Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data

Marc Finzi [1] **Samuel Stanton** [1] **Pavel Izmailov** [1] **Andrew Gordon Wilson** [1]

Abstract

The translation equivariance of convolutional layers enables convolutional neural networks to generalize well on image problems. While translation equivariance provides a powerful inductive bias for images, we often additionally desire equivariance to other transformations, such as rotations, especially for non-image data. We propose a general method to construct a convolutional layer that is equivariant to transformations from any specified Lie group with a surjective exponential map. Incorporating equivariance to a new group requires implementing only the group exponential and logarithm maps, enabling rapid prototyping. Showcasing the simplicity and generality of our method, we apply the same model architecture to images, ball-and-stick molecular data, and Hamiltonian dynamical systems. For Hamiltonian systems, the equivariance of our models is especially impactful, leading to exact conservation of linear and angular momentum.

1. Introduction

Symmetry pervades the natural world. The same law of gravitation governs a game of catch, the orbits of our planets, and the formation of galaxies. It is precisely because of the order of the universe that we can hope to understand it. Once we started to understand the symmetries inherent in physical laws, we could predict behavior in galaxies billions of light-years away by studying our own local region of time and space. For statistical models to achieve their full potential, it is essential to incorporate our knowledge of naturally occurring symmetries into the design of algorithms and architectures. An example of this principle is the translation equivariance of convolutional layers in neural networks (LeCun et al., 1995): when an input (e.g. an image) is translated, the output of a convolutional layer is translated in the same way.

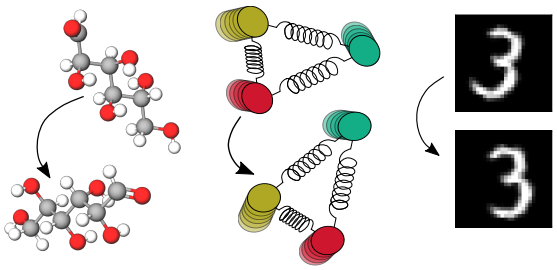


Figure 1. Many modalities of spatial data do not lie on a grid, but still possess important symmetries. We propose a single model to learn from continuous spatial data that can be specialized to respect a given continuous symmetry group.

Group theory provides a mechanism to reason about symmetry and equivariance. Convolutional layers are equivariant to translations, and are a special case of group convolution. A group convolution is a general linear transformation equivariant to a given group, used in group equivariant convolutional networks (Cohen and Welling, 2016a).

In this paper, we develop a general framework for equivariant models on arbitrary continuous (spatial) data represented as coordinates and values $\{(x_i, f_i)\}_{i=1}^N$. Spatial data is a broad category, including ball-and-stick representations of molecules, the coordinates of a dynamical system, and images (shown in Figure 1). When the inputs or group elements lie on a grid (e.g., image data) one can simply enumerate the values of the convolutional kernel at each group element. But in order to extend to continuous data, we define the convolutional kernel as a continuous function on the group parameterized by a neural network.

We consider the large class of continuous groups known as Lie groups. In most cases, Lie groups can be parameterized in terms of a vector space of infinitesimal generators (the Lie algebra) via the logarithm and exponential maps. Many useful transformations are Lie groups, including translations, rotations, and scalings. We propose LieConv, a convolutional layer that can be made equivariant to a given Lie group by defining $\exp$ and $\log$ maps. We demonstrate the

[1]New York University. Correspondence to: Marc Finzi <maf820@nyu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

expressivity and generality of LieConv with experiments on images, molecular data, and dynamical systems. We emphasize that we use the *same network architecture* for all transformation groups and data types. LieConv achieves state-of-the-art performance in these domains, even compared to domain-specific architectures. In short, the main contributions of this work are as follows:

- We propose LieConv, a new convolutional layer equivariant to transformations from Lie groups. Models composed with LieConv layers can be applied to non-homogeneous spaces and arbitrary spatial data.
- We evaluate LieConv on the image classification benchmark dataset rotMNIST (Larochelle et al., 2007), and the regression benchmark dataset QM9 (Blum and Reymond, 2009; Rupp et al., 2012). LieConv outperforms state-of-the-art methods on some tasks in QM9, and in all cases achieves competitive results.
- We apply LieConv to modeling the Hamiltonian of physical systems, where equivariance corresponds to the preservation of physical quantities (energy, angular momentum, etc.). LieConv outperforms state-of-the-art methods for the modeling of dynamical systems.

We make code available at
https://github.com/mfinzi/LieConv

2. Related Work

One approach to constructing equivariant CNNs, first introduced in Cohen and Welling (2016a), is to use standard convolutional kernels and transform them or the feature maps for each of the elements in the group. For discrete groups this approach leads to exact equivariance and uses the so-called regular representation of the group (Cohen et al., 2019). This approach is easy to implement, and has also been used when the feature maps are vector fields (Zhou et al., 2017; Marcos et al., 2017), and with other representations (Cohen and Welling, 2016b), but only on image data where locations are discrete and the group cardinality is small. This approach has the disadvantage that the computation grows quickly with the size of the group, and some groups like 3D rotations cannot be easily discretized onto a lattice that is also a subgroup.

Another approach, drawing on harmonic analysis, finds a basis of equivariant functions and parametrizes convolutional kernels in that basis (Worrall et al., 2017; Weiler and Cesa, 2019; Jacobsen et al., 2017). These kernels can be used to construct networks that are exactly equivariant to continuous groups. While the approach has been applied on general data types like spherical images (Esteves et al., 2018; Cohen et al., 2018), voxel data (Weiler et al., 2018), and point clouds (Thomas et al., 2018; Anderson et al., 2019), the requirement of working out the representation theory for the group can be cumbersome and is limited to compact groups. Our approach reduces the amount of work to implement equivariance to a new group, enabling rapid prototyping.

