Automatic Symmetry Discovery with Lie Algebra Convolutional Network

Anonymous Author(s)
Affiliation
Address
email

Abstract

Existing equivariant neural networks for continuous groups require discretization or group representations. All these approaches require detailed knowledge of the group parametrization and cannot learn entirely new symmetries. We propose to work with the Lie algebra (infinitesimal generators) instead of the Lie group. Our model, the Lie algebra convolutional network (L-conv) can learn potential symmetries and does not require discretization of the group. We show that L-conv can serve as a building block to construct any group equivariant architecture. We discuss how CNNs and Graph Convolutional Networks are related to and can be expressed as L-conv with appropriate groups. We also derive the MSE loss for a single L-conv layer and find a deep relation with Lagrangians used in physics, with some of the physics aiding in defining generalization and symmetries in the loss landscape. Conversely, L-conv could be used to propose more general equivariant ansätze for scientific machine learning.

1 Introduction

Encoding symmetries into a machine learning (ML) architecture (Cohen & Welling, 2016a; Bogatskiy et al., 2020; Maron et al., 2020; Bronstein et al., 2021) can reduce data requirements and improve generalization, while significantly reducing the number of model parameters (Cohen et al., 2019b; Cohen & Welling, 2016b; Ravanbakhsh et al., 2017; Ravanbakhsh, 2020). For instance, Convolutional Neural Networks (CNN) (LeCun et al., 1989, 1998) incorporate translation symmetry into the architecture. General principles for constructing group equivariant networks were introduced in Cohen & Welling (2016b), Kondor & Trivedi (2018), and Cohen et al. (2019b). Many existing works on equivariant architectures focus on *finite groups* such as permutations Hartford et al. (2018); Ravanbakhsh et al. (2017); Zaheer et al. (2017), 90 degree rotations Cohen et al. (2018) or dihedral groups D_N $E(2)$ Weiler & Cesa (2019).

For a continuous group, the number of group elements is infinite. To incorporate them into the architecture, existing approaches either discretize the group Weiler et al. (2018a,b); Cohen & Welling (2016a), or use a truncated sum over irreps Weiler & Cesa (2019); Weiler et al. (2018a) via spherical harmonics in Worrall et al. (2017) or more general Clebsch-Gordon coefficients Kondor et al. (2018); Bogatskiy et al. (2020). These approaches are prone to approximation error. Both parametrizing Lie groups for sampling and finding irreps are computationally expensive.

Another approach taken in Finzi et al. (2020) approximates the integral over the Lie group by Monte Carlo sampling. However, their approach is quite involved, using the matrix logarithm and exponential to define a distance function used to obtain a sampling neighborhood for each point. Finzi et al. (2021) provides a unified approach for encoding continuous groups multi-layer perceptrons (MLP). Their method requires knowledge of the group, encoding its irreducible representations, and solving a set of symmetry constraints.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Contributions Existing approaches require hand-coding group representations or elements into the architecture. Each Lie group has a different parametrization and this task is quite tedious. Our goal is to significantly simplify this process. Our intuition stems from the fact that Lie groups can be constructed from a set of infinitesimal generators, called the Lie algebra. The Lie algebra has a finite basis, assuming the group is not infinite-dimensional (e.g. in quantum or conformal field theory (Belavin et al., 1984)). By working with the Lie algebra basis, we avoid having to encode an infinite group by discretizing or summing over irreps. The other benefit of this approach is that all Lie algebras have the same general structure and hence can be implemented the same way. Additionally, for most symmetries we can expect the number of Lie algebra basis elements to be small and we could make them learnable parameters. Hence, our architecture is potentially capable of learning symmetries in data without imposing inductive biases.

We propose Lie Algebra Convolutional Network (**L-conv**), a novel architecture that can automatically discover symmetry from data. Our main contributions can be summarized as follows:

- We propose the Lie algebra convolutional network (**L-conv**), a building block for constructing group equivariant architectures.
- We show that multi-layer L-conv can approximate group convolutional layers, including CNN, and find graph convolutional networks to be a special case of L-conv.
- The Lie algebra generators in our L-conv can be learned to automatically discover symmetries.
- We find that MSE loss function for L-conv generalizes important Lagrangians in physics.
- We find generalization error and equivariance in the loss landscape can be expressed as Euler-Lagrange equations and Noether currents, transferring ideas from physics to ML.

2 Background

We now review core concepts which we build upon: equivariance, group convolution and Lie algebras.

Notations Unless explicitly stated, a in A^a is an index, not an exponent. We use the Einstein notation, where a repeated upper and lower index are summed over, meaning $A^a B_{ab} = \sum_a A^a B_{ab} = [AB]_b$.

Equivariance Let $\mathcal{S}$ be a topological space on which a Lie group G (continuous group) acts from the left, meaning for all $\boldsymbol{x} \in \mathcal{S}$ and $g \in G$, $g\boldsymbol{x} \in \mathcal{S}$. We refer to $\mathcal{S}$ as the base space. Let $\mathcal{F}$, the "feature space", be the vector space $\mathcal{F} = \mathbb{R}^m$. Each data point is a feature map $f : \mathcal{S} \to \mathcal{F}$. For instance, in images $f(\boldsymbol{x}) \in \mathcal{F}$ are the colors at pixel $\boldsymbol{x} \in \mathcal{S}$. The action of G on the input of f induces an action on feature maps. For $u \in G$, the transformed features $u \cdot f$ are given by

$$u \cdot f(\boldsymbol{x}) = f(u^{-1}\boldsymbol{x}). \tag{1}$$

Denote the space of all functions from $\mathcal{S}$ to $\mathcal{F}$ by $\mathcal{F}^{\mathcal{S}}$, so that $f \in \mathcal{F}^{\mathcal{S}}$. Let F be a mapping to a new feature space $\mathcal{F}' = \mathbb{R}^{m'}$, meaning $F : \mathcal{F}^{\mathcal{S}} \to \mathcal{F}'^{\mathcal{S}}$. We say F is *equivariant* under G if G acts on $\mathcal{F}'$ and for $u \in G$

$$u \cdot (F(f)) = F(u \cdot f). \tag{2}$$

Group Convolution Kondor & Trivedi (2018) showed that F is a linear equivariant map if and only if it performs a group convolution (G-conv). To define G-conv, we first lift $\boldsymbol{x}$ to elements in G (Kondor & Trivedi, 2018). To do the lift, we pick an origin $\boldsymbol{x}_0 \in \mathcal{S}$ and replace each point $\boldsymbol{x} = g\boldsymbol{x}_0$ by g. We will often drop $\boldsymbol{x}_0$ for brevity and write $f(g) \equiv f(g\boldsymbol{x}_0)$. Let $\kappa : G \to \mathbb{R}^{m'} \otimes \mathbb{R}^m$ be a linear transformation from $\mathcal{F}$ to $\mathcal{F}'$. G-conv is defined as

$$[\kappa \star f](g) = \int_G \kappa(g^{-1}v) f(v) dv = \int_G \kappa(v) f(gv) dv, \tag{3}$$

where for brevity, we are denoting the Haar measure on G as $dv \equiv d\mu(v)$.

