.70 ± 0.03
LieTransformer-SE2	0.71 ± 0.01	0.71 ± 0.01	0.69 ± 0.02	0.71 ± 0.01	0.69 ± 0.02	0.72 ± 0.04

Table 1. Mean and standard deviation of test accuracies on the shape counting task at convergence (over 8 random initialisations).

that are invariant to $T(2)$ and $SE(2)$ respectively, against the baseline `SetTransformer` (Lee et al., 2019), a Transformer-based model that is permutation invariant, but not invariant to rotations nor translations. We use a similar number of parameters for each model. See Appendix I.1 for further details on the setup.

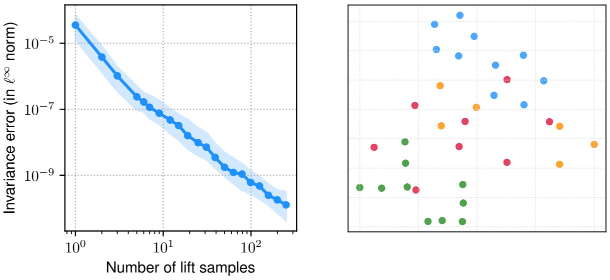

Figure 3. (Left) $SE(2)$ invariance error on output logits vs. number of lift samples for a single layer `LieTransformer-SE2`. Plot shows median and interquartile range across 100 runs, randomizing over model seed, input example and transformation applied to input. (Right) An example 2D point cloud from D_{train}. Each colour corresponds to a different pattern.

Note that the test accuracy of `LieTransformer-T2` and `LieTransformer-SE2` remains unchanged when the train/test set is augmented with $T(2)$. For `LieTransformer-SE2`, this is not quite true for $SE(2)$ augmentations because the model is only $SE(2)$ equivariant in expectation and not exactly equivariant given a finite number of lifts samples ($|\hat{H}|$). However the changes in accuracy for $SE(2)$ augmentation are much smaller compared to `LieTransformer-T2`. The test accuracy of `SetTransformer`, the non-invariant baseline, is always lower than `LieTransformer`. Note that `LieTransformer-T2` does slightly better than `LieTransformer-SE2` on D_{test} and D^{T2}_{test}. We suspect that the variance in the sampling of the lifting layer for `LieTransformer-SE2` is making optimisation more difficult, and will continue to explore these results.

In Figure 3 (left), we report the equivariance error of `LieTransformer-SE2` when increasing the number of lift samples ($|\hat{H}|$) used in the Monte Carlo approximation of `LieSelfAttention`. As expected the invariance error decreases monotonically with the number of lift samples, and already with 3 lift samples, the error is small ($\approx 10^{-6}$).

5.2. QM9: Molecular Property Regression

We apply the `LieTransformer` to the QM9 molecule property prediction task (Ruddigkeit et al., 2012; Ramakrishnan et al., 2014). This dataset consists of 133,885 small inorganic molecules described by the location and charge of each atom in the molecule, along with the bonding structure of the molecule. The dataset includes 19 properties of each molecule, such as various rotational constants, energies and enthalpies, and 12 of these are used as regression tasks. We expect these molecular properties to be invariant to 3D roto-translations $SE(3)$. We follow the customary practice of performing hyperparameter search on the ϵ_{HOMO} task and use the same hyperparameters for training on the other 11 tasks. Further details of the exact experimental setup can be found in Appendix I.2.

We trained four variants of both `LieTransformer` and `LieConv`, namely the $T(3)$ and $SE(3)$ invariant models with and without $SO(3)$ (rotation) data augmentation. We set x_i to be the atomic position and $\mathtt{f}_i$ to be the charge. Table 2 shows the test error of all models and baselines on the 12 tasks. The table is divided into 3 sections. **Upper**: non-invariant models specifically designed for the QM9 task. **Middle**: invariant models specifically designed for the QM9 task. **Lower**: invariant models that are general-purpose. We show very competitive results, and perform best of general-purpose models on 8/12 tasks. In particular when comparing against `LieConv`, we see better performance on the majority of tasks, suggesting that the attention framework is better suited to these tasks than convolutions.