There is also work applying Lie group theory to deep neural networks. Huang et al. (2017) define a network where the intermediate activations of the network are 3D rotations representing skeletal poses and embed elements into the Lie algebra using the $\log$ map. Bekkers (2019) use the $\log$ map to express an equivariant convolution kernel through the use of B-splines, which they evaluate on a grid and apply to image problems. While similar in motivation, their method is not readily applicable to point data and can only be used when the equivariance group acts transitively on the input space. Both of these issues are addressed by our work.

3. Background

3.1. Equivariance

A mapping $h(\cdot)$ is equivariant to a set of transformations G if when we apply any transformation g to the input of h, the output is also transformed by g. The most common example of equivariance in deep learning is the translation equivariance of convolutional layers: if we translate the input image by an integer number of pixels in x and y, the output is also translated by the same amount (ignoring the regions close to the boundary of the image). Formally, if $h : A \to A$, and G is a set of transformations acting on A, we say h is equivariant to G if $\forall a \in A$, $\forall g \in G$,

$$h(ga) = gh(a). \tag{1}$$

The continuous convolution of a function $f : \mathbb{R} \to \mathbb{R}$ with the kernel $k : \mathbb{R} \to \mathbb{R}$ is equivariant to translations in the sense that $L_t(k * f) = k * L_t f$ where L_t translates the function by t: $L_t f(x) = f(x - t)$.

It is easy to construct invariant functions, where transformations on the input do not affect the output, by simply discarding information. Strict invariance unnecessarily limits the expressive power by discarding relevant information, and instead it is necessary to use equivariant transformations that preserve the information.

3.2. Groups of Transformations and Lie Groups

Many important sets of transformations form a group. To form a group the set must be closed under composition, include an identity transformation, each element must have an inverse, and composition must be associative. The set of 2D rotations, SO(2), is a simple and instructive example. Composing two rotations r_1 and r_2, yields another rotation $r = r_2 \circ r_1$. There exists an identity $\mathrm{id} \in G$ that maps every point in $\mathbb{R}^2$ to itself (i.e., rotation by a zero angle).

And for every rotation r, there exists an inverse rotation r^{-1} such that $r \circ r^{-1} = r^{-1} \circ r = \text{id}$. Finally, the composition of rotations is an *associative* operation: $(r_1 \circ r_2) \circ r_3 = r_1 \circ (r_2 \circ r_3)$. Satisfying these conditions, SO(2) is indeed a group.

We can also adopt a more familiar view of SO(2) in terms of angles, where a rotation matrix $R : \mathbb{R} \to \mathbb{R}^{2\times 2}$ is parametrized as $R(\theta) = \exp(J\theta)$. J is the antisymmetric matrix $J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ (an infinitesimal generator of the group) and $\exp$ is the matrix exponential. Note that θ is totally unconstrained. Using $R(\theta)$ we can add and subtract rotations. Given θ_1, θ_2 we can compute $R(\theta_1)^{-1}R(\theta_2) = \exp(-J\theta_1 + J\theta_2) = R(\theta_2 - \theta_1)$. $R(\theta) = \exp(J\theta)$ is an example of the Lie algebra parametrization of a group, and SO(2) forms a Lie group.

More generally, a Lie group is a group whose elements form a smooth manifold. Since G is not necessarily a vector space, we cannot add or subtract group elements. However, the Lie algebra of G, the tangent space at the identity, $\mathfrak{g} = T_{\text{id}}G$, *is* a vector space and can be understood informally as a space of infinitesimal transformations from the group. As a vector space, one can readily expand elements in a basis $A = \sum_k a^k e_k$ and use the components for calculations.

The exponential map $\exp : \mathfrak{g} \to G$ gives a mapping from the Lie algebra to the Lie group, converting infinitesimal transformations to group elements. In many cases, the image of the exponential map covers the group, and an inverse mapping $\log : G \to \mathfrak{g}$ can be defined. For matrix groups the $\exp$ map coincides with the matrix exponential ($\exp(A) = I + A + A^2/2! + \ldots$), and the $\log$ map with the matrix logarithm. Matrix groups are particularly amenable to our method because in many cases the $\exp$ and $\log$ maps can be computed in closed form. For example, there are analytic solutions for the translation group T(d), the $3D$ rotation group SO(3), the translation and rotation group SE(d) for $d = 2, 3$, the rotation-scale group $\mathbb{R}^* \times$ SO(2), and many others (Eade, 2014). In the event that an analytic solution is not available there are reliable numerical methods at our disposal (Moler and Van Loan, 2003).

3.3. Group Convolutions

Adopting the convention of left equivariance, one can define a *group convolution* between two functions on the group, which generalizes the translation equivariance of convolution to other groups:

Definition 1. *Let $k, f : G \to \mathbb{R}$, and $\mu(\cdot)$ be the Haar measure on G. For any $u \in G$, the convolution of k and f on G at u is given by*

$$h(u) = (k * f)(u) = \int_G k(v^{-1}u) f(v) d\mu(v). \quad (2)$$

(Kondor and Trivedi, 2018; Cohen et al., 2019)

3.4. PointConv Trick

In order to extend learnable convolution layers to point clouds, not having the regular grid structure in images, Dai et al. (2017), Simonovsky and Komodakis (2017), and Wu et al. (2019)go back to the continuous definition of a convolution for a single channel between a learned function (convolutional filter) $k_\theta(\cdot) : \mathbb{R}^d \to \mathbb{R}^{c_{out} \times c_{in}}$ and an input feature map $f(\cdot) : \mathbb{R}^d \to \mathbb{R}^{c_{in}}$ yielding the function $h(\cdot) : \mathbb{R}^d \to \mathbb{R}^{c_{out}}$,

$$h(x) = (k_\theta * f)(x) = \int k_\theta(x - y) f(y) dy. \quad (3)$$

We approximate the integral using a discretization:

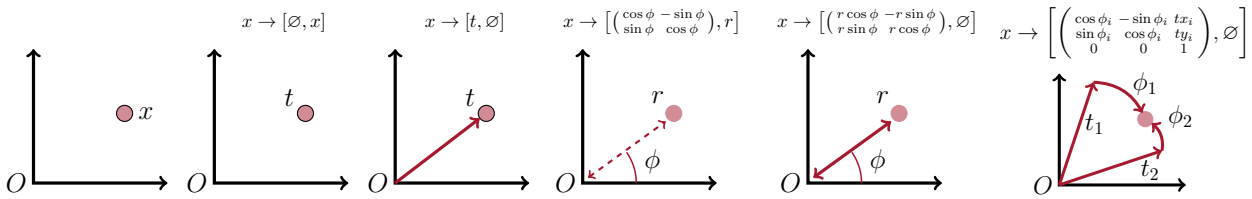
$$h(x_i) = (V/n) \sum_j k_\theta(x_i - x_j) f(x_j)\,. \quad (4)$$

Here V is the volume of the space integrated over and n is the number of quadrature points. In a 3×3 convolutional layer for images, where points fall on a uniform square grid, the filter k_θ has independent parameters for each of the inputs $(-1, -1), (-1, 0), \ldots, (1, 1)$. In order to accommodate points that are not on a regular grid, k_θ can be parametrized as a small neural network, mapping input offsets to filter matrices, explored with MLPs in Simonovsky and Komodakis (2017). The compute and memory costs has severely limited this approach, for typical CIFAR-10 images with $\text{batchsize} = 32, N = 32 \times 32, c_{in} = c_{out} = 256, n = 3 \times 3$, evaluating a single layer requires computing 20 billion values for k.

In PointConv, Wu et al. (2019) develop a trick where clever reordering of the computation cuts memory and computational requirements by ~ 2 orders of magnitude, allowing them to scale to the point cloud classification, segmentation datasets ModelNet40 and ShapeNet, and the image dataset CIFAR-10. We review and generalize the Efficient-PointConv trick in Appendix A.1, which we will use to accelerate our method.

4. Convolutional Layers on Lie Groups

We now introduce LieConv, a new convolutional layer that can be made equivariant to a given Lie group. Models with LieConv layers can act on arbitrary collections of coordinates and values $\{(x_i, f_i)\}_{i=1}^N$, for $x_i \in \mathcal{X}$ and $f_i \in V$ where V is a vector space. The domain $\mathcal{X}$ is usually a low dimensional domain like $\mathbb{R}^2$ or $\mathbb{R}^3$ such as for molecules, point clouds, the configurations of a mechanical system, images, time series, videos, geostatistics, and other kinds of spatial data.



(a) Data (b) Trivial (c) $T(2)$ (d) $SO(2)$ (e) $\mathbb{R}^* \times SO(2)$ (f) $SE(2)$

Figure 2. Visualization of the lifting procedure. Panel (a) shows a point x in the original input space $\mathcal{X}$. In panels **(b)–(f)** we illustrate the lifted embeddings for different groups in the form $[u, q]$, where $u \in G$ is an element of the group and $q \in \mathcal{X}/G$ identifies the orbit (see Section 4.5). For SE(2) the lifting is multi-valued.

We begin with a high-level overview of the method. In Section 4.1 we discuss transforming raw inputs x_i into group elements u_i on which we can perform group convolution. We refer to this process as *lifting*. Section 4.2 addresses the irregular and varied arrangements of group elements that result from lifting arbitrary continuous input data by parametrizing the convolutional kernel k as a neural network. In Section 4.3, we show how to enforce the locality of the kernel by defining an invariant distance on the group. In Section 4.4, we define a Monte Carlo estimator for the group convolution integral in Eq. (2) and show that this estimator is equivariant in distribution. In Section 4.5, we extend the procedure to cases where the group does not act transitively on the input space (when we cannot map any point to any other point with a transformation from the group). Additionally, in Appendix A.2, we show that our method generalizes coordinate transform equivariance when G is Abelian. At the end of Section 4.5 we provide a concise algorithmic description of the lifting procedure and our new convolution layer.

4.1. Lifting from $\mathcal{X}$ to G

If $\mathcal{X}$ is a homogeneous space of G, then every two elements in $\mathcal{X}$ are connected by an element in G, and one can lift elements by simply picking an origin o and defining $\text{Lift}(x) = \{u \in G : uo = x\}$: all elements in the group that map the origin to x. This procedure enables lifting tuples of coordinates and features $\{(x_i, f_i)\}_{i=1}^N \to \{(u_{ik}, f_i)\}_{i=1,k=1}^{N,K}$, with up to K group elements for each input.[1] To find all the elements $\{u \in G : uo = x\}$, one simply needs to find one element u_x and use the elements in the stabilizer of the origin $H = \{h \in G : ho = o\}$, to generate the rest with $\text{Lift}(x) = \{u_x h \text{ for } h \in H\}$. For continuous groups the stabilizer may be infinite, and in these cases we sample uniformly using the Haar measure μ which is described in Appendix C.2. We visualize the lifting procedure for different groups in Figure 2.

[1] When $f_i = f(x_i)$, lifting in this way is equivalent to defining $f^\uparrow(u) = f(uo)$ as in Kondor and Trivedi (2018).

4.2. Parameterization of the Kernel

The conventional method for implementing an equivariant convolutional network (Cohen and Welling, 2016a) requires enumerating the values of $k(\cdot)$ over the elements of the group, with separate parameters for each element. This procedure is infeasible for irregularly sampled data and problematic even for a discretization because there is no generalization between different group elements. Instead of having a discrete mapping from each group element to the kernel values, we parametrize the convolutional kernel as a continuous function k_θ using a fully connected neural network with Swish activations, varying smoothly over the elements in the Lie group.

However, as neural networks are best suited to learn on euclidean data and G does not form a vector space, we propose to model k by mapping onto the Lie Algebra $\mathfrak{g}$, which *is* a vector space, and expanding in a basis for the space. To do so, we restrict our attention in this paper to Lie groups whose exponential maps are surjective, where every element has a logarithm. This means defining $k_\theta(u) = (k \circ \exp)_\theta(\log u)$, where $\tilde{k}_\theta = (k \circ \exp)_\theta$ is the function parametrized by an MLP, $\tilde{k}_\theta : \mathfrak{g} \to \mathbb{R}^{c_{out} \times c_{in}}$. Surjectivity of the exp map guarantees that $\exp \circ \log = \text{id}$, although not in the other order.