Equivariance of G-conv As shown in Kondor & Trivedi (2018), G-conv is equivariant. The equivariance of equation 3 can be readily checked using equation 1. For $w \in G$ we have

$$\begin{aligned}[\kappa \star w \cdot f](g) &= \int_G \kappa(v) w \cdot f(gv) dv = \int_G \kappa(v) f(w^{-1}gv) dv \\ &= [\kappa \star f](w^{-1}g) = w \cdot [\kappa \star f](g)\end{aligned} \tag{4}$$

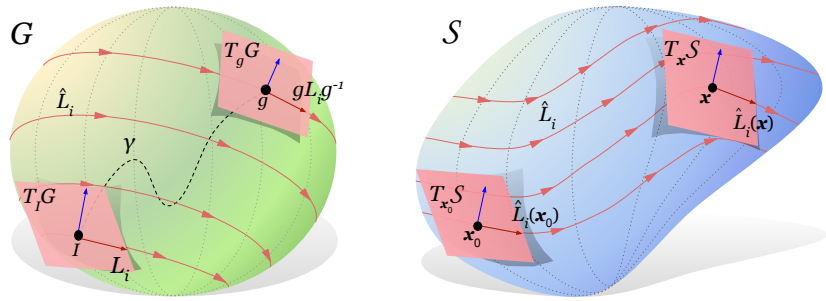

Figure 1: **Lie group and Lie algebra:** Illustration of the group manifold of a Lie group G (left). The Lie algebra $\mathfrak{g} = T_IG$ is the tangent space at the identity I. L_i are a basis for T_IG. If G is connected, $\forall g \in G$ there exist paths like γ from I to g and g can be written as a path-ordered integral $g = P\exp[\int_\gamma dt^i L_i]$. **Base space** Right is a schematic of the base space $\mathcal{S}$ as a manifold. The lift $\boldsymbol{x} = g\boldsymbol{x}_0$ takes $\boldsymbol{x} \in \mathcal{S}$ to $g \in G$, and maps the tangent spaces $T_{\boldsymbol{x}}\mathcal{S} \to T_gG$. Each Lie algebra basis $L_i \in \mathfrak{g} = T_IG$ generates a vector field $\hat{L}_i$ on the tangent bundle TG via the pushforward $\hat{L}_i(g) = gL_ig^{-1}$. Via the lift, L_i also generates a vector field $\hat{L}_i = \hat{L}_i^\alpha(\boldsymbol{x})\partial_\alpha = [gL_i\boldsymbol{x}_0]^\alpha\partial_\alpha$.

Most existing literature on equivariance implements $\int_G$ by discretizing the group or summing over irreducible representations. We take a different approach and instead use the infinitesimal generators of the group. While a Lie group G is infinite, usually it can be generated using a small number of infinitesimal generator, comprising its "Lie algebra". We use the Lie algebra to introduce a building block from which any G-conv can be constructed. Next, we review core concepts about Lie algebras. Figure 1 shows an illustration of a Lie group, Lie algebra and the concept we discuss below.

Lie algebra Let G be a Lie group, which includes common continuous groups. Group elements $u \in G$ infinitesimally close to the identity element I can be written as $u \approx I + \epsilon^i L_i$ (note Einstein summation), where $L_i \in \mathfrak{g}$ with the Lie algebra $\mathfrak{g} = T_IG$ is the tangent space of G at the identity element. The Lie algebra has the property that it is closed under a Lie bracket $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$

$$[L_i, L_j] = c_{ij}{}^k L_k, \tag{5}$$

which is skew-symmetric and satisfies the Jacobi identity. Here the coefficients $c_{ij}{}^k \in \mathbb{R}$ or $\mathbb{C}$ are called the structure constants of the Lie algebra. For matrix representations of $\mathfrak{g}$, $[L_i, L_j] = L_iL_j - L_jL_i$ is the commutator. The L_i are called the infinitesimal generators of the Lie group.

Exponential map If the manifold of G is connected [1], an exponential map $\exp : \mathfrak{g} \to G$ can be defined such that $g = \exp[t^iL_i] \in G$. For matrix groups, if G is connected and compact, the matrix exponential is such a map and it is surjective. For most other groups (except $\mathrm{GL}_d(\mathbb{C})$ and nilpotent groups) it is not surjective. Nevertheless, for any connected group every $g \in G$ can be written as a product $g = \prod_a \exp[t_a^iL_i]$ (Hall, 2015). Making t_a^i infinitesimal steps $dt^i(s)$ tangent to a path γ from I to g on G yields the surjective path-ordered exponential in physics, denoted as $g = P\exp[\int_\gamma dt^iL_i]$ (SI A, and see Time-ordering in Weinberg (1995, p143)).

Pushforward $L_i \in T_IG$ can be pushed forward to $\hat{L}_i(g) = gL_ig^{-1} \in T_gG$ to form a basis for T_gG, satisfying the same Lie algebra $[\hat{L}_i(g), \hat{L}_j(g)] = c_{ij}{}^k\hat{L}_k(g)$. The manifold of G together with the set of all T_gG attached to each g forms the tangent bundle TG, a type of fiber bundle (Lee et al., 2009). $\hat{L}_i$ is a vector field on TG. The lift maps $\hat{L}_i$ to an equivalent vector field on $T\mathcal{S}$, which we will also denote by $\hat{L}_i$. Figure 1 illustrates the flow of these vector fields on TG and $T\mathcal{S}$.