As expected for both `LieTransformer` and `LieConv`, the $SE(3)$ models tend to outperform the $T(3)$ models without $SO(3)$ data augmentation (on 10/12 tasks and 7/12 tasks respectively), showing that being invariant to rotations improves generalisation. Moreover the $SE(3)$ models perform similarly with and without augmentation, whereas the $T(3)$ models greatly benefit from augmentation, showing evidence that the $SE(3)$ models are indeed invariant to rotations. However the $T(3)$ models with augmentation outperform the $SE(3)$ counterparts on most tasks for both `LieTransformer-SE3` and `LieConv-SE3`. As for the experiments in Section 5.1, we suspect that the variance in the sampling of the lifting layer of $SE(3)$ models, along with the $SE(3)$ log-map (Appendix F) in the location attention is making optimisation more difficult, and plan to

Task	α	$\Delta\epsilon$	ϵ_{HOMO}	ϵ_{LUMO}	μ	C_ν	G	H	R^2	U	U_0	ZPVE
Units	bohr^3	meV	meV	meV	D	cal/mol K	meV	meV	bohr^2	meV	meV	meV
WaveScatt (Hirn et al., 2017)	.160	118	85	76	.340	.049	–	–	–	–	–	–
NMP (Gilmer et al., 2017)	**.092**	69	43	38	**.030**	.040	19	17	.180	20	20	**1.50**
SchNet (Schütt et al., 2017)	.235	**63**	**41**	**34**	.033	**.033**	**14**	**14**	**<u>.073</u>**	**19**	**14**	1.70
Cormorant (Anderson et al., 2019)	.085	61	34	38	.038	.026	20	21	.961	21	22	2.03
DimeNet++ (Klicpera et al., 2020) *	.049	34	26	20	.033	.024	7.7	7.1	.387	6.7	6.9	1.23
L1Net (Miller et al., 2020)	.088	68	45	35	.043	.031	14	14	.354	14	13	1.56
TFN (Thomas et al., 2018)	.223	58	40	38	.064	.101	–	–	–	–	–	–
SE3-Transformer (Fuchs et al., 2020)	.148	53	36	33	.053	.057	–	–	–	–	–	–
LieConv-T3 (Finzi et al., 2020) †	.125	60	36	32	.057	.046	35	37	1.54	36	35	3.62
LieConv-T3 + SO3 Aug (Finzi et al., 2020)	.084	49	30	**25**	**.032**	.038	22	24	.800	19	19	2.28
LieConv-SE3 (Finzi et al., 2020)†	.097	**45**	**27**	**25**	.039	.041	39	46	2.18	49	48	3.27
LieConv-SE3 + SO3 Aug (Finzi et al., 2020)†	.088	**45**	**27**	**25**	.038	.043	47	46	2.12	44	45	3.25
LieTransformer-T3 (Us)	.179	67	47	37	.063	.046	27	29	.717	27	28	2.75
LieTransformer-T3 + SO3 Aug (Us)	**.082**	51	33	27	.041	**.035**	**19**	**17**	**.448**	**16**	**17**	**2.10**
LieTransformer-SE3 (Us)	.104	52	33	29	.061	.041	23	27	2.29	26	26	3.55
LieTransformer-SE3 + SO3 Aug (Us)	.105	52	33	29	.062	.041	22	25	2.31	24	25	3.67

Table 2. QM9 molecular property prediction mean absolute error. Bold indicates best performance in a given section, underlined indicates best overall performance. *These results are from our own runs of the Dimenet++ model. The original paper used different train/valid/test splits to the other papers listed here. †These results are from our owns runs of LieConv as these ablations were not present in the original paper.

continue investigating the source of this discrepancy in performance. Note however that `LieTransformer-SE3` and `LieConv-SE3` tend to outperform the irreducible representation (irrep) based $SE(3)$-Transformer and TFN. This can be seen as further evidence that regular representation approaches tend to outperform irrep approaches, in line with the empirical observations of Weiler & Cesa (2019).