4.3. Enforcing Locality

Important both to the inductive biases of convolutional neural networks and their computational efficiency is the fact that convolutional filters are local, $k_\theta(u_i - u_j) = 0$ for $\|u_i - u_j\| > r$. In order to quantify locality on matrix groups, we introduce the function:

$$d(u, v) := \|\log(u^{-1}v)\|_F, \tag{5}$$

where log is the matrix logarithm, and F is the Frobenius norm. The function is left invariant, since $d(wu, wv) = \|\log(u^{-1}w^{-1}wv)\|_F = d(u, v)$, and is a semi-metric (it

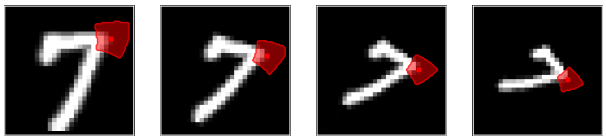


Figure 3. A visualization of the local neighborhood for $\mathbb{R}^* \times \mathrm{SO}(2)$, in terms of the points in the input space. For the computation of h at the point in orange, elements are sampled from colored region. Notice that the same points enter the calculation when the image is transformed by a rotation and scaling. We visualize the neighborhoods for other groups in Appendix C.6.

does not necessarily satisfy the triangle inequality). In Appendix A.3 we show the conditions under which $d(u, v)$ is additionally the distance along the geodesic connecting u, v, a generalization of the well known formula for the geodesic distance between rotations $\|\log(R_1^T R_2)\|_F$ (Kuffner, 2004).

To enforce that our learned convolutional filter k is local, we can use our definition of distance to only evaluate the sum for $d(u, v) < r$, implicitly setting $k_\theta(v^{-1}u) = 0$ outside a local neighborhood $\mathrm{nbhd}(u) = \{v : d(u, v) \leq r\}$,

$$h(u) = \int_{v \in \mathrm{nbhd}(u)} k_\theta(v^{-1}u) f(v) d\mu(v). \tag{6}$$

This restriction to a local neighborhood does not break equivariance precisely because $d(\cdot, \cdot)$ is left invariant. Since $d(u, v) = d(v^{-1}u, \mathrm{id})$ this restriction is equivalent to multiplying by the indicator function $k_\theta(v^{-1}u) \to k_\theta(v^{-1}u)\mathbb{1}_{[d(v^{-1}u,\mathrm{id}) \leq r]}$ which depends only on $v^{-1}u$. Note that equivariance *would* have been broken if we used neighborhoods that depend on fixed regions in the input space like the square 3×3 region. Figure 3 shows what these neighborhoods look like in terms of the input space.

4.4. Discretization of the Integral

Assuming that we have a collection of quadrature points $\{v_j\}_{j=1}^N$ as input and the function $f_j = f(v_j)$ evaluated at these points, we can judiciously choose to evaluate the convolution at another set of group elements $\{u_i\}_{i=1}^N$, so as to have a set of quadrature points to approximate an integral in a subsequent layer. Because we have restricted the integral (6) to the compact neighbourhood $\mathrm{nbhd}(u)$, we can define a proper sampling distribution $\mu|_{\mathrm{nbhd}(u)}$ to estimate the integral, unlike for the possibly unbounded G. Computing the outputs only at these target points, we use the Monte Carlo estimator for (1) as

$$h(u_i) = (k \hat{*} f)(u_i) = \frac{1}{n_i} \sum_{j \in \mathrm{nbhd}(i)} k(v_j^{-1}u_i) f(v_j), \tag{7}$$

where $n_i = |\mathrm{nbhd}(i)|$ the number of points sampled in each neighborhood.

For $v_j \sim \mu|_{\mathrm{nbhd}(u)}(\cdot)$, the Monte Carlo estimator is equivariant (in distribution).

Proof: Recalling that we can absorb the local neighborhood into the definition of k_θ using an indicator function, we have

$$\begin{aligned}(k \hat{*} L_w f)(u_i) &= (1/n_i) \sum_j k(v_j^{-1}u_i) f(w^{-1}v_j) \\ &= (1/n_i) \sum_j k(\tilde{v}_j^{-1}w^{-1}u_i) f(\tilde{v}_j) \\ &\overset{d}{=} (k \hat{*} f)(w^{-1}u_i) = L_w(k \hat{*} f)(u_i).\end{aligned}$$

Here $\tilde{v}_j := wv_j$, and the last line follows from the fact that the random variables $wv_j \overset{d}{=} v_j$ are equal in distribution because they are sampled from the Haar measure with property $d\mu(wv) = d\mu(v)$. The equivariance also holds deterministically when the sampling locations are transformed along with the function $v_j \to wv_j$. Now that we have the discretization $h_i = (1/n_i) \sum_{j \in \mathrm{nbhd}(i)} \tilde{k}_\theta(\log(v_j^{-1}u_i)) f_i$, we can accelerate this computation using the Efficient-PointConv trick, with the argument of $a_{ij} = \log(v_j^{-1}u_i)$ for the MLP. See Appendix A.1 for more details. Note that we can also apply this discretization of the convolution when the inputs are not functions $f_i = f(x_i)$, but simply coordinates and values $\{(x_i, f_i)\}_{i=1}^N$, and the mapping $\{(u_i, f_i)\}_{i=1}^N \to \{(u_i, h_i)\}_{i=1}^N$ is still equivariant, which we also demonstrate empirically in Table B.1. We also detail two methods for equivariantly subsampling the elements to further reduce the cost in Appendix A.4.

