Appendices

A. Derivations and Additional Methodology

A.1. Generalized PointConv Trick

The matrix notation becomes very cumbersome for manipulating these higher order n-dimensional arrays, so we will instead use index notation with Latin indices i, j, k indexing points, Greek indices α, β, γ indexing feature channels, and c indexing the coordinate dimensions of which there are $d = 3$ for PointConv and $d = \dim(G) + 2\dim(Q)$ for LieConv.[3] As the objects are not geometric tensors but simply n-dimensional arrays, we will make no distinction between upper and lower indices. After expanding into indices, it should be assumed that all values are scalars, and that any free indices can range over all of the values.

Let $k_{ij}^{\alpha,\beta}$ be the output of the MLP k_θ which takes $\{a_{ij}^c\}$ as input and acts independently over the locations i, j. For PointConv, the input $a_{ij}^c = x_i^c - x_j^c$ and for LieConv the input $a_{ij}^c = \mathrm{Concat}([\log(v_j^{-1}u_i), q_i, q_j])^c$.

We wish to compute

$$h_i^\alpha = \sum_{j,\beta} k_{ij}^{\alpha,\beta} f_j^\beta. \tag{12}$$

In Wu et al. (2019), it was observed that since $k_{ij}^{\alpha,\beta}$ is the output of an MLP, $k_{ij}^{\alpha,\beta} = \sum_\gamma W_\gamma^{\alpha,\beta} s_{i,j}^\gamma$ for some final weight matrix W and penultimate activations $s_{i,j}^\gamma$ ($s_{i,j}^\gamma$ is simply the result of the MLP after the last nonlinearity). With this in mind, we can rewrite (12)

$$h_i^\alpha = \sum_{j,\beta} \big(\sum_\gamma W_\gamma^{\alpha,\beta} s_{i,j}^\gamma\big) f_j^\beta \tag{13}$$

$$= \sum_{\beta,\gamma} W_\gamma^{\alpha,\beta} \big(\sum_j s_{i,j}^\gamma f_j^\beta\big) \tag{14}$$

In practice, the intermediate number of channels is much less than the product of c_{in} and c_{out}: $|\gamma| < |\alpha||\beta|$ and so this reordering of the computation leads to a massive reduction in both memory and compute. Furthermore, $b_i^{\gamma,\beta} = \sum_j s_{i,j}^\gamma f_j^\beta$ can be implemented with regular matrix multiplication and $h_i^\alpha = \sum_{\beta,\gamma} W_\gamma^{\alpha,\beta} b_i^{\gamma,\beta}$ can be also by flattening (β, γ) into a single axis ε: $h_i^\alpha = \sum_\varepsilon W^{\alpha,\varepsilon} b_i^\varepsilon$.

The sum over index j can be restricted to a subset $j(i)$ (such as a chosen neighborhood) by computing $f_{(\cdot)}^\beta$ at each of the required indices and padding to the size of the maximum subset with zeros, and computing $b_i^{\gamma,\beta} = \sum_j s_{i,j(i)}^\gamma f_{j(i)}^\beta$ using dense matrix multiplication. Masking out of the values at indices i and j is also necessary when there are different numbers of points per minibatch but batched together using zero padding. The generalized PointConv trick can thus be applied in batch mode when there may be varied number of points per example and varied number of points per neighborhood.

A.2. Abelian G and Coordinate Transforms

For Abelian groups that cover $\mathcal{X}$ in a single orbit, the computation is very similar to ordinary Euclidean convolution. Defining $a_i = \log(u_i)$, $b_j = \log(v_j)$, and using the fact that $e^{-b_j}e^{a_i} = e^{a_i - b_j}$ means that $\log(v_j^{-1}u_i) = (\log \circ \exp)(a_i - b_j)$. Defining $\tilde{f} = f \circ \exp$, $\tilde{h} = h \circ \exp$; we get

$$\tilde{h}(a_i) = \frac{1}{n} \sum_{j \in \mathrm{nbhd}(i)} (\tilde{k}_\theta \circ \mathrm{proj})(a_i - b_j) \tilde{f}(b_j), \tag{15}$$

where $\mathrm{proj} = \log \circ \exp$ projects to the image of the logarithm map. Apart from a projection and a change to logarithmic coordinates, this is equivalent to Euclidean convolution in a vector space with dimensionality of the group. When the group is Abelian and $\mathcal{X}$ is a homogeneous space, then the dimension of the group is the dimension of the input. In these cases we have a trivial stabilizer group H and single origin o, so we can view f and h as acting on the input $x_i = u_i o$.

This directly generalizes some of the existing coordinate transform methods for achieving equivariance from the literature such as log polar coordinates for rotation and scaling equivariance (Esteves et al., 2017), and using hyperbolic coordinates for squeeze and scaling equivariance.

Log Polar Coordinates: Consider the Abelian Lie group of positive scalings and rotations: $G = \mathbb{R}^* \times$ SO(2) acting on $\mathbb{R}^2$. Elements of the group $M \in G$ can be expressed as a 2×2 matrix

$$M(r, \theta) = \begin{bmatrix} r\cos(\theta) & -r\sin(\theta) \\ r\sin(\theta) & r\cos(\theta) \end{bmatrix}$$

for $r \in \mathbb{R}^+$ and $\theta \in \mathbb{R}$. The matrix logarithm is[4]

$$\log\left(\begin{bmatrix} r\cos(\theta) & -r\sin(\theta) \\ r\sin(\theta) & r\cos(\theta) \end{bmatrix}\right) = \begin{bmatrix} \log(r) & -\theta \bmod 2\pi \\ \theta \bmod 2\pi & \log(r) \end{bmatrix},$$

or more compactly $\log(M(r,\theta)) = \log(r)I + (\theta \bmod 2\pi)J$, which is $[\log(r), \theta \bmod 2\pi]$ in the basis for the Lie algebra $[I, J]$. It is clear that $\mathrm{proj} = \log \circ \exp$ is simply $\bmod 2\pi$ on the J component.

As $\mathbb{R}^2$ is a homogeneous space of G, one can choose the global origin $o = [1, 0] \in \mathbb{R}^2$. A little algebra shows that

[3] $\dim(Q)$ is the dimension of the space into which Q, the orbit identifiers, are embedded.

[4] Here $\theta \bmod 2\pi$ is defined to mean $\theta + 2\pi n$ for the integer n such that the value is in $(-\pi, \pi)$, consistent with the principal matrix logarithm. $(\theta + \pi)\%2\pi - \pi$ in programming notation.

lifting to the group yields the transformation $u_i = M(r_i, \theta_i)$ for each point $p_i = u_i o$, where $r = \sqrt{x^2 + y^2}$, and $\theta = \mathrm{atan2}(y, x)$ are the polar coordinates of the point p_i. Observe that the logarithm of $v_j^{-1}u_i$ has a simple expression highlighting the fact that it is invariant to scale and rotational transformations of the elements,

$$\begin{aligned}\log(v_j^{-1}u_i) &= \log(M(r_j,\theta_j)^{-1}M(r_i,\theta_i)) \\ &= \log(r_i/r_j)I + (\theta_i - \theta_j \bmod 2\pi)J.\end{aligned}$$

Now writing out our Monte Carlo estimation of the integral:

$$h(p_i) = \frac{1}{n}\sum_j \tilde{k}_\theta(\log(r_i/r_j), \theta_i - \theta_j \bmod 2\pi)f(p_j),$$

which is a discretization of the log polar convolution from Esteves et al. (2017). This can be trivially extended to encompass cylindrical coordinates with the group $T(1) \times \mathbb{R}^* \times$ SO(2).

Hyperbolic coordinates: For another nontrivial example, consider the group of scalings and squeezes $G = \mathbb{R}^* \times$ SQ acting on the positive orthant $\mathcal{X} = \{(x, y) \in \mathbb{R}^2 : x > 0, y > 0\}$. Elements of the group can be expressed as the product of a squeeze mapping and a scaling

$$M(r,s) = \begin{bmatrix} s & 0 \\ 0 & 1/s \end{bmatrix}\begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} = \begin{bmatrix} rs & 0 \\ 0 & r/s \end{bmatrix}$$

for any $r, s \in \mathbb{R}^+$. As the group is abelian, the logarithm splits nicely in terms of the two generators I and A:

$$\log\left(\begin{bmatrix} rs & 0 \\ 0 & r/s \end{bmatrix}\right) = (\log r)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + (\log s)\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Again $\mathcal{X}$ is a homogeneous space of G, and we choose a single origin $o = [1, 1]$. With a little algebra, it is clear that $M(r_i, s_i)o = p_i$ where $r = \sqrt{xy}$ and $s = \sqrt{x/y}$ are the hyperbolic coordinates of p_i.

Expressed in the basis $\mathcal{B} = [I, A]$ for the Lie algebra above, we see that

$$\log(v_j^{-1}u_i) = \log(r_i/r_j)I + \log(s_i/s_j)A$$

yielding the expression for convolution

$$h(p_i) = \frac{1}{n}\sum_j \tilde{k}_\theta(\log(r_i/r_j), \log(s_i/s_j))f(p_j),$$

which is equivariant to squeezes and scalings.

As demonstrated, equivariance to groups that contain the input space in a single orbit and are abelian can be achieved with a simple coordinate transform; however our approach generalizes to groups that are both 'larger' and 'smaller' than the input space, including coordinate transform equivariance as a special case.

A.3. Sufficient Conditions for Geodesic Distance

In general, the function $d(u, v) = \|\log(v^{-1}u)\|_F$, defined on the domain of $\mathrm{GL}(d)$ covered by the exponential map, satisfies the first three conditions of a distance metric but not the triangle inequality, making it a semi-metric:

1. $d(u, v) \geq 0$
2. $d(u, v) = 0 \Leftrightarrow \log(u^{-1}v) = 0 \Leftrightarrow u = v$
3. $d(u, v) = \|\log(v^{-1}u)\| = \|-\log(u^{-1}v)\| = d(v, u)$.

However for certain subgroups of $\mathrm{GL}(d)$ with additional structure, the triangle inequality holds and the function is the distance along geodesics connecting group elements u and v according to the metric tensor

$$\langle A, B\rangle_u := \mathrm{Tr}(A^T u^{-T} u^{-1} B), \tag{16}$$

where $-T$ denotes inverse and transpose.

Specifically, if the subgroup G is in the image of the $\exp : \mathfrak{g} \to G$ map and each infinitesmal generator commutes with its transpose: $[A, A^T] = 0$ for $\forall A \in \mathfrak{g}$, then $d(u, v) = \|\log(v^{-1}u)\|_F$ is the geodesic distance between u, v.

Geodesic Equation: Geodesics of (16) satisfying $\nabla_{\dot\gamma}\dot\gamma = 0$ can equivalently be derived by minimizing the energy functional

$$E[\gamma] = \int_\gamma \langle\dot\gamma, \dot\gamma\rangle_\gamma dt = \int_0^1 \mathrm{Tr}(\dot\gamma^T\gamma^{-T}\gamma^{-1}\dot\gamma)dt$$

using the calculus of variations. Minimizing curves $\gamma(t)$, connecting elements u and v in G ($\gamma(0) = v, \gamma(1) = u$) satisfy

$$0 = \delta E = \delta\int_0^1 \mathrm{Tr}(\dot\gamma^T\gamma^{-T}\gamma^{-1}\dot\gamma)dt$$

Noting that $\delta(\gamma^{-1}) = -\gamma^{-1}\delta\gamma\gamma^{-1}$ and the linearity of the trace,

$$2\int_0^1 \mathrm{Tr}(\dot\gamma^T\gamma^{-T}\gamma^{-1}\delta\dot\gamma) - \mathrm{Tr}(\dot\gamma^T\gamma^{-T}\gamma^{-1}\delta\gamma\gamma^{-1}\dot\gamma)dt = 0.$$

Using the cyclic property of the trace and integrating by parts, we have that

$$-2\int_0^1 \mathrm{Tr}\left(\left(\frac{d}{dt}(\dot\gamma^T\gamma^{-T}\gamma^{-1}) + \gamma^{-1}\dot\gamma\dot\gamma^T\gamma^{-T}\gamma^{-1}\right)\delta\gamma\right)dt = 0,$$

where the boundary term $\mathrm{Tr}(\dot\gamma\gamma^{-T}\gamma^{-1}\delta\gamma)\big|_0^1$ vanishes since $(\delta\gamma)(0) = (\delta\gamma)(1) = 0$.

As $\delta\gamma$ may be chosen to vary arbitrarily along the path, γ must satisfy the geodesic equation:

$$\frac{d}{dt}(\dot\gamma^T\gamma^{-T}\gamma^{-1}) + \gamma^{-1}\dot\gamma\dot\gamma^T\gamma^{-T}\gamma^{-1} = 0. \tag{17}$$

Solutions: When $A = \log(v^{-1}u)$ satisfies $[A, A^T] = 0$, the curve $\gamma(t) = v\exp(t\log(v^{-1}u))$ is a solution to the geodesic equation (17). Clearly γ connects u and v, $\gamma(0) = v$ and $\gamma(1) = u$. Plugging in $\dot{\gamma} = \gamma A$ into the left hand side of equation (17), we have

$$\begin{aligned}&= \frac{d}{dt}(A^T\gamma^{-1}) + AA^T\gamma^{-1}\\ &= -A^T\gamma^{-1}\dot{\gamma}\gamma^{-1} + AA^T\gamma^{-1}\\ &= [A, A^T]\gamma^{-1} = 0\end{aligned}$$

Length of γ: The length of the curve γ connecting u and v is $\|\log(v^{-1}u)\|_F$,

$$\begin{aligned}L[\gamma] &= \int_\gamma \sqrt{\langle\dot{\gamma},\dot{\gamma}\rangle_\gamma}dt = \int_0^1 \sqrt{\mathrm{Tr}(\dot{\gamma}^T\gamma^{-T}\gamma^{-1}\dot{\gamma})}dt\\ &= \int_0^1 \sqrt{\mathrm{Tr}(A^TA)}dt = \|A\|_F = \|\log(v^{-1}u)\|_F\end{aligned}$$

Of the Lie Groups that we consider in this paper, all of which have a single connected component, the groups $G = T(\mathrm{d}), SO(\mathrm{d}), \mathbb{R}^* \times SO(\mathrm{d}), \mathbb{R}^* \times \mathrm{SQ}$ satisfy this property that $[\mathfrak{g}, \mathfrak{g}^T] = 0$; however, the SE($d$) groups do not.

A.4. Equivariant Subsampling

Even if all distances and neighborhoods are precomputed, the cost of computing equation (6) for $i = 1, ..., N$ is still quadratic, $O(nN) = O(N^2)$, because the number of points in each neighborhood n grows linearly with N as f is more densely evaluated. So that our method can scale to handle a large number of points, we show two ways two equivariantly subsample the group elements, which we can use both for the locations at which we evaluate the convolution and the locations that we use for the Monte Carlo estimator. Since the elements are spaced irregularly, we cannot readily use the coset pooling method described in (Cohen and Welling, 2016a), instead we can perform:

Random Selection: Randomly selecting a subset of p points from the original n preserves the original sampling distribution, so it can be used.

Farthest Point Sampling: Given a set of group elements $S = \{u_i\}_{i=1}^k \in G$, we can select a subset S_p^* of size p by maximizes the minimum distance between any two elements in that subset,

$$\mathrm{Sub}_p(S) := S_p^* = \underset{S_p \subset S}{\arg\max} \min_{u,v \in S_p : u \neq v} d(u, v), \quad (18)$$

farthest point sampling on the group. Acting on a set of elements, $\mathrm{Sub}_p : S \mapsto S_p^*$, the farthest point subsampling is equivariant $\mathrm{Sub}_p(wS) = w\mathrm{Sub}_p(S)$ for any $w \in G$. Meaning that applying a group element to each of the elements does not change the chosen indices in the subsampled set because the distances are left invariant $d(u_i, u_j) = d(wu_i, wu_j)$.

Now we can use either of these methods for $\mathrm{Sub}_p(\cdot)$ to equivariantly subsample the quadrature points in each neighborhood used to estimate the integral to a fixed number p,

$$h_i = \frac{1}{p} \sum_{j \in \mathrm{Sub}_p(\mathrm{nbhd}(u_i))} k_\theta(v_j^{-1}u_i) f_j. \quad (19)$$

Doing so has reduced the cost of estimating the convolution from $O(N^2)$ to $O(pN)$, ignoring the cost of computing Sub_p and $\{\mathrm{nbhd}(u_i)\}_{i=1}^N$.

A.5. Review and Implications of Noether's Theorem

In the Hamiltonian setting, Noether's theorem relates the continuous symmetries of the Hamiltonian of a system with conserved quantities, and has been deeply impactful in the understanding of classical physics. We give a review of Noether's theorem, loosely following Butterfield (2006).

More on Hamiltonian Dynamics

As introduced earlier, the Hamiltonian is a function acting on the state $H(z) = H(q, p)$, (we will ignore time dependence for now) can be viewed more formally as a function on the cotangent bundle $(q, p) = z \in M = T^*C$ where C is the coordinate configuration space, and this is the setting for Hamiltonian dynamics.

In general, on a manifold $\mathcal{M}$, a vector field X can be viewed as an assignment of a directional derivative along $\mathcal{M}$ for each point $z \in \mathcal{M}$. It can be expanded in a basis using coordinate charts $X = \sum_\alpha X^\alpha \partial_\alpha$, where $\partial_\alpha = \frac{\partial}{\partial z^\alpha}$ and acts on functions f by $X(f) = \sum_\alpha X^\alpha \partial_\alpha f$. In the chart, each of the components X^α are functions of z.

In Hamiltonian mechanics, for two functions on M, there is the Poisson bracket which can be written in terms of the canonical coordinates q_i, p_i, [5]

$$\{f, g\} = \sum_i \frac{\partial f}{\partial p_i}\frac{\partial g}{\partial q_i} - \frac{\partial f}{\partial q_i}\frac{\partial g}{\partial p_i}.$$

The Poisson bracket can be used to associate each function f to a vector field

$$X_f = \{f, \cdot\} = \sum_i \frac{\partial f}{\partial p_i}\frac{\partial}{\partial q_i} - \frac{\partial f}{\partial q_i}\frac{\partial}{\partial p_i},$$

which specifies, by its action on another function g, the directional derivative of g along X_f: $X_f(g) = \{f, g\}$. Vector fields that can be written in this way are known as Hamiltonian vector fields, and the Hamiltonian dynamics of the

[5]Here we take the definition of the Poisson bracket to be negative of the usual definition in order to streamline notation.

system is a special example $X_H = \{H, \cdot\}$. This vector field in canonical coordinates $z = (p, q)$ is the vector field $X_H = F(z) = J\nabla_z H$ (i.e. the symplectic gradient, as discussed in Section 6.1). Making this connection clear, a given scalar quantity evolves through time as $\dot{f} = \{H, f\}$. But this bracket can be used to evaluate the rate of change of a scalar quantity along the flows of vector fields other than the dynamics, such as the flows of continuous symmetries.

Noether's Theorem

The flow ϕ_λ^X by $\lambda \in \mathbb{R}$ of a vector field X is the set of integral curves, the unique solution to the system of ODEs $\dot{z}^\alpha = X^\alpha$ with initial condition z and at parameter value λ, or more abstractly the iterated application of X: $\phi_\lambda^X = \exp(\lambda X)$. Continuous symmetries transformation are the transformations that can be written as the flow ϕ_λ^X of a vector field. The directional derivative characterizes how a function such as the Hamiltonian changes along the flow of X and is a special case of the Lie Derivative $\mathcal{L}$.

$$\mathcal{L}_X H = \frac{d}{d\lambda}(H \circ \phi_\lambda^X)\big|_{\lambda=0} = X(H)$$

A scalar function is invariant to the flow of a vector field if and only if the Lie Derivative is zero

$$H(\phi_\lambda^X(z)) = H(z) \Leftrightarrow \mathcal{L}_X H = 0.$$

For all transformations that respect the Poisson Bracket[6], which we add as a requirement for a symmetry, the vector field X is (locally) Hamiltonian and there exists a function f such that $X = X_f = \{f, \cdot\}$. If M is a contractible domain such as $\mathbb{R}^{2n}$, then f is globally defined. For every continuous symmetry $\phi_\lambda^{X_f}$,

$$\mathcal{L}_{X_f} H = X_f(H) = \{f, H\} = -\{H, f\} = -X_H(f),$$

by the antisymmetry of the Poisson bracket. So if ϕ_λ^X is a symmetry of H, then $X = X_f$ for some function f, and $H(\phi_\lambda^{X_f}(z)) = H(z)$ implies

$$\mathcal{L}_{X_f} H = 0 \Leftrightarrow \mathcal{L}_{X_H} f = 0 \Leftrightarrow f(\phi_\tau^{X_H}(z)) = f(z)$$

or in other words $f(z(t+\tau)) = f(z(t))$ and f is a conserved quantity of the dynamics.

This implication goes both ways, if f is conserved then $\phi_\lambda^{X_f}$ is necessarily a symmetry of the Hamiltonian, and if $\phi_\lambda^{X_f}$ is a symmetry of the Hamiltonian then f is conserved.

Hamiltonian vs Dynamical Symmetries

So far we have been discussing Hamiltonian symmetries, invariances of the Hamiltonian. But in the study of dynamical systems there is a related concept of dynamical symmetries, symmetries of the equations of motion. This notion is also captured by the Lie Derivative, but between vector fields. A dynamical system $\dot{z} = F(z)$, has a continuous dynamical symmetry ϕ_λ^X if the flow along the dynamical system commutes with the symmetry:

$$\phi_\lambda^X(\phi_t^F(z)) = \phi_t^F(\phi_\lambda^X(z)). \tag{20}$$

Meaning that applying the symmetry transformation to the state and then flowing along the dynamical system is equivalent to flowing first and then applying the symmetry transformation. Equation (20) is satisfied if and only if the Lie Derivative is zero:

$$\mathcal{L}_X F = [X, F] = 0,$$

where $[\cdot, \cdot]$ is the Lie bracket on vector fields.[7]

For Hamiltonian systems, every Hamiltonian symmetry is also a dynamical symmetry. In fact, it is not hard to show that the Lie and Poisson brackets are related,

$$[X_f, X_g] = X_{\{f,g\}}$$

and this directly shows the implication. If X_f is a Hamiltonian symmetry, $\{f, H\} = 0$, and then

$$[X_f, F] = [X_f, X_H] = X_{\{f,H\}} = 0.$$

However, the converse is not true, dynamical symmetries of a Hamiltonian system are not necessarily Hamiltonian symmetries and thus might not correspond to conserved quantities. Furthermore even if the system has a dynamical symmetry which is the flow along a Hamiltonian vector field ϕ_λ^X, $X = X_f = \{f, \cdot\}$, but the dynamics F are not Hamiltonian, then the dynamics will not conserve f in general. Both the symmetry and the dynamics must be Hamiltonian for the conservation laws.

This fact is demonstrated by Figure 9, where the dynamics of the (non-Hamiltonian) equivariant LieConv-T(2) model has a T(2) dynamical symmetry with the generators ∂_x, ∂_y which are Hamiltonian vector fields for $f = p_x, f = p_y$, and yet linear momentum is not conserved by the model.

[6]More precisely, the Poisson Bracket can be formulated in a coordinate free manner in terms of a symplectic two form ω, $\{f, g\} = \omega(X_f, X_g)$. In the original coordinates $\omega = \sum_i dp_i \wedge dq^i$, and this coordinate basis, ω is represented by the matrix J from earlier. The dynamics X_H are determined by $\mathrm{d}H = \omega(X_H, \cdot) = \iota_{X_H}\omega$. Transformations which respect the Poisson Bracket are symplectic, $\mathcal{L}_X \omega = 0$. With Cartan's magic formula, this implies that $\mathrm{d}(\iota_X \omega) = 0$. Because the form $\iota_X \omega$ is closed, Poincare's Lemma implies that locally $(\iota_X \omega) = \mathrm{d}f)$ for some function f and hence $X = X_f$ is (locally) a Hamiltonian vector field. For more details see Butterfield (2006).

[7]The Lie bracket on vector fields produces another vector field and is defined by how it acts on functions, for any smooth function g: $[X, F](g) = X(F(g)) - F(X(g))$

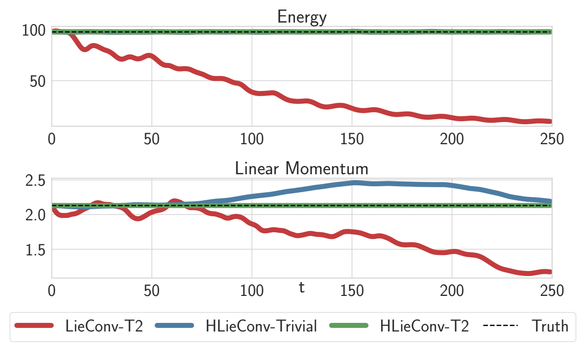


Figure 9. Equivariance alone is not sufficient, for conservation we need *both* to model $\mathcal{H}$ and incorporate the given symmmetry. For comparison, LieConv-T(2) is T(2)-equivariant but models F, and HLieConv-Trivial models $\mathcal{H}$ but is not T(2)-equivariant. Only HLieConv-T(2) conserves linear momentum.

Conserving Linear and Angular Momentum

Consider a system of N interacting particles described in Euclidean coordinates with position and momentum q_{im}, p_{im}, such as the multi-body spring problem. Here the first index $i = 1, 2, 3$ indexes the spatial coordinates and the second $m = 1, 2, ..., N$ indexes the particles. We will use the bolded notation $\mathbf{q}_m, \mathbf{p}_m$ to suppress the spatial indices, but still indexing the particles m as in Section 6.1.

The total linear momentum along a given direction $\mathbf{n}$ is $\mathbf{n} \cdot \mathbf{P} = \sum_{i,m} n_i p_{im} = \mathbf{n} \cdot (\sum_m \mathbf{p}_m)$. Expanding the Poisson bracket, the Hamiltonian vector field

$$X_{\mathbf{nP}} = \{\mathbf{n} \cdot \mathbf{P}, \cdot\} = \sum_{i,m} n_i \frac{\partial}{\partial q_{im}} = \mathbf{n} \cdot \sum_m \frac{\partial}{\partial \mathbf{q}_m}$$

which has the flow $\phi_\lambda^{X_{\mathbf{nP}}}(\mathbf{q}_m, \mathbf{p}_m) = (\mathbf{q}_m + \lambda \mathbf{n}, \mathbf{p}_m)$, a translation of all particles by $\lambda \mathbf{n}$. So our model of the Hamiltonian conserves linear momentum if and only if it is invariant to a global translation of all particles, (e.g. T(2) invariance for a 2D spring system).

The total angular momentum along a given axis $\mathbf{n}$ is

$$\mathbf{n} \cdot \mathbf{L} = \mathbf{n} \cdot \sum_m \mathbf{q}_m \times \mathbf{p}_m = \sum_{i,j,k,m} \epsilon_{ijk} n_i q_{jm} p_{km} = \sum_m \mathbf{p}_m^T A \mathbf{q}_m$$

, where ϵ_{ijk} is the Levi-Civita symbol and we have defined the antisymmetric matrix A by $A_{kj} = \sum_i \epsilon_{ijk} n_i$.

$$X_{\mathbf{nL}} = \{\mathbf{n} \cdot \mathbf{L}, \cdot\} = \sum_{j,k,m} A_{kj} q_{jm} \frac{\partial}{\partial q_{km}} - A_{jk} p_{jm} \frac{\partial}{\partial p_{km}}$$

$$X_{\mathbf{nL}} = \sum_m (\mathbf{q}_m^T A^T \frac{\partial}{\partial \mathbf{q}_m} + \mathbf{p}_m^T A^T \frac{\partial}{\partial \mathbf{p}_m})$$

where the second line follows from the antisymmetry of A. We can find the flow of $X_{\mathbf{nL}}$ from the differential equations $\dot{\mathbf{q}}_m = A\mathbf{q}, \dot{\mathbf{p}}_m = A\mathbf{q}$ which have the solution

$$\phi_\theta^{X_{\mathbf{nL}}}(\mathbf{q}_m, \mathbf{p}_m) = (e^{\theta A}\mathbf{q}_m, e^{\theta A}\mathbf{p}_m) = (R_\theta \mathbf{q}_m, R_\theta \mathbf{p}_m),$$

where R_θ is a rotation about the axis $\mathbf{n}$ by the angle θ, which follows from the Rodriguez rotation formula. Therefore, the flow of the Hamiltonian vector field of angular momentum along a given axis is a global rotation of the position and momentum of each particle about that axis. Again, the dynamics of a neural network modeling a Hamiltonian conserve total angular momentum if and only if the network is invariant to simultaneous rotation of all particle positions and momenta.

B. Additional Experiments

B.1. Equivariance Demo

While (7) shows that the convolution estimator is equivariant, we have conducted the ablation study below examining the equivariance of the network empirically. We trained LieConv (Trivial, T(3), SO(3), SE(3)) models on a limited subset of 20k training examples (out of 100k) of the HOMO task on QM9 without any data augmentation. We then evaluate these models on a series of modified test sets where each example has been randomly transformed by an element of the given group (the test translations in T(3) and SE(3) are sampled from a normal with stddev 0.5). In table B.1 the rows are the models configured with a given group equivariance and the columns N/G denote no augmentation at training time and transformations from G applied to the test set (test translations in T(3) and SE(3) are sampled from a normal with stddev 0.5).

Model	N/N	N/T(3)	N/SO(3)	N/SE(3)
Trivial	**173**	183	239	243
T(3)	**113**	**113**	133	133
SO(3)	**159**	238	**160**	240
SE(3)	**62**	**62**	**63**	**62**

Table 4. Test MAE (in meV) on HOMO test set randomly transformed by elements of G. Despite no data augmentation (N), G equivariant models perform as well on G transformed test data.

Notably, the performance of the LieConv-G models do not degrade when random G transformations are applied to the test set. Also, in this low data regime, the added equivariances are especially important.

B.2. RotMNIST Comparison

While the RotMNIST dataset consists of 12k rotated MNIST digits, it is standard to separate out 10k to be used for training and 2k for validation. However, in Ti-Pooling and E(2)-Steerable CNNs, it appears that after hyperparameters were tuned the validation set is folded back into the training set to be used as additional training data, a common approach used on other datasets. Although in table 1 we only use 10k training points, in the table below we report the performance with and without augmentation trained on the full 12k examples.

Aug	Trivial	T_y	T(2)	SO(2)	SO(2)×$\mathbb{R}^*$	SE(2)
SO(2)	1.44	1.35	1.32	1.27	1.13	1.13
None	1.60	2.64	2.34	1.26	1.25	1.15

Table 5. Classification Error (%) on RotMNIST dataset for LieConv with different group equivariances and baselines:

C. Implementation Details

C.1. Practical Considerations

While the high-level summary of the lifting procedure (Algorithm 1) and the LieConv layer (Algorithm 2) provides a useful conceptual understanding of our method, there are some additional details that are important for a practical implementation.

1. According to Algorithm 2, a_{ij} is computed in every LieConv layer, which is both highly redundant and costly. In practice, we precompute a_{ij} once after lifting and feed it through the network with layers operating on the state $\left(\{a_{ij}\}_{i,j}^{N,N}, \{f_i\}_{i=1}^{N}\right)$ instead of $\{(u_i, q_i, f_i)\}_{i=1}^{N}$. Doing so requires fixing the group elements that will be used at each layer for a given forwards pass.

2. In practice only p elements of nbhd_i are sampled (randomly) for computing the Monte Carlo estimator in order to limit the computational burden (see Appendix A.4).

3. We use the analytic forms for the exponential and logarithm maps of the various groups as described in Eade (2014).

C.2. Sampling from the Haar Measure for Various groups

When the lifting map from $\mathcal{X} \to G \times \mathcal{X}/G$ is multi-valued, we need to sample elements of $u \in G$ that project down to x: $uo = x$ in a way consistent with the Haar measure $\mu(\cdot)$. In other words, since the restriction $\mu(\cdot)|_{\mathrm{nbhd}}$ is a distribution, then we must sample from the conditional distribution $u \sim \mu(u|uo = x)|_{\mathrm{nbhd}}$. In general this can be done by parametrizing the distribution of μ as a collection of random variables that includes x, and then sampling the remaining variables.

In this paper, the groups we use in which the lifting map is multi-valued are SE(2), SO(3), and SE(3). The process is especially straightforward for SE(2) and SE(3) as these groups can be expressed as a semi-direct product of two groups $G = H \ltimes N$,

$$d\mu_G(h, n) = \delta(h) d\mu_H(h) d\mu_N(n), \tag{21}$$

where $\delta(h) = \frac{d\mu_N(n)}{d\mu_N(hnh^{-1})}$ (Willson, 2009). For $G = \mathrm{SE}(d) = \mathrm{SO}(d) \ltimes \mathrm{T}(d)$, $\delta(h) = 1$ since the Lebesgue measure $d\mu_{\mathrm{T}(d)}(x) = d\lambda(x) = dx$ is invariant to rotations. So simply $d\mu_{\mathrm{SE}(d)}(R, x) = d\mu_{\mathrm{SO}(d)}(R)dx$.

So lifts of a point $x \in \mathcal{X}$ to SE(d) consistent with the μ are just $T_x R$, the multiplication of a translation by x and randomly sampled rotations $R \sim \mu_{\mathrm{SO}(d)}(\cdot)$. There are multiple easy methods to sample uniformly from SO(d) given in (Kuffner, 2004), for example sampling uniformly from SO(3) can be done by sampling a unit quaternion from the 3-sphere, and identifying it with the corresponding rotation matrix.

C.3. Model Architecture

We employ a ResNet-style architecture (He et al., 2016), using bottleneck blocks (Zagoruyko and Komodakis, 2016), and replacing ReLUs with Swish activations (Ramachandran et al., 2017). The convolutional kernel g_θ internal to each LieConv layer is parametrized by a 3-layer MLP with 32 hidden units, batch norm, and Swish nonlinearities. Not only do the Swish activations improve performance slightly, but unlike ReLUs they are twice differentiable which is a requirement for backpropagating through the Hamiltonian dynamics. The stack of elementwise linear and bottleneck blocks is followed by a global pooling layer that computes the average over all elements, but not over channels. Like for regular image bottleneck blocks, the channels for the convolutional layer in the middle are smaller by a factor of 4 for increased parameter and computational efficiency.

Downsampling: As is traditional for image data, we increase the number of channels and the receptive field at every downsampling step. The downsampling is performed with the farthest point downsampling method described in Appendix A.4. For a downsampling by a factor of $s < 1$, the radius of the neighborhood is scaled up by $s^{-1/2}$ and the channels are scaled up by $s^{-1/2}$. When an image is downsampled with $s = (1/2)^2$ that is typical in a CNN, this results in 2x more channels and a radius or dilation of 2x. In the bottleneck block, the downsampling operation is fused with the LieConv layer, so that the convolution is only evaluated at the downsampled query locations. We perform downsampling only on the image datasets, which have more points.

BatchNorm: In order to handle the varied number of group elements per example and within each neighborhood, we use a modified batchnorm that computes statistics only over elements from a given mask. The batch norm is computed per channel, with statistics averaged over the batch size and each of the valid locations.

C.4. Details for Hamiltonian Models

Model Symmetries:

As the position vectors are mean centered in the model forward pass $\mathbf{q}_i' = \mathbf{q}_i - \bar{\mathbf{q}}$, HOGN and HLieConv-SO2* have additional T(2) invariance, yielding SE(2) invariance for HLieConv-SO2*. We also experimented with a HLieConv-SE2 equivariant model, but found that the exponential map for SE2 (involving taylor expands and masking) was not numerically stable enough for for second derivatives, required for optimizing through the Hamiltonian dynamics. So instead we benchmark the HLieConv-SO2 (without centering) and the HLieConv-SO2* (with centering) models separately. Layer equivariance is preferable for not prematurely discarding useful information and for better modeling performance, but invariance alone is sufficient for the conservation laws. Additionally, since we know *a priori* that the spring problem has Euclidean coordinates, we need not model the kinetic energy $K(\mathbf{p}, m) = \sum_{j=1}^{n} \| \mathbf{p}_j \|^2 / m_j$ and instead focus on modeling the potential $V(\mathbf{q}, k)$. We observe that this additional inductive bias of Euclidean coordinates improves model performance. Table 6 shows the invariance and equivariance properties of the relevant models and baselines. For Noether conservation, we need both to model the Hamiltonian and have the symmetry property.

Dataset Generation: To generate the spring dynamics datasets we generated D systems each with $N = 6$ particles connected by springs. The system parameters, mass and spring constant, are set by sampling $\{m_1^{(i)}, \dots m_6^{(i)}, k_1^{(i)}, \dots, k_6^{(i)}\}_{i=1}^{N}$, $m_j^{(i)} \sim \mathcal{U}(0.1, 3.1)$, $k_j^{(i)} \sim \mathcal{U}(0, 5)$. Following Sanchez-Gonzalez et al. (2019), we set the spring constants as $k_{ij} = k_i k_j$. For each system

	$F(\mathbf{z}, t)$	$\mathcal{H}(\mathbf{z}, t)$	$T(2)$	$SO(2)$
FC	●			
OGN	●			
HOGN		●	★	
LieConv-T(2)	●		✪	
HLieConv-Trivial		●		
HLieConv-T(2)		●	✪	
HLieConv-SO(2)		●		✪
HLieConv-SO(2)*		●	★	✪

Table 6. Model characteristics. Models with layers invariant to G are denoted with ★, and those with equivariant layers with ✪.

i, the position and momentum of body j were distributed as $\mathbf{q}_j^{(i)} \sim \mathcal{N}(0, 0.16I)$, $\mathbf{p}_j^{(i)} \sim \mathcal{N}(0, 0.36I)$. Using the analytic form of the Hamiltonian for the spring problem, $\mathcal{H}(\mathbf{q}, \mathbf{p}) = K(\mathbf{p}, m) + V(\mathbf{q}, k)$, we use the RK4 numerical integration scheme to generate 5 second ground truth trajectories broken up into 500 evaluation timesteps. We use a fixed step size scheme for RK4 chosen automatically (as implemented in Chen et al. (2018)) with a relative tolerance of 1e-8 in double precision arithmetic. We then randomly selected a single segment for each trajectory, consisting of an initial state $\mathbf{z}_t$ and $\tau = 4$ transition states: $(\mathbf{z}_{t+1}^{(i)}, \dots, \mathbf{z}_{t+\tau}^{(i)})$.

Training: All models were trained in single precision arithmetic (double precision did not make any appreciable difference) with an integrator tolerance of 1e-4. We use a cosine decay for the learning rate schedule and perform early stopping over the validation MSE. We trained with a minibatch size of 200 and for 100 epochs each using the Adam optimizer (Kingma and Ba, 2014) without batch normalization. With 3k training examples, the HLieConv model takes about 20 minutes to train on one 1080Ti.

For the examination of performance over the range of dataset sizes in 8, we cap the validation set to the size of the training set to make the setting more realistic, and we also scale the number of training epochs up as the size of the dataset shrinks (epochs $= 100(\sqrt{10^3/D})$) which we found to be sufficient to fit the training set. For $D \leq 200$ we use the full dataset in each minibatch.

Hyperparameters:

	channels	layers	lr
(H)FC	256	4	1e-2
(H)OGN	256	1	1e-2
(H)LieConv	384	4	1e-3

Hyperparameter tuning: Model hyperparameters were tuned by grid search over channel width, number of layers, and learning rate. The models were tuned with training, validation, and test datasets consisting of 3000, 2000, and 2000 trajectory segments respectively.

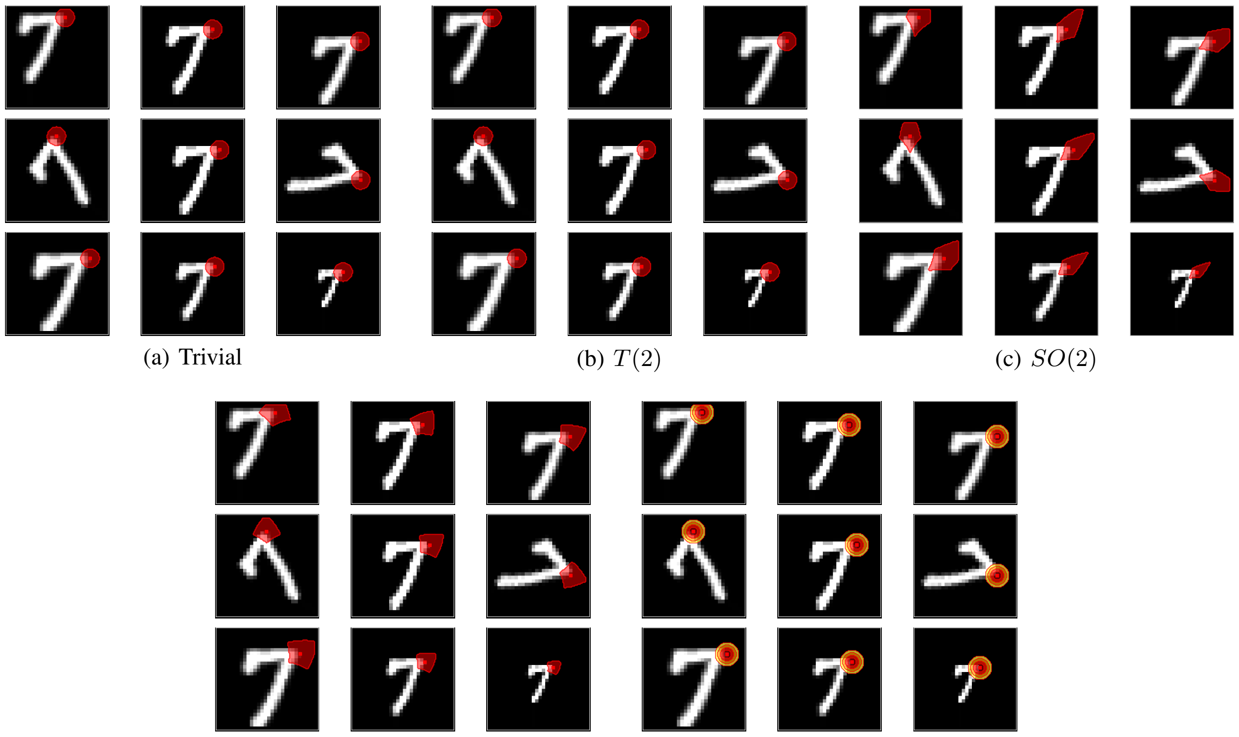
C.5. Details for Image and Molecular Experiments

RotMNIST Hyperparameters: For RotMNIST we train each model for 500 epochs using the Adam optimizer with learning rate 3e-3 and batch size 25. The first linear layer maps the 1-channel grayscale input to $k = 128$ channels, and the number of channels in the bottleneck blocks follow the scaling law from Appendix C.3 as the group elements are downsampled. We use 6 bottleneck blocks, and the total downsampling factor $S = 1/10$ is split geometrically between the blocks as $s = (1/10)^{1/6}$ per block. The initial radius r of the local neighborhoods in the first layer is set so as to include 1/15 of the total number of elements in each neighborhood and is scaled accordingly. The subsampled neighborhood used to compute the Monte Carlo convolution estimator uses $p = 25$ elements. The models take less than 12 hours to train on a 1080Ti.

QM9 Hyperparameters: For the QM9 molecular data, we use the featurization from Anderson et al. (2019), where the input features f_i are determined by the atom type (C,H,N,O,F) and the atomic charge. The coordinates x_i are simply the raw atomic coordinates measured in angstroms. A separate model is trained for each prediction task, all using the same hyperparameters and early stopping on the validation MAE. We use the same train, validation, test split as Anderson et al. (2019), with 100k molecules for train, 10% for test and the remaining for validation. Like with the other experiments, we use a cosine learning rate decay schedule. Each model is trained using the Adam optimizer for 1000 epochs with a learning rate of 3e-3 and batch size of 100. We use SO(3) data augmentation, 6 bottleneck blocks, each with $k = 1536$ channels. The radius of the local neighborhood is set to $r = \infty$ to include all elements. The model takes about 48 hours to train on a single 1080Ti.

C.6. Local Neighborhood Visualizations

In Figure 10 we visualize the local neighborhood used with different groups under three different types of transformations: translations, rotations and scaling. The distance and neighborhood are defined for the tuples of group elements and orbit. For Trivial, T(2), SO(2), $\mathbb{R} \times SO(2)$ the correspondence between points and these tuples is one-to-one and we can identify the neighborhood in terms of the input points. For SE(2) each point is mapped to multiple tuples, each of which defines its own neighborhood in terms of other tuples. In the Figure, for SE(2) for a given point we visualize the distribution of points that enter the computation of the convolution at a specific tuple.



(a) Trivial (b) $T(2)$ (c) $SO(2)$

(d) $\mathbb{R}^* \times SO(2)$ (e) $SE(2)$

Figure 10. A visualization of the local neighborhood for different groups, in terms of the points in the input space. For the computation of the convolution at the point in red, elements are sampled from colored region. In each panel, the top row shows translations, middle row shows rotations and bottom row shows scalings of the same image. For SE(2) we visualize the distribution of points entering the computation of the convolution over multiple lift samples. For each of the equivariant models that respects a given symmetry, the points that enter into the computation are not affected by the transformation.