5.3. Modelling Particle Trajectories with Hamiltonian Dynamics

We also apply the `LieTransformer` to a physics simulation task in the context of Hamiltonian dynamics, a formalism for describing the evolution of a physical system using a single scalar function $H(q, p)$, called the Hamiltonian.

We consider the case of n particles in d dimensions, writing the position $\mathbf{q} \in \mathbb{R}^{nd}$ and momentum $\mathbf{p} \in \mathbb{R}^{nd}$ of all particles as a single state $\mathbf{z} = (\mathbf{q}, \mathbf{p})$. The Hamiltonian $H : \mathbb{R}^{2nd} \to \mathbb{R}$ takes as input $\mathbf{z}$ and returns its total (potential plus kinetic) energy. The time evolution of the particles is then given by *Hamilton's equations*:

$$\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t} = \frac{\partial H}{\partial \mathbf{p}}, \quad \frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -\frac{\partial H}{\partial \mathbf{q}}. \tag{6}$$

Several recent works have shown that modelling physical systems by learning its Hamiltonian significantly outperforms approaches that learn the dynamics directly (Greydanus et al., 2019; Sanchez-Gonzalez et al., 2019; Zhong et al., 2020; Finzi et al., 2020). Specifically, we can parameterise the Hamiltonian of a system by a neural network H_θ that is learned by ensuring that trajectories from the ground truth and learned system are close to each other. Given a learned H_θ, we can simulate the system for T timesteps by solving equation (6) with a numerical ODE solver to obtain a trajectory $\{\hat{\mathbf{z}}_t(\theta)\}_{t=1}^T$ and minimize the ℓ^2-norm between this trajectory and the ground truth $\{\mathbf{z}_t\}_{t=1}^T$.

However we know a-priori that such physical systems have symmetries, namely conserved quantities such as linear and angular momentum. A notable result is Noether's theorem (Noether, 1918), which states that the system has a conserved quantity if and only if the Hamiltonian is group-invariant. For example, translation invariance of the Hamiltonian implies conservation of momentum and rotation invariance implies conservation of angular momentum. Hence in our experiments, we parameterise the Hamiltonian H_θ by a `LieTransformer` and endow it with the symmetries corresponding to the conservation laws of the physical system we are modelling. We test our model on the spring dynamics task proposed in Sanchez-Gonzalez et al. (2019) – we consider a system of 6 particles with randomly sampled massses in 2D, where each particle connected to all others by springs. This system conserves both linear and angular momentum, so the ground truth Hamiltonian will be both translationally and rotationally invariant, that is, $SE(2)$-invariant. We simulate this system for 500 timesteps from random initial conditions and use random subsets of length 5 from these roll-outs to train the model (see Appendix I.3 for full experimental details).

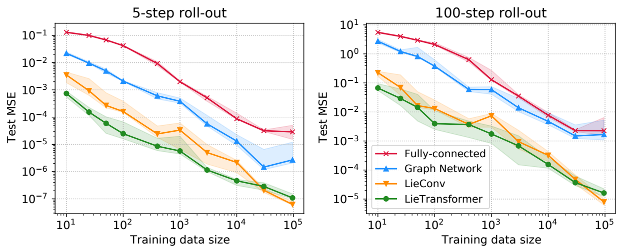

Figure 4. Data efficiency on Hamiltonian spring dynamics. All models are trained using 5-step roll-outs, with test performance on 5-step (left) and 100-step (right) roll-outs. Plots show median MSE and interquartile range (IQR) across 10 random seeds.

We compare our method to different parameterisations of H_θ, namely Fully-connected network (Chen et al., 2018), Graph Network (Sanchez-Gonzalez et al., 2019) and `LieConv`. Only `LieTransformer` and `LieConv` incorporate invariance. In Figures 4, 5, and 6, we use `LieTransformer-T2` and `LieConv-T2` since Finzi et al. (2020) report that there are numerical instabilities for `LieConv-SE2` on this task, due to which `LieConv-T2` is their default model and performs the best. However in Figure 7, we also consider $SE(2)$-invariant versions of both models with modifications to the lifting procedure, which fixed the instabilities as outlined in Appendix F.