4.5. More Than One Orbit?

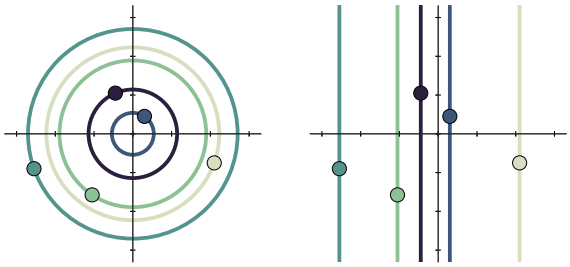


Figure 4. Orbits of SO(2) and $\mathrm{T}(1)_y$ containing input points in $\mathbb{R}^2$. Unlike T(2) and SE(2), not all points are not contained in a single orbit of these small groups.

In this paper, we consider groups both large and small, and we require the ability to enable or disable equivariances like translations. To achieve this functionality, we need to go beyond the usual setting of homogeneous spaces considered in the literature, where every pair of elements in $\mathcal{X}$ are related by an element in G. Instead, we consider the quotient space $Q = \mathcal{X}/G$, consisting of the distinct orbits of G in

$\mathcal{X}$ (visualized in Figure 4).[2] Each of these orbits $q \in Q$ is a homogeneous space of the group, and when $\mathcal{X}$ is a homogeneous space of G then there is only a single orbit. But in general, there will be many distinct orbits, and lifting should preserve the information on which orbit each point is on.

Since the most general equivariant mappings will use this orbit information, throughout the network the space of elements should not be G but rather $G \times \mathcal{X}/G$, and $x \in \mathcal{X}$ is lifted to the tuples (u, q) for $u \in G$ and $q \in Q$. This mapping may be one-to-one or one-to-many depending on the size of H, but will preserve the information in x as $uo_q = x$ where o_q is the chosen origin for each orbit. General equivariant linear transforms can depend on both the input and output orbit, and equivariance only constrains the dependence on group elements and not the orbits.

When the space of orbits Q is continuous we can write the equivariant integral transform as

$$h(u, q) = \int_{G,Q} k(v^{-1}u, q, q')f(v, q')d\mu(v)dq'. \quad (8)$$

When G is the trivial group $\{\text{id}\}$, this equation simplifies to the integral transform $h(x) = \int k(x, x')f(x')dx'$ where each element in $\mathcal{X}$ is in its own orbit.

In general, even if $\mathcal{X}$ is a smooth manifold and G is a Lie group it is not guaranteed that $\mathcal{X}/G$ is a manifold (Kono and Ishitoya, 1987). However in practice this is not an issue as we will only have a finite number of orbits present in the data. All we need is an invertible way of embedding the orbit information into a vector space to be fed into k_θ. One option is to use an embedding of the orbit origin o_q, or simply find enough invariants of the group to identify the orbit. To give a few examples:

1. $\mathcal{X} = \mathbb{R}^d$ and $G = \text{SO}(d)$: $\text{Embed}(q(x)) = \|x\|$
2. $\mathcal{X} = \mathbb{R}^d$ and $G = \mathbb{R}^*$: $\text{Embed}(q(x)) = \dfrac{x}{\|x\|}$
3. $\mathcal{X} = \mathbb{R}^d$ and $G = \text{T}(k)$: $\text{Embed}(q(x)) = x_{[k+1:d]}$

Discretizing (8) as we did in (7), we get

$$h_i = \frac{1}{n_i} \sum_{j \in \text{nbhd(i)}} \tilde{k}_\theta(\log(v_j^{-1}u_i), q_i, q_j)f_j, \quad (9)$$

which again can be accelerated with the Efficient-PointConv trick by feeding in $a_{ij} = \text{Concat}([\log(v_j^{-1}u_i), q_i, q_j])$ as input to the MLP. If we want the filter to be local over orbits also, we can extend the distance $d((u_i, q_i), (v_j, q_j))^2 = d(u_i, v_j)^2 + \alpha\|q_i - q_j\|^2$, which need not be invariant to transformations on q. To the best of our knowledge, we are the first to systematically address equivariances of this kind, where $\mathcal{X}$ is not a homogeneous space of G.

To recap, Algorithms 1 and 2 give a concise overview of our lifting procedure and our new convolution layer respectively. Please consult Appendix C.1 for additional implementation details.

Algorithm 1 Lifting from $\mathcal{X}$ to $G \times \mathcal{X}/G$

Inputs: spatial data $\{(x_i, f_i)\}_{i=1}^N$ ($x_i \in \mathcal{X}$, $f_i \in \mathbb{R}^{c_{in}}$).
Returns: matrix-orbit-value tuples $\{(u_j, q_j, f_j)\}_{j=1}^{NK}$.
For each orbit $q \in \mathcal{X}/G$, choose an origin o_q.
For each o_q, compute its stabilizer H_q.
for $i = 1, \dots, N$ **do**
  Find the orbit $q_i \in \mathcal{X}/G$, s.t. $x_i \in q_i$.
  Sample $\{v_j\}_{j=1}^K$, where $v_j \sim \mu(H_{q_i})$ (see C.2).
  Compute an element $u_i \in G$ s.t. $u_i o_q = x_i$.
  $Z_i = \{(u_i v_j, q_i, f_i)\}_{j=1}^K$.
end
return Z

Algorithm 2 The Lie Group Convolution Layer

Inputs: matrix-orbit-value tuples $\{(u_j, q_j, f_j)\}_{j=1}^m$
Returns: convolved matrix-orbit-values $\{(u_i, q_i, h_i)\}_{i=1}^m$
for $i = 1, \dots, m$ **do**
  $u_i^{-1} = \exp(-\log(u_i))$.
  $\text{nbhd}_i = \{j : d((u_i, q_i), (u_j, q_j)) < r)\}$.
  for $j = 1, \dots, m$ **do**
    $a_{ij} = \text{Concat}([\log(u_i^{-1}u_j), q_i, q_j])$.
  end
  $h_i = (1/n_i) \sum_{j \in \text{nbhd}_i} k_\theta(a_{ij})f_j$ (see A.1).
end
return $(\mathbf{u}, \mathbf{q}, \mathbf{h})$

5. Applications to Image and Molecular Data

First, we evaluate LieConv on two types of problems: classification on image data and regression on molecular data. With LieConv as the convolution layers, we implement a bottleneck ResNet architecture with a final global pooling layer (Figure 5). For a detailed architecture description, see Appendix C.3. We use the same model architecture for all tasks and achieve performance competitive with task-specific specialized methods.