3 Lie Algebra Convolutional Network

We now use the Lie algebra to expad G-conv near identity of the group. This yields basic building blocks from which an arbitrary G-conv can be constructed. We consider a G-conv equation 3 with a

[1] When G has multiple connected components, these results hold for the component containing I, and generalize easily for mutli-component groups such as $\mathbb{Z}_k \otimes G$ (Finzi et al., 2021).

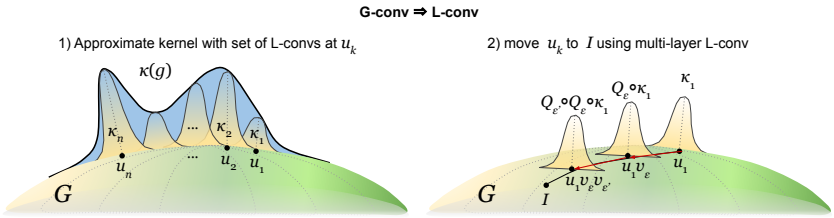

Figure 2: Sketch of the procedure for approximating G-conv using L-conv. First, the kernel is written as the sum of a number of localized kernels κ_k with support around u_k (left). Each of the κ_k is then moved toward identity by composing multiple L-conv layers $Q_{\epsilon'} \circ Q_\epsilon \ldots \kappa_k$ (right).

kernel localized near identity I. Let $\kappa_0(u) = c\delta_\eta(u)$, with constants $c \in \mathbb{R}^{m'} \otimes \mathbb{R}^m$ and a normalized kernel $\delta_\eta(u) \in \mathbb{R}$ with support on an η neighborhood of I, meaning $\delta_\eta(I + \epsilon^i L_i) \to 0$ if $\|\epsilon\|^2 > \eta^2$.

Lie algebra convolutional (L-conv) layer We can expand a G-conv whose kernel is κ_0 in the Lie algebra of G to linear order to obtain the L-conv architecture

$$\begin{aligned} Q[f](g) \equiv c[\delta_\eta \star f](g) &= c\int d\epsilon \delta_\eta(I + \epsilon^i L_i)\left[I + \epsilon^i g L_i \cdot \frac{d}{dg}\right] f(g) + O(\eta^2) \\ &= W^0\left[I + \bar{\epsilon}^i g L_i \cdot \frac{d}{dg}\right] f(g) + O(\eta^2) \\ &= [W^0]_b f^a\left(g\left(I + [\bar{\epsilon}^i]^b_a L_i\right)\right) + O(\eta^2) \end{aligned} \tag{6}$$

where $d\epsilon$ is the integration measure on the Lie algebra induced by the Haar measure dv on G. In a matrix representation of G, we have $gL_i \cdot \frac{df}{dg} = [gL_i]^\beta_\alpha \frac{df}{dg^\beta_\alpha} = \mathrm{Tr}\left[[gL_i]^T \frac{df}{dg}\right]$. Here

$$W^0 = c\int d\epsilon \delta_\eta(I + \epsilon^i L_i) \in \mathbb{R}^{m'} \otimes \mathbb{R}^m, \qquad \bar{\epsilon}^i = \int d\epsilon \delta_\eta(I + \epsilon^i L_i)\epsilon^i \in \mathbb{R}^m \otimes \mathbb{R}^m \tag{7}$$

with $\|\bar{\epsilon}\| < \eta$. Being an expansion of G-conv, L-conv is also equivariant.

Proposition 1 (Equivariance of L-conv)**.** *With assumptions above, L-conv is equivariant to $O(\eta^2)$.*

Proof: From equation 6 it follows that L-conv is equivariant up to $O(\bar{\epsilon}^2) \sim O(\eta^2)$, as for $w \in G$

$$w \cdot Q[f](g) = Q[f](w^{-1}g) = W^0 f\left(w^{-1}g\left(I + \bar{\epsilon}^i L_i\right)\right) + O(\eta^2) = Q[w \cdot f](g). \tag{8}$$

□

3.1 Approximating G-conv using L-conv

L-conv can be used as a basic building block which can be combined to make more G-conv with more general kernels. Figure 2 sketches the argument described here (see SI A.1 for more discussion).

Theorem 1 (G-conv from L-convs)**.** *G-conv equation 3 can be approximated using L-conv layers.*

Proof: The procedure involves two steps, as illustrated in Fig. 2: 1) approximate the kernel using localized kernels as the δ_η in L-conv; 2) move the kernels towards identity using multiple L-conv layers. The following lemma outline the details. □

Lemma 1 (Approximating the kernel)**.** *Any kernel κ can be approximated using a set of kernels with η support to arbitrary accuracy.*

Proof: This simply follows from the universal approximation theorem for neural networks (Hornik et al., 1989; Cybenko, 1989) and the same approximation errors apply. $\delta_\eta(g)$ can be written as the sum of two sigmoid functions $\delta_\eta(g) = \mathrm{sig}(v_\eta g) + \mathrm{sig}(v_\eta^{-1} g)$. Thus, we can approximate any kernel as $\kappa(g) \approx \sum_k \kappa_k(g)$ where the local kernels $\kappa_k(g) = c_k \delta_\eta(u_k^{-1} g)$ have support only on an η neighborhood of $u_k \in G$. Here $c_k \in \mathbb{R}^{m'} \otimes \mathbb{R}^m$ are constants and $\delta_\eta(u)$ is as in equation 6. Using this, G-conv equation 3 becomes

$$[\kappa \star f](g) = \sum_k c_k \int dv \delta_\eta(u_k^{-1} v) f(gv) = \sum_k c_k [\delta_\eta \star f](g u_k). \tag{9}$$

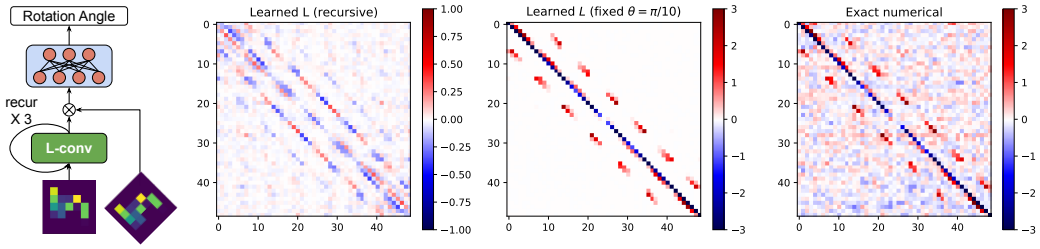

Figure 3: **Learning the infinitesimal generator of** $SO(2)$ Left shows the architecture for learning rotation angles between pairs of images (SI C.3). Next to it is the L learned using recursive L-conv in this experiment. Middle L is learned using a fixed small rotation angle $\theta = \pi/10$, and right shows L found using the numeric solution from the data.