Figure 4 compares the performance of all methods as a function of the number of training examples. `LieTransformer` is highly data-efficient: the inductive bias from the symmetries of the Hamiltonian allow us to accurately learn the dynamics even from a small training set. Our method consistently outperforms non-invariant methods (fully-connected and graph networks), typically by 1-3 orders of magnitude. Furthermore, our method outperforms `LieConv` for most data sizes except the largest sizes where the errors are similar, suggesting that the attention framework more suited for this task.

Figure 5 shows the test error as function of the roll-out time step for a training data size of 10,000 (corresponding plots for other training data sizes are included in Appendix J.1). Here we show that the `LieTransformer` shows better generalisation than `LieConv` across all roll-out lengths, the error being low ($< 10^{-3}$) for 100 step-roll-outs even though we only train on 5-step roll-outs. We also include example trajectories of our model in Figure 6 (more examples can be found in the appendix, including ones where `LieConv` performs better than `LieTransformer`) illustrating the accuracy of our model on this task.

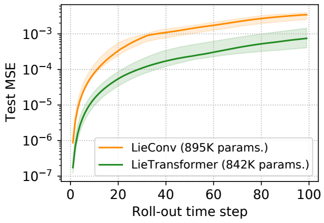

Figure 5. Test error vs. roll-out time step. Plots show median MSE and IQR across 10 random seeds.

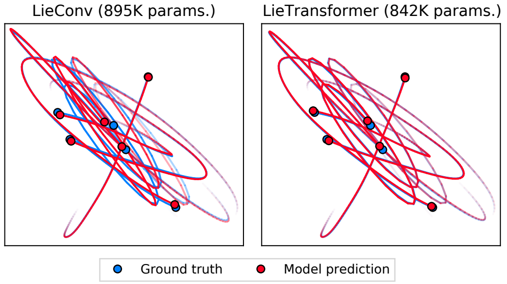

Figure 6. Example trajectory predictions on the spring dynamics task. LieTransformer closely follows the ground truth while LieConv diverges from the ground truth at later timesteps.

Lastly, Figure 7 compares `LieConv` and `LieTransformer` for different model sizes (number of parameters) and equivariance groups. We first note `LieTransformer` outperforms `LieConv` given a fixed model size and group. For $T(2)$-invariant models, our method benefits from a larger model, whereas `LieConv` deteriorates (`LieConv-T(2) (895K)` is their default architecture on this task). However, for both methods, the $SE(2)$-invariant models perform at par with or better than their $T(2)$-invariant counterparts despite having smaller model sizes. In particular, `LieTransformer-SE(2) (139K)` outperforms all other models in this comparison despite having the smallest number of parameters, which highlights the advantage of incorporating the correct task symmetries into the architecture and the attention framework. Overall, we have shown that our model is suitable for use in a neural ODE setting that requires equivariant drift functions.

6. Limitations and Future Work

From the algorithmic perspective, `LieTransformer` shares the weakness of `LieConv` in being memory-expensive ($O(|\hat{G}_f||\texttt{nbhd}_\eta|)$ memory cost (Appendix G) due to: 1. The lifting procedure that increases the number of inputs by $|\hat{H}|$, and 2. Quadratic complexity in the number of inputs from having to compute the kernel value at each pair of inputs. Although the first is a weakness shared by all lifting-based equivariant neural networks, the second can be addressed by incorporating works that study efficient variants of self-attention (Wang et al., 2020; Kitaev et al., 2020; Zaheer et al., 2020; Katharopoulos et al., 2020). An alternative is to incorporate information about pairs of inputs (such as bonding information for the QM9 task) as masking in self-attention (c.f. Appendix I.2).