[2] When $\mathcal{X}$ is a homogeneous space and the quantity of interest is the quotient with the stabilizer of the origin H: $G/H \simeq \mathcal{X}$, which has been examined extensively in the literature. Here we concerned with the separate quotient space $Q = \mathcal{X}/G$, relevant when $\mathcal{X}$ is not a homogeneous space.

Table 1. Classification Error (%) on RotMNIST dataset for LieConv with different group equivariances and baselines: G-CNN (Cohen and Welling, 2016a), H-Net (Worrall et al., 2017), ORN (Zhou et al., 2017), TI-Pooling (Laptev et al., 2016), RotEqNet (Marcos et al., 2017), E(2)-Steerable CNNs (Weiler and Cesa, 2019) .

Baseline Methods						LieConv (Ours)					
G-CNN	H-NET	ORN	TI-Pooling	RotEqNet	E(2)-Steerable	Trivial	$T(1)_y$	T(2)	SO(2)	SO(2)$\times\mathbb{R}^*$	SE(2)
2.28	1.69	1.54	1.2	1.09	**0.68**	1.58	1.49	1.44	1.42	1.27	**1.24**

Table 2. QM9 Molecular Property Mean Absolute Error

Task	α	$\Delta\varepsilon$	$\varepsilon_{\text{HOMO}}$	$\varepsilon_{\text{LUMO}}$	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
NMP	.092	69	43	38	**.030**	.040	19	17	.180	20	20	**1.500**
SchNet	.235	63	41	34	.033	.033	**14**	**14**	**.073**	**19**	**14**	1.700
Cormorant	.085	61	34	38	.038	**.026**	20	21	.961	21	22	2.027
LieConv(T3)	**.084**	**49**	**30**	**25**	.032	.038	22	24	.800	**19**	19	2.280

5.1. Image Equivariance Benchmark

The RotMNIST dataset consists of 12k randomly rotated MNIST digits with rotations sampled uniformly from SO(2), separated into 10k for training and 2k for validation. This commonly used dataset has been a standard benchmark for equivariant CNNs on image data. To apply LieConv to image data we interpret each input image as a collection of $N = 28 \times 28$ points on $\mathcal{X} = \mathbb{R}^2$ with associated binary values: $\{x_i, f(x_i)\}_{i=1}^{784}$ to which we apply a circular center crop. We note that LieConv is broadly targeting generic spatial data, and more practical equivariant methods exist specialized to images (e.g. Weiler and Cesa (2019)). However, as we demonstrate in Table 1, we are able to easily incorporate equivariance to a variety of different groups without changes to the method or the architecture of the network, while achieving performance competitive with methods that are not applicable beyond image data.

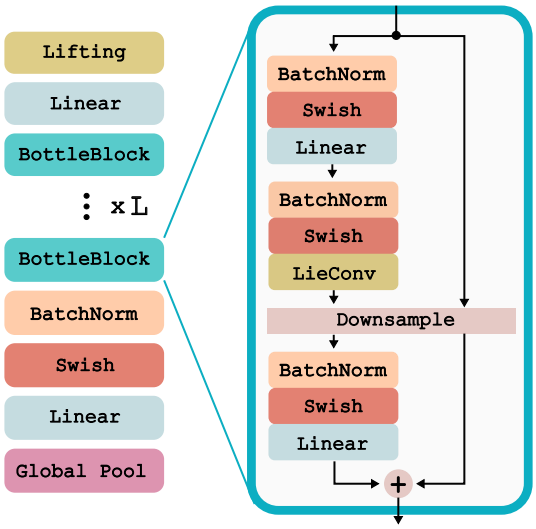


Figure 5. A visual overview of the LieConv model architecture, which is composed of L LieConv bottleneck blocks that couple the values at different group elements together. The BottleBlock is a residual block with a LieConv layer between two linear layers.

5.2. Molecular Data

Now we apply LieConv to the QM9 molecular property learning task (Wu et al., 2018). The QM9 regression dataset consists of small inorganic molecules encoded as a collection of 3D spatial coordinates for each of the atoms, and their atomic charges. The labels consist of various properties of the molecules such as heat capacity. This is a challenging task as there is no canonical origin or orientation for each molecule, and the target distribution is invariant to E(3) (translation, rotation, and reflection) transformations of the coordinates. Successful models must generalize across different spatial locations and orientations.

We first perform an ablation study on the `Homo` problem of predicting the energy of the highest occupied molecular orbital for the molecules. We apply LieConv with different equivariance groups, combined with SO(3) data augmentation. The results are reported in Table 5.2. Of the three groups, our SE(3) network performs the best. We then apply T(3)-equivariant LieConv layers to the full range of tasks in the QM9 dataset and report the results in Table 2. We perform competitively with state-of-the-art methods (Gilmer et al., 2017; Schütt et al., 2018; Anderson et al., 2019), with lowest MAE on several of the tasks. See B.1 for a demonstration of the equivariance property and efficiency with limited data.

Cormorant	Trivial	SO(3)	T(3)	SE(3)
34	31.7	65.4	29.6	**26.8**

Table 3. LieConv performance (Mean Absolute Error in meV) for different groups on the `HOMO` regression problem.

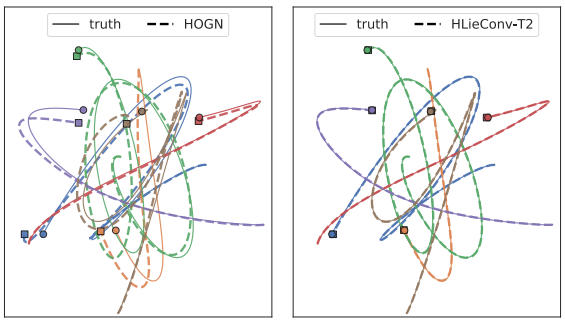


Figure 6. A qualitative example of the trajectory predictions over 100 time steps on the 2D spring problem given a set of initial conditions. We see that HLieConv (right) yields predictions that are accurate over a longer time than HOGN (left), a SOTA architecture for modeling interacting physical systems.