The kernels κ_k are localized around u_k, whereas in L-conv the kernel is around identity. We can compose L-conv layers to move κ_k from u_k to identity. □

Lemma 2 (Moving kernels to identity)**.** *The local kernel κ_k can be moved near identity.*

Proof: The local kernels $\kappa_k(g) = \delta_\eta(gu_k)$ are nonzero around u_k. An L-conv with $W^0 = I$ performs $G[\kappa_k \star f](g) = \kappa_k \star f(gv_\epsilon)$, which moves the κ_k by $v_\epsilon = (I + \bar{\epsilon}^i L_i)$ on G. Thus, by applying multiple L-conv layers we can find $Q_p \circ \ldots Q_1 \circ \kappa_k \approx c_k \delta_\eta$. This can always be done using a set of $v_{\epsilon_a} = (I + \bar{\epsilon}^i_a L_i)$ such that $u_k \approx \prod_{a=1}^{p} v_{\epsilon_a}$. With $\|\epsilon_a\| < \eta$, the error in u_k is $O(\eta^{p+1})$. □

Thus, we conclude that any G-conv equation 3 can be approximated by multilayer L-conv. We conducted small controlled experiments to verify this (SI C). We briefly discuss them here.

Learning symmetries using L-conv Rao & Ruderman (1999) introduced a basic version of L-conv and showed that it can learn 1D translation and 2D rotation. The learned L_i for 1D translation reproduced finite translation well using $(I + \epsilon L)^N$, which is N recursive L-conv layers. Hence, their results prove that L-conv can be used to approximate CNN, as well as $SO(2)$ G-conv. We discuss the details of L-conv approximating CNN below in sec. 5. We also conducted more complex experiments using recursive L-conv to learn large rotation angle between two images (SI C). Figure 3 shows the learned L. Left shows $L \in so(2)$ learned using L-conv in 3 recursive layers to learn rotation angles between a pair of 7×7 random images $\boldsymbol{f}$ and $R(\theta)\boldsymbol{f}$ with $\theta \in [0, \pi/3)$. Middle and right of Fig. 3 are experiments with fixed small rotation angle $\theta = \pi/10$ (SI C.2). Middle is the L learned using L-conv and right is using the exact solution $R = (YX^T)(X^TX)^{-1}$. While the middle L is less noisy, it does not capture weights beyond first neighbors of each pixel. (also see SI C for a discussion on symmetry discovery literature.)

L-conv can potentially replace other equivariant layers in an architectures. We conducted limited experiments for this on small image datasets (SI D). L-conv allows one to look for potential symmetries in data which may have been scrambled or harbors hidden symmetries. Next, we will provide some examples of explicit forms of L-conv.

3.2 Interpretation and Examples

The $gL_i \cdot df/dg$ in equation 6 can be written in terms of partial derivatives $\partial_\alpha f(\boldsymbol{x}) = \partial f/\partial \boldsymbol{x}^\alpha$. In general, using $\boldsymbol{x}^\rho = g^\rho_\sigma \boldsymbol{x}^\sigma_0$, we have $\frac{df(g\boldsymbol{x}_0)}{dg^\alpha_\beta} = \boldsymbol{x}^\beta_0 \partial_\alpha f(\boldsymbol{x})$, and so

$$gL_i \cdot \frac{df}{dg} = [gL_i]^\alpha_\beta \boldsymbol{x}^\beta_0 \partial_\alpha f(\boldsymbol{x}) = [gL_i \boldsymbol{x}_0] \cdot \nabla f(\boldsymbol{x}) = \hat{L}_i f(\boldsymbol{x}) \tag{10}$$

Hence, for each L_i, the pushforward gL_i generates a flow on $\mathcal{S}$ through the vector field $\hat{L}_i \equiv gL_i \cdot d/dg = [gL_i \boldsymbol{x}_0]^\alpha \partial_\alpha$ (Fig. 1). Being a vector field $\hat{L}_i \in T\mathcal{S}$ (i.e. 1-tensor), $\hat{L}_i$ is basis independent, meaning for $v \in G$, $\hat{L}_i(v\boldsymbol{x}) = \hat{L}_i$. Its components transform as $[\hat{L}_i(v\boldsymbol{x})]^\alpha = [vgL_i\boldsymbol{x}_0]^\alpha = v^\alpha_\beta \hat{L}_i(\boldsymbol{x})^\beta$, while the partial transforms as $\partial/\partial[v\boldsymbol{x}]^\alpha = [v^{-1}]^\gamma_\alpha \partial_\gamma$.

Explicit examples Using equation 10 we can calculate the form of L-conv for specific groups (details in SI A.2). For **translations** $G = T_n = (\mathbb{R}^n, +)$ we find the generators become simple partial derivatives $\hat{L}_i = \partial_i$ (SI A.2.2), yielding $f(\boldsymbol{x}) + \epsilon^\alpha \partial_\alpha f(\boldsymbol{x})$. For **2D rotations** (SI A.2.1) we find $\hat{L} \equiv (x\partial_y - y\partial_x) = \partial_\theta$, which in physics is called the angular momentum operator about the z-axis in quantum mechanics and field theories. For rotatons with scaling, **Rotation and scaling** $G = SO(2) \times \mathbb{R}^+$, we have two L_i, one $\hat{L}_\theta = \partial_\theta$ from $so(2)$ and a scaling with $L_r = I$, yielding $\hat{L}_r = x\partial_x + y\partial_y = r\partial_r$. Next, we discuss the form of L-conv on discrete data.

4 Tensor notation

In many datasets, such as images, $f(\boldsymbol{x})$ is not given as continuous function, but rather as a discrete array, with $\mathcal{S} = \{\boldsymbol{x}_0, \dots \boldsymbol{x}_{d-1}\}$ containing d points. Each $\boldsymbol{x}_\mu$ represents a coordinate in higher dimensional space, e.g. on a 10×10 image, $\boldsymbol{x}_0$ is $(x, y) = (0, 0)$ point and $\boldsymbol{x}_{99}$ is $(x, y) = (9, 9)$.