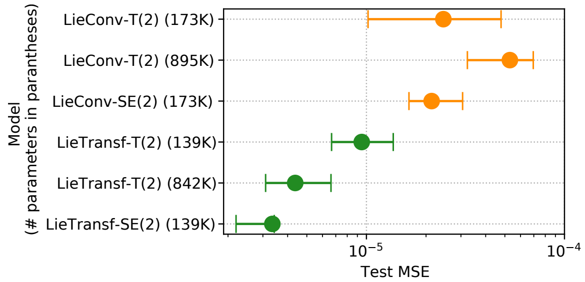

Figure 7. Median test MSE and IQR on 5-step trajectories, across 5 random seeds. Results for 100-step trajectories in Figure 8.

From the methodological perspective, a key weakness of the `LieTransformer` that is also shared with `LieConv` is its approximate equivariance due to MC estimation of the integral in `LieSelfAttention` for the case where H is infinite. The aforementioned directions for memory-efficiency can help to reduce the approximation error by allowing to use more lift samples ($|\hat{H}|$). Other directions include incorporating the notion of *steerability* (Cohen & Welling, 2017) to deal with vector fields in an equivariant manner (given inputs $(x_i, \mathtt{f}_i)$, the group acts non-trivially on $\mathtt{f}_i$ as well as x_i), and extending to non-homogeneous input spaces as outlined in Finzi et al. (2020).

ACKNOWLEDGEMENTS

The authors would like to thank Adam R.Kosiorek for setting up the initial codebase at the beginning of the project, and David W. Romero & Jean-Baptiste Cordonnier for useful discussions. Michael is supported by the EPSRC Centre for Doctoral Training in Modern Statistics and Statistical Machine Learning (EP/S023151/1). Charline acknowledges funding from the EPSRC grant agreement no. EP/N509711/1. Sheheryar wishes to acknowledge support from Aker Scholarship. Emilien acknowledges support of his PhD funding from Google DeepMind. Yee Whye Teh's research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071.

We would also like to thank the Python community (Van Rossum & Drake Jr, 1995; Oliphant, 2007) for developing the tools that enabled this work, including Pytorch (Paszke et al., 2017), NumPy (Oliphant, 2006; Walt et al., 2011; Harris et al., 2020), SciPy (Jones et al., 2001), and Matplotlib (Hunter, 2007).

References

Anderson, B., Hy, T. S., and Kondor, R. Cormorant: Covariant molecular neural networks. In *NeurIPS*, 2019.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bekkers, E. J. B-spline CNNs on Lie groups. In *ICLR*, 2020.

Bloem-Reddy, B. and Teh, Y. W. Probabilistic symmetries and invariant neural networks. *Journal of Machine Learning Research*, 21(90):1–61, 2020.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *ICML*, 2016.

Cohen, T. S. and Welling, M. Steerable CNNs. In *ICLR*, 2017.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical CNNs. In *ICLR*, 2018.

Cohen, T. S., Geiger, M., and Weiler, M. A general theory of equivariant CNNs on homogeneous spaces. In *NeurIPS*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning SO(3) equivariant representations with spherical CNNs. In *ECCV*, 2018.

Esteves, C., Makadia, A., and Daniilidis, K. Spin-weighted spherical CNNs. In *NeurIPS*, 2020.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to Lie groups on arbitrary continuous data. In *ICML*, 2020.

Fuchs, F. B., Worrall, D. E., Fischer, V., and Welling, M. SE(3)-Transformers: 3D roto-translation equivariant attention networks. In *NeurIPS*, 2020.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. 2017.

Graves, A. and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, 2014.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. In *NeurIPS*, 2019.

Hall, B. *Lie groups, Lie algebras, and representations: an elementary introduction*, volume 222. Springer, 2015.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

Hirn, M., Mallat, S., and Poilvert, N. Wavelet scattering regression of quantum chemical energies. *Multiscale Modeling & Simulation*, 15(2):827–863, 2017.

Hoogeboom, E., Peters, J. W., Cohen, T. S., and Welling, M. HexaConv. In *ICLR*, 2018.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Simon, I., Hawthorne, C., Shazeer, N., Dai, A. M., Hoffman, M. D., Dinculescu, M., and Eck, D. Music Transformer. In *ICLR*, 2019.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Jones, E., Oliphant, T., Peterson, P., et al. Scipy: Open source scientific tools for python. 2001.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. 2015.