6. Modeling Dynamical Systems

Accurate transition models for macroscopic physical systems are critical components in control systems (Lenz et al., 2015; Kamthe and Deisenroth, 2017; Chua et al., 2018) and data-efficient reinforcement learning algorithms (Nagabandi et al., 2018; Janner et al., 2019). In this section we show how to enforce conservation of quantities such as linear and angular momentum in the modeling of Hamiltonian systems through LieConv symmetries.

6.1. Predicting Trajectories with Hamiltonian Mechanics

For dynamical systems, the equations of motion can be written in terms of the state $\mathbf{z}$ and time t: $\dot{\mathbf{z}} = F(\mathbf{z}, t)$. Many physically occurring systems have Hamiltonian structure, meaning that the state can be split into generalized coordinates and momenta $\mathbf{z} = (\mathbf{q}, \mathbf{p})$, and the dynamics can be written as

$$\frac{d\,\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial\,\mathbf{p}} \qquad \frac{d\,\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial\,\mathbf{q}} \tag{10}$$

for some choice of scalar Hamiltonian $\mathcal{H}(\mathbf{q}, \mathbf{p}, t)$. $\mathcal{H}$ is often the total energy of the system, and can sometimes be split into kinetic and potential energy terms $\mathcal{H}(\mathbf{q}, \mathbf{p}) = K(\mathbf{p}) + V(\mathbf{q})$. The dynamics can also be written compactly as $\dot{\mathbf{z}} = J\nabla_{\mathbf{z}}\mathcal{H}$ for $J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix}$.

As shown in Greydanus et al. (2019), a neural network parametrizing $\hat{\mathcal{H}}_\theta(\mathbf{z}, t)$ can be learned directly from trajectory data, providing substantial benefits in generalization over directly modeling $F_\theta(\mathbf{z}, t)$, and with better energy conservation. We follow the approach of Sanchez-Gonzalez et al. (2019) and Zhong et al. (2019). Given an initial condition $\mathbf{z}_0$ and $F_\theta(\mathbf{z}, t) = J\nabla_{\mathbf{z}}\hat{\mathcal{H}}_\theta$, we employ a twice-differentiable model architecture and a differentiable ODE solver (Chen et al., 2018) to compute predicted states $(\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_T) = \text{ODESolve}(\mathbf{z}_0, F_\theta, (t_1, t_2, ..., t_T))$. The parameters of the Hamiltonian model $\hat{\mathcal{H}}_\theta$ can be trained directly through the $L2$ loss,

$$L(\theta) = \frac{1}{T}\sum_{t=1}^{T} ||\hat{\mathbf{z}}_t - \mathbf{z}_t\,||_2^2. \tag{11}$$

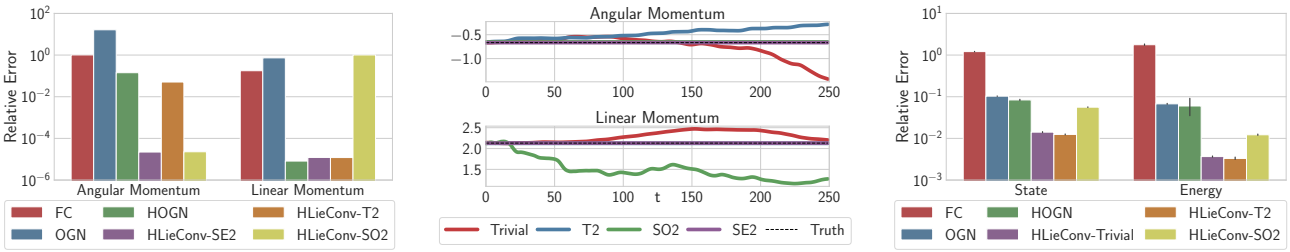
6.2. Exact Conservation of Momentum

While equivariance is broadly useful as an inductive bias, it has a very special implication for the modeling of Hamiltonian systems. Noether's Hamiltonian theorem states that each continuous symmetry in the Hamiltonian of a dynamical system has a corresponding conserved quantity (Noether, 1971; Butterfield, 2006). Symmetry with respect to the continuous transformations of translations and rotations lead directly to conservation of the total linear and angular momentum of the system, an extremely valuable property for modeling dynamical systems. In fact, all models that exactly conserve linear and angular momentum must have a corresponding translational and rotational symmetry. See Appendix A.5 for a primer on Hamiltonian symmetries, Noether's theorem, and the implications in the current setting.

As showed in Section 4, we can construct models that are equivariant to a large variety of *continuous* Lie Group symmetries, and therefore we can exactly conserve associated quantities like linear and angular momentum. Figure 7(a) shows that using LieConv layers with a given T(2) and/or SO(2) symmetry, the model trajectories conserve linear and/or angular momentum with relative error close to machine epsilon, determined by the integrator tolerance. As there is no corresponding Noether conservation for discrete symmetry groups, discrete approaches to enforcing symmetry (Cohen and Welling, 2016a; Marcos et al., 2017) would not be nearly as effective.