Feature maps and group action In the tensor notation, we encode $\boldsymbol{x}_\mu \in \mathcal{S}$ as the canonical basis (one-hot) vectors in $\boldsymbol{x}_\mu \in \mathbb{R}^d$ with $[\boldsymbol{x}_\mu]_\nu = \delta_{\mu\nu}$ (Kronecker delta), e.g. $\boldsymbol{x}_0 = (1, 0, \dots, 0)$. The features become $\boldsymbol{f} \in \mathcal{F} = \mathbb{R}^d \otimes \mathbb{R}^m$, meaning $d \times m$ tensors, with $f(\boldsymbol{x}_\mu) = \boldsymbol{x}_\mu^T \boldsymbol{f} = \boldsymbol{f}_\mu$. Although $\mathcal{S}$ is discrete, the group acting on $\mathcal{F}$ can be continuous (e.g. image rotations). Any $G \subseteq \mathrm{GL}_d(\mathbb{R})$ of the general linear group (invertible $d \times d$ matrices) acts on $\boldsymbol{x}_\mu \in \mathbb{R}^d$ and $\boldsymbol{f} \in \mathcal{F}$. We define $f(g \cdot \boldsymbol{x}_\mu) = \boldsymbol{x}_\mu^T g^T \boldsymbol{f}, \forall g \in G$, so that for $w \in G$ we have

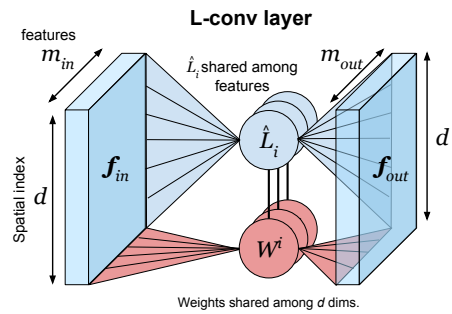

Figure 4: L-conv layer architecture. L_i only act on the d flattened spatial dimensions, and W^i only act on the m_{in} input features and returns m_{out} output features. For each i, L-conv is analogous to a Graph Convolutional Network with d nodes and m_{in} features.

$$w \cdot f(\boldsymbol{x}_\mu) = f(w^{-1} \cdot \boldsymbol{x}_\mu) = \boldsymbol{x}_\mu^T w^{-1T} \boldsymbol{f} = [w^{-1}\boldsymbol{x}_\mu]^T \boldsymbol{f} \tag{11}$$

Dropping the position $\boldsymbol{x}_\mu$, the transformed features are matrix product $w \cdot \boldsymbol{f} = w^{-1T}\boldsymbol{f}$.

G-conv and L-conv in tensor notation Writing G-conv equation 3 in the tensor notation we have

$$[\kappa \star f](g\boldsymbol{x}_0) = \int_G \kappa(v) f(gv\boldsymbol{x}_0) dv = \boldsymbol{x}_0^T \int_G v^T g^T \boldsymbol{f} \kappa^T(v) dv \equiv \boldsymbol{x}_0^T [\boldsymbol{f} \star \kappa](g) \tag{12}$$

where we moved $\kappa^T(v) \in \mathbb{R}^m \otimes \mathbb{R}^{m'}$ to the right of $\boldsymbol{f}$ because it acts as a matrix on the output index of $\boldsymbol{f}$. The equivariance of equation 12 is readily checked with $w \in G$

$$w \cdot [\boldsymbol{f} \star \kappa](g) = [\boldsymbol{f} \star \kappa](w^{-1}g) = \boldsymbol{x}_0^T \int_G v^T g^T w^{-1T} \boldsymbol{f} \kappa^T(v) dv = [(w \cdot \boldsymbol{f}) \star \kappa](g) \tag{13}$$

where we used $[w^{-1}g]^T \boldsymbol{f} = g^T w^{-1T} \boldsymbol{f}$. Similarly, we can rewrite L-conv equation 6 in the tensor notation. Defining $v_\epsilon = I + \overline{\epsilon}^i L_i$

$$\begin{aligned} Q[\boldsymbol{f}](g) &= W^0 f\left(g\left(I + \overline{\epsilon}^i L_i\right)\right) = \boldsymbol{x}_0^T \left(I + \overline{\epsilon}^i L_i\right)^T g^T \boldsymbol{f} W^{0T} \\ &= \left(\boldsymbol{x}_0 + \overline{\epsilon}^i [g L_i \boldsymbol{x}_0]\right)^T \boldsymbol{f} W^{0T}. \end{aligned} \tag{14}$$

Here, $\hat{L}_i = gL_i\boldsymbol{x}_0$ is eactly the tensor analogue of pushforward vector field $\hat{L}_i$ in equation 10. We will make this analogy more precise below. The equivariance of L-conv in tensor notation is again evident from the $g^T\boldsymbol{f}$, resulting in

$$Q[w \cdot \boldsymbol{f}](g) = \boldsymbol{x}_0^T v_\epsilon^T g^T w^{-1T} \boldsymbol{f} W^{0T} = Q[\boldsymbol{f}](w^{-1}g) = w \cdot Q[\boldsymbol{f}](g) \tag{15}$$

Lie Algebra Convolutional Layer In conclusion, the discrete space L-conv equation 58 can be rewritten using the global Lie algebra basis $\hat{L}_i$

$$Q[\boldsymbol{f}] = \left(\boldsymbol{f} + \hat{L}_i \boldsymbol{f} \overline{\epsilon}^i\right) W^{0T}, \qquad Q[\boldsymbol{f}]^a_\mu = \boldsymbol{f}^b_\mu [W^{0T}]^a_b + [\hat{L}_i]^\nu_\mu \boldsymbol{f}^c_\nu \left[W^i\right]^a_c \tag{16}$$

Where $W^i = W^0 \overline{\epsilon}^i$, $W^0 \in \mathbb{R}^{m_{in}} \otimes \mathbb{R}^{m_{out}}$ and $\overline{\epsilon}^i \in \mathbb{R}^{m_{in}} \otimes \mathbb{R}^{m_{in}}$ are trainable weights. The $\hat{L}_i$ can be either inserted as inductive bias or they can be learned to discover symmetries.

Discrete vector field and tangent bundle We can show (SI B.1) that when $\mathcal{S}$ has a topology $\hat{L}_i = gL_i\boldsymbol{x}_0$ is indeed the discrete version of the $\hat{L}_i$ in equation 10. The topology, to lowest order, using a graph (1-simplex) is encoded as an incidence matrix $\boldsymbol{B} : \mathcal{S} \times \mathcal{E} \to \{0, 1, -1\}$ with vertices $\mathcal{S}$ and edge set $\mathcal{E}$. $\boldsymbol{B}^{\mu}_{\alpha} = 1$ or -1 if edge α starts or ends at node μ, respectively, and $\boldsymbol{B}^{\mu}_{\alpha} = 0$ otherwise. $\boldsymbol{B}$ is used to extend differential geometry to graphs (Schaub et al., 2020), since for α an outgoing edge from vertex μ, we have $\boldsymbol{B}_{\alpha}\boldsymbol{f} = \boldsymbol{f}_{\mu} - \boldsymbol{f}_{\nu} \sim \partial_{\alpha}\boldsymbol{f}$. We show that for G to preserve the topology, components of $\hat{L}_i$ should act on the local neighborhood of each node and related to each other via the pushforward

$$[\hat{L}_i]^{\nu}_{\mu} = [g_{\mu}L_i\boldsymbol{x}_0]^{\nu} = [\hat{\ell}_i]^{\alpha}_{\mu}\boldsymbol{B}^{\nu}_{\alpha} = [\hat{\ell}_i]^{\alpha}_{0}\boldsymbol{B}^{\rho}_{\alpha}[g_{\mu}]^{\nu}_{\rho}. \tag{17}$$

$\hat{L}_i \equiv \hat{\ell}^{\alpha}_i\boldsymbol{B}_{\alpha}$ is the discrete $\mathcal{S}$ version of the vector field $\hat{L}_i(\boldsymbol{x}) = [gL_i\boldsymbol{x}_0]^{\alpha}\partial_{\alpha}$ in equation 10. Here $[\hat{\ell}_i]^{\alpha}_0 \in \mathbb{R}$ are weightings for edges α connected to $\boldsymbol{x}_0$ and capture the action of L_i. This weighting is similar to Gauge Equivariant Mesh (GEM) CNN (Cohen et al., 2019a). The main difference is that in GEM-CNN the basis of the tangent spaces $T_{\boldsymbol{x}_{\mu}}\mathcal{S}$ are not fixed and are chosen by a gauge. In L-conv the lift $\boldsymbol{x}_{\mu} = g_{\mu}\boldsymbol{x}_0$ fixes the gauge by mapping neighbors of $\boldsymbol{x}_0$ to neighbors of $\boldsymbol{x}_{\mu}$. Changing how the discrete $\mathcal{S}$ samples an underlying continuous space changes g_{μ} and hence the gauge, similar to GEM-CNN. Next, we discuss the relation between L-conv and other neural architectures.

5 Relation to other architectures

L-conv can reproduce some familiar architectures for specific Lie groups, as we discuss now.

CNN This is a special case of expressing G-conv as L-conv when the group is continuous 1D translations. The arguments here generalize trivially to higher dimensions. Rao & Ruderman (1999, sec. 4) used the Shannon-Whittaker Interpolation (Whitaker, 1915) to define continuous translation on periodic 1D arrays as $\boldsymbol{f}'_{\rho} = g(z)^{\nu}_{\rho}\boldsymbol{f}_{\nu}$. Here $g(z)^{\nu}_{\rho} = \frac{1}{d}\sum_{p=-d/2}^{d/2}\cos\left(\frac{2\pi p}{d}(z+\rho-\nu)\right)$ approximates the shift operator for continuous z. These $g(z)$ form a 1D translation group G as $g(w)g(z) = g(w+z)$ with $g(0)^{\nu}_{\rho} = \delta^{\nu}_{\rho}$. For any $z = \mu \in \mathbb{Z}$, $g_{\mu} = g(z = \mu)$ are circulant matrices that shift by μ as $[g_{\mu}]^{\rho}_{\nu} = \delta^{\rho}_{\nu-\mu}$. Thus, a 1D CNN with kernel size k can be written suing g_{μ} as

$$F(\boldsymbol{f})^{a}_{\nu} = \sigma\left(\sum_{\mu=0}^{k}\boldsymbol{f}^{c}_{\nu-\mu}[W^{\mu}]^{a}_{c} + b^{a}\right) = \sigma\left(\sum_{\mu=0}^{k}[g_{\mu}\boldsymbol{f}]^{c}_{\nu}[W^{\mu}]^{a}_{c} + b^{a}\right) \tag{18}$$

where W, b are the filter weights and biases. g_{μ} can be approximated using the Lie algebra and written as multi-layer L-conv as in sec. 3.1. Using $g(0)^{\nu}_{\rho} \approx \delta(\rho - \nu)$, the single Lie algebra basis $[\hat{L}]_0 = \partial_z g(z)|_{z\to 0}$, acts as $\hat{L}f(z) \approx -\partial_z f(z)$ (because $\int \partial_z\delta(z-\nu)f(z) = -\partial_{\nu}f(\nu)$). Its components are $\hat{L}^{\nu}_{\rho} = L(\rho-\nu) = \sum_p \frac{2\pi p}{d^2}\sin\left(\frac{2\pi p}{d}(\rho-\nu)\right)$, which are also circulant due to the $(\rho-\nu)$ dependence. Hence, $[\hat{L}\boldsymbol{f}]_{\rho} = \sum_{\nu} L(\rho-\nu)\boldsymbol{f}_{\nu} = [L\star\boldsymbol{f}]_{\nu}$ is a convolution. Rao & Ruderman (1999) already showed that this $\hat{L}$ can reproduce finite discrete shifts g_{μ} used in CNN. They used a primitive version of L-conv with $g_{\mu} = (I + \epsilon\hat{L})^N$. Thus, L-conv can approximate 1D CNN. This result generalizes easily to higehr dimensions.

Relation with Graph Convolutional Networks (GCN) Let $\boldsymbol{A}$ be the adjacency matrix of a graph. In equation 16 if $\hat{L}_i = h(\boldsymbol{A})$, such as $\hat{L}_i = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, we obtain a GCN (Kipf & Welling, 2016) ($\boldsymbol{D}_{\mu\nu} = \delta_{\mu\nu}\sum_{\rho}\boldsymbol{A}_{\mu\rho}$ being the degree matrix). So in the special case where all neighbors of each node $<\mu>$ have the same edge weight, meaning $[\hat{L}_i]^{\nu}_{\mu} = [\hat{L}_i]^{\rho}_{\mu}$, $\forall\nu,\rho \in <\mu>$, equation 6 is uniformly aggregating over neighbors and L-conv reduces to a GCN. Note that this similarity is not just superficial. In GCN $h(\boldsymbol{A}) = \hat{L}$ is in fact a Lie algebra basis. When $\hat{L} = h(\boldsymbol{A})$, the vector field is the flow of isotropic diffusion $d\boldsymbol{f}/dt = h(\boldsymbol{A})\boldsymbol{f}$ from each node to its neighbors. This vector field defines one parameter Lie group with elements $g(t) = \exp[h(\boldsymbol{A})t]$. Hence, L-conv for flow groups with a single generator are GCN. These flow groups include Hamiltonian flows and other linear dynamical systems.

The main difference between L-conv and GCN is that L-conv can assign a different weight to each neighbor of the same node. This weighting is similar to Gauge Equivariant Mesh (GEM) CNN

(Cohen et al., 2019a). The main difference is that in GEM-CNN the basis of the tangent spaces $< \mu >= T_{\boldsymbol{x}_\mu}\mathcal{S}$ are not fixed and are chosen by a gauge. In L-conv the lift $\boldsymbol{x}_\mu = g_\mu \boldsymbol{x}_0$ fixes the gauge, meaning it determines how the basis at $< 0 >$ maps to the basis at $< \mu >$. Changing how the discrete $(\mathcal{S}, \boldsymbol{A})$ samples the underlying topological space $\mathcal{S}_0$ changes the lift g_μ and hence the gauge, similar to GEM-CNN. Next, we discuss the mathematical properties of loss functions for L-conv.

6 Group invariant loss

Loss functions of equivariant architectures are rarely discussed. Yet, recent work by Kunin et al. (2020) showed existence of symmetry directions in the loss landscape. To understand how the symmetry generators in L-conv manifest themselves in the loss landscape, we work out the explicit example of a mean square loss (MSE). Because G is the symmetry group, f and $g \cdot f$ should result in the same optimal parameters. Hence, the minima of the loss function need to be *group invariant*. One way to satisfy this is for the loss itself to be group invariant, which can be constructed by integrating over G (global pooling (Bronstein et al., 2021)). A function $I = \int_G dg F(g)$ is G-invariant (SI A.3). We can also change the integration to $\int_{\mathcal{S}} d^n x$ by change of variable $dg/d\boldsymbol{x}$ (see SI A.3 for discussion on stabilizers).

MSE loss The MSE is given by $I = \sum_n \int_G dg \|Q[f_n](g)\|^2$, where f_n are data samples and $Q[f]$ is L-conv or another G-equivariant function. In supervised learning the input is a pair f_n, y_n. G can also act on the labels y_n. We assme that y_n are either also scalar features $y_n : \mathcal{S} \to \mathbb{R}^{m_y}$ with a group action $g \cdot y_n(\boldsymbol{x}) = y_n(g^{-1}\boldsymbol{x})$ (e.g. f_n and y_n are both images), or that y_n are categorical. In the latter case $g \cdot y_n = y_n$ because the only representations of a continuous G on a discrete set are constant. We can concatenate the inputs to $\phi_n \equiv [f_n|y_n]$ with a well-defined G action $g \cdot \phi_n = [g \cdot f_n | g \cdot y_n]$. The collection of combined inputs $\Phi = (\phi_1, \ldots, \phi_N)^T$ is an $(m + m_y) \times N$ matrix. Using equations 6 and 10, the MSE loss with parameters $W = \{W^0, \overline{\epsilon}\}$ becomes (SI A.3.1)

$$
\begin{aligned}
I[\Phi; W] &= \int_G dg \mathcal{L}[\Phi; W] = \int_G dg \left\| W^0 \left[I + \overline{\epsilon}^i [\hat{L}_i]^\alpha \partial_\alpha \right] \Phi(g) \right\|^2 \\
&= \int_{\mathcal{S}} \frac{d^n x}{\left| \frac{\partial x}{\partial g} \right|} \left[\Phi^T \mathbf{m}_2 \Phi + \partial_\alpha \Phi^T \mathbf{h}^{\alpha\beta} \partial_\beta \Phi + [\hat{L}_i]^\alpha \partial_\alpha \left(\Phi^T \mathbf{v}^i \Phi \right) \right]
\end{aligned} \tag{19}
$$

where $\left| \frac{\partial x}{\partial g} \right|$ is the determinant of the Jacobian, $W^i = W^0 \overline{\epsilon}^i$ and

$$
\mathbf{m}_2 = W^{0T} W^0, \qquad \mathbf{h}^{\alpha\beta}(\boldsymbol{x}) = \overline{\epsilon}^{iT} \mathbf{m}_2 \overline{\epsilon}^j [\hat{L}_i]^\alpha [\hat{L}_j]^\beta, \qquad \mathbf{v}^i = \mathbf{m}_2 \overline{\epsilon}^i. \tag{20}
$$

Note that $\mathbf{h}$ has feature space indices via $[\overline{\epsilon}^{iT} \mathbf{m}_2 \overline{\epsilon}^j]_{ab}$, with index symmetry $\mathbf{h}^{\alpha\beta}_{ab} = \mathbf{h}^{\beta\alpha}_{ba}$. When $\mathcal{F} = \mathbb{R}$ (i.e. f is a 1D scalar), $\mathbf{h}^{\alpha\beta}$ becomes a a Riemannian metric for $\mathcal{S}$. In general $\mathbf{h}$ combines a 2-tensor $\mathbf{h}_{ab} = \mathbf{h}^{\alpha\beta}_{ab} \partial_\alpha \partial_\beta \in T\mathcal{S} \otimes T\mathcal{S}$ with an inner product $h^T \mathbf{h}^{\alpha\beta} f$ on the feature space $\mathcal{F}$. Hence $\mathbf{h} \in T\mathcal{S} \otimes T\mathcal{S} \otimes \mathcal{F}^* \otimes \mathcal{F}^*$ is a $(2,2)$-tensor, with $\mathcal{F}^*$ being the dual space of $\mathcal{F}$.

A simpler version of the loss function in equation 19 is commonly used in field theories in physics (Polyakov, 2018). There too, the motivation is to preserve spatial symmetries for the metric $\mathbf{h}$. In equation 19, $\mathbf{h}$ transforms equivariantly as a 2-tensor $v \cdot \mathbf{h}^{\alpha\beta} = [v^{-1}]^\alpha_\rho [v^{-1}]^\beta_\gamma \mathbf{h}^{\rho\gamma}(\boldsymbol{x})$ for $v \in G$ (SI A.3). In physics the last term in equation 19 is absent. We can show that for L-conv also when G is translations T_n, or rotations $SO(n)$, by partial integration the last term becomes a boundary term and expected to vanish (SI A.3). We discuss the implications of this overlap between physics and equivariant neural networks further below. One aspect of the physics theory which may benefit machine learning is that for ϕ_n which minimize the loss, the loss function is robust to small perturbations. This yields a natural way to probe generalization in equivariant neural networks, discussed next.

Generalization error Equivariant neural networks are expected to be more robust than others, meaning equivariance should improve generalization. One way to check this is to see how the network would perform for an input $\phi' = \phi + \delta\phi$ which adds a small perturbation $\delta\phi$ to a real data point ϕ. Note $\delta\phi$ here s a random noise, not an adversarial attack. Robustness to such perturbation would mean that, for optimal parameters W^* , the loss function would not change, i.e. $I[\phi'; W^*] = I[\phi; W^*]$.

This can be cast as a variational equation, requiring I to be minimized around real data points ϕ. Writing $I[\phi; W] = \int d^n x \mathcal{L}[\phi; W]$, we have

$$\delta I[\phi; W^*] = \int_{\mathcal{S}} d^n x \left[\frac{\partial \mathcal{L}}{\partial \phi^a} \delta \phi^a + \frac{\partial \mathcal{L}}{\partial (\partial_\alpha \phi^a)} \partial_\alpha (\delta \phi^a) \right] \tag{21}$$

Doing a partial integration on the second term, we get a vanishing boundary term plus the familiar Euler-Lagrange (EL) equation (SI A.4), widely used in physics and control theory. Thus, requiring good generalization, i.e. $\delta I[\phi; W^*]/\delta\phi = 0$ means for optimal parameters W^*, real data points ϕ satisfy the EL equations

$$\text{Generalization Error Minimization} \iff \text{EL:} \quad \frac{\partial \mathcal{L}}{\partial \phi^b} - \partial_\alpha \frac{\partial \mathcal{L}}{\partial (\partial_\alpha \phi^b)} = 0 \tag{22}$$

Another idea which can be transferred from physics to equivariant neural networks is that symmetries produce conserved currents, via Noether's theorem, also discussed in Kunin et al. (2020). We briefly discuss motivate this below.

Conservation laws The idea is that the equivariance condition equation 2 can be written for the integrand of the loss $\mathcal{L}[\phi, W]$. If we write the equivariance equation for infinitesimal v_ϵ, we obtain a vector field which is divergence free. Since G is the symmetry of the system, transforming an input $\phi \to w \cdot \phi$ by $w \in G$ the integrand should change equivariantly, meaning $\mathcal{L}[w \cdot \phi] = w \cdot \mathcal{L}[\phi]$. When generalization error is minimized as in equation 22, an infinitesimal $w \approx I + \eta^i L_i$, with $\delta\phi = \epsilon^i \hat{L}_i \phi$, results in a conserved current (SI A.4)

$$\text{Noether current: } J^\alpha = \frac{\partial \mathcal{L}}{\partial (\partial_\alpha \phi^b)} \delta \phi^b - \frac{\partial \mathcal{L}}{\partial \boldsymbol{x}^\alpha} \delta \boldsymbol{x}^\alpha, \qquad \delta I[\phi; W^*] = 0 \quad \Rightarrow \partial_\alpha J^\alpha = 0 \tag{23}$$

This J is called the Noether current in physics. It captures the flow of the Lagrangian $\mathcal{L}$ along symmetry direction $\hat{L}_i$. It would be interesting to see if these conserved currents can be used in practice as an alternative way for identifying or discovering symmetries. We conclude by remarking on similarity between ML and physics problems.

Relation to physics Although equation 19 was derived purely using ML and equivariance, physicists will recognize equation 19 as a generalization of the non-interacting field theory action (integral of a Lagrangian). In field theory, $\phi(x)$ (each input feature $[f|y]$) is called a field. $\mathbf{m}_2$ is the "mass matrix" whose eigenvalues are the squares of masses of the fields. The second term $\partial\phi^T \cdot \mathbf{h} \cdot \partial\phi = \partial^\mu \phi^a \partial_\mu \phi_a$ is the "kinetic" term. In physics, the loss in equation 19 is used in two ways. When the parameters $\mathbf{m}_2$ and $\mathbf{h}$ are known, a variational procedure known as the Euler-Lagrange equations is used to find optimal $\phi(\boldsymbol{x}) = \arg\min_\phi I[\phi]$. When $\mathbf{m}_2$ and $\mathbf{h}$ are not known, ϕ_n are observational data and physicists use equation 19 the same way as ML. In this case equation 19 is a variational ansatz used to identify parameters $\mathbf{m}_2$ and $\mathbf{h}$ such that the expected loss for the observed ϕ_n is minimized. The main difference with physics is that physics assumes a simpler form in which $\mathbf{h}$ factorizes into a tensor product $\mathbf{h} = \eta \otimes \delta$, where $\eta_{\alpha\beta}$ is a Riemannian metric on $T\mathcal{S}$ and δ, usually the Kronecker delta, is the metric on $\mathcal{F}$. We conclude with some closing remarks.

7 Discussions

In conclusion, we found that expanding group convolutional architectures into a smaller building block, L-conv, has a number of benefits. First, L-conv has a much simpler and more familiar structure than G-conv. We showed that L-conv layers can be combined to approximate any G-conv, without the need to encode many irreps or discretizing the group. Additionally, L-conv's simple structure allows us to discover symmetries from data. L-conv is easy to implement and can be implemented like a GCN. But perhaps the most interesting discovery here is the connection made with symmetry encoding models in physics. This