Kitaev, N., Kaiser, Ł., and Levskaya, A. Reformer: The efficient transformer. In *ICLR*, 2020.

Klicpera, J., Groß, J., and Günnemann, S. Directional message passing for molecular graphs. In *ICLR*, 2019.

Klicpera, J., Giri, S., Margraf, J. T., and Günnemann, S. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115*, 2020.

Kondor, R. and Trivedi, S. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *ICML*, 2018.

Kondor, R., Lin, Z., and Trivedi, S. Clebsch–Gordan nets: a fully Fourier space spherical convolutional neural network. In *NeurIPS*, 2018.

Kosiorek, A., Sabour, S., Teh, Y. W., and Hinton, G. E. Stacked capsule autoencoders. In *NeurIPS*, 2019.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

Miller, B. K., Geiger, M., Smidt, T. E., and Noé, F. Relevance of rotationally equivariant convolutions for predicting molecular properties. *arXiv preprint arXiv:2008.08461*, 2020.

Noether, E. Invariant variation problems. *Zu Göttingen, Math-phys*, pp. 235–257, 1918.

Oliphant, T. E. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

Oliphant, T. E. Python for scientific computing. *Computing in Science & Engineering*, 9(3):10–20, 2007.

Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gulcehre, C., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M. M., Heess, N., and Hadsell, R. Stabilizing Transformers for reinforcement learning. In *ICML*, 2020.

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. Stand-alone self-attention in vision models. In *NeurIPS*. 2019a.

Parmar, N., Ramachandran, P., Vaswani, A., Bello, I., Levskaya, A., and Shlens, J. Stand-alone self-attention in vision models. In *NeurIPS*, 2019b.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

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

Romero, D. W. and Cordonnier, J.-B. Group equivariant stand-alone self-attention for vision. In *ICLR*, 2021.

Romero, D. W. and Hoogendoorn, M. Co-attentive equivariant neural networks: Focusing equivariance on transformations co-occurring in data. In *ICLR*, 2020.

Romero, D. W., Bekkers, E. J., Tomczak, J. M., and Hoogendoorn, M. Attentive group equivariant convolutional networks. In *ICML*, 2020.

Ruddigkeit, L., van Deursen, R., Blum, L. C., and Reymond, J.-L. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *J. Chem. Inf. Model.*, 52(11):2864–2875, November 2012. ISSN 1549-9596. doi: 10.1021/ci300415d. URL `https://doi.org/10.1021/ci300415d`. Publisher: American Chemical Society.

Sanchez-Gonzalez, A., Bapst, V., Cranmer, K., and Battaglia, P. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Schütt, K., Kindermans, P.-J., Felix, H. E. S., Chmiela, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions. In *NeurIPS*, 2017.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3D point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Tsai, Y.-H. H., Bai, S., Yamada, M., Morency, L.-P., and Salakhutdinov, R. Transformer dissection: An unified understanding for Transformer's attention via the lens of kernel. In *EMNLP-IJCNLP*, 2019.

Van Rossum, G. and Drake Jr, F. L. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.

Wang, S., Li, B., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Wang, X., Girshick, R., Gupta, A., and He, K. Non-local neural networks. In *CVPR*, 2018.

Weiler, M. and Cesa, G. General E(2)-equivariant steerable CNNs. In *NeurIPS*, 2019.

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. 3D steerable CNNs: Learning rotationally equivariant features in volumetric data. In *NeurIPS*, 2018a.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant CNNs. In *CVPR*, 2018b.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *CVPR*, 2017.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

Zhang, H., Goodfellow, I., Metaxas, D., and Odena, A. Self-attention Generative Adversarial Networks. In *ICML*, 2019.

Zhong, Y. D., Dey, B., and Chakraborty, A. Symplectic ode-net: Learning hamiltonian dynamics with control. 2020.