6.3. Results

For evaluation, we compare a fully-connected (FC) Neural-ODE model (Chen et al., 2018), ODE graph networks (OGN) (Battaglia et al., 2016), Hamiltonian ODE graph networks (HOGN) (Sanchez-Gonzalez et al., 2019), and our own LieConv architecture on predicting the motion of point particles connected by springs as described in (Sanchez-Gonzalez et al., 2019). Figure 6 shows exam-



(a) (b) (c)

Figure 7. **Left:** We can directly control whether linear and angular momentum is conserved by changing the model's symmetries. The components of linear and angular momentum of the integrated trajectories are conserved up to integrator tolerance. **Middle:** Momentum along the rollout trajectories for the LieConv models with different imposed symmetries. **Right:** Our method outperforms HOGN, a state-of-the-art model, on both state rollout error and system energy conservation.

ple rollout trajectories, and our quantitative results are presented in Figure 7. In the spring problem N bodies with mass $m_1, \ldots, m_N$ interact through pairwise spring forces with constants $k_1, \ldots, k_{N\times N}$. The system preserves energy, linear momentum, and angular momentum. The behavior of the system depends both the values of the system parameters ($\mathbf{s} = (k, m)$) and the initial conditions $\mathbf{z}_0$. The dynamics model must learn not only to predict trajectories across a broad range of initial conditions, but also infer the dependence on varied system parameters, which are additional inputs to the model. We compare models that attempt to learn the dynamics $F_\theta(\mathbf{z}, t) = d\mathbf{z}/dt$ directly against models that learn the Hamiltonian as described in section 6.1.

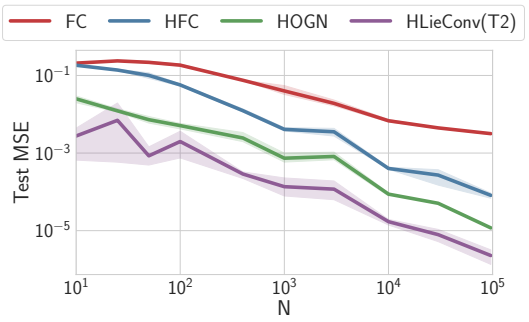


Figure 8. Test MSE as a function of the number of examples in the training dataset, N. As the inductive biases of Hamiltonian, Graph-Network, and LieConv equivariance are added, generalization performance improves. LieConv outperforms the other methods across all dataset sizes. The shaded region corresponds to a 95% confidence interval, estimated across 3 trials.

In Figure 7(a) and 7(b) we show that by changing the invariance of our Hamiltonian models, we have direct control over the conservation of linear and angular momentum in the predicted trajectories. Figure 7(c) demonstrates that our method outperforms HOGN, a SOTA architecture for dynamics problems, and achieves significant improvement over the naïve fully-connected (FC) model. We summarize the various models and their symmetries in Table 6. Finally, in Figure 8 we evaluate test MSE of the different models over a range of training dataset sizes, highlighting the additive improvements in generalization from the Hamiltonian, Graph-Network, and equivariance inductive biases successively.

7. Discussion

We presented a convolutional layer to build networks that can handle a wide variety of data types, and flexibly swap out the equivariance of the model. While the image, molecular, and dynamics experiments demonstrate the generality of our method, there are many exciting application domains (e.g. time-series, geostats, audio, mesh) and directions for future work. We also believe that it will be possible to benefit from the inductive biases of HLieConv models even for systems that do not exactly preserve energy or momentum, such as those found in control systems and reinforcement learning.

The success of convolutional neural networks on images has highlighted the power of encoding symmetries in models for learning from raw sensory data. But the variety and complexity of other modalities of data is a significant challenge in further developing this approach. More general data may not be on a grid, it may possess other kinds of symmetries, or it may contain quantities that cannot be easily combined. We believe that central to solving this problem is a decoupling of convenient computational representations of data as dense arrays from the set of geometrically sensible operations they may have. We hope to move towards models that can 'see' molecules, dynamical systems, multi-scale objects, heterogeneous measurements, and higher mathematical objects, in the way that convolutional neural networks perceive images.

Acknowledgements

MF, SS, PI and AGW are supported by an Amazon Research Award, Amazon Machine Learning Research Award, Facebook Research, NSF I-DISRE 193471, NIH R01 DA048764-01A1, NSF IIS-1910266, NSF 1922658 NRT-HDR: FUTURE Foundations, Translation, and Responsibility for Data Science, and by the United States Department of Defense through the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program. We thank Alex Wang for useful comments.

References

Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In *Advances in Neural Information Processing Systems*, pages 14510–14519, 2019.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.

Erik J Bekkers. B-spline cnns on lie groups. *arXiv preprint arXiv:1909.12057*, 2019.

L. C. Blum and J.-L. Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *J. Am. Chem. Soc.*, 131:8732, 2009.

Jeremy Butterfield. On symmetry and conserved quantities in classical mechanics. In *Physical theory and its interpretation*, pages 43–100. Springer, 2006.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016a.

Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016b.

Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. In *Advances in Neural Information Processing Systems*, pages 9142–9153, 2019.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

Ethan Eade. Lie groups for computer vision. *Cambridge Univ., Cam-bridge, UK, Tech. Rep*, 2014.

Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15353–15363, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Zhiwu Huang, Chengde Wan, Thomas Probst, and Luc Van Gool. Deep learning on lie groups for skeleton-based action recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6099–6108, 2017.

Jörn-Henrik Jacobsen, Bert De Brabandere, and Arnold WM Smeulders. Dynamic steerable blocks in deep residual networks. *arXiv preprint arXiv:1706.00598*, 2017.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

Sanket Kamthe and Marc Peter Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. *arXiv preprint arXiv:1706.06491*, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv preprint arXiv:1802.03690*, 2018.

Akira Kono and Kiminao Ishitoya. Squaring operations in mod 2 cohomology of quotients of compact lie groups by maximal tori. In *Algebraic Topology Barcelona 1986*, pages 192–206. Springer, 1987.

James J Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3993–3998. IEEE, 2004.

Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 289–297, 2016.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015.

Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5048–5057, 2017.

Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

Emmy Noether. Invariant variation problems. *Transport Theory and Statistical Physics*, 1(3):186–207, 1971.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, 108:058301, 2012.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Kristof T Schütt, Huziel E Sauceda, P-J Kindermans, Alexandre Tkatchenko, and K-R Müller. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, 2018.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. In *Advances in Neural Information Processing Systems*, pages 14334–14345, 2019.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco S Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, pages 10381–10392, 2018.

Benjamin Willson. Reiter nets for semidirect products of amenable groups and semigroups. *Proceedings of the American Mathematical Society*, 137(11):3823–3832, 2009.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.

Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528, 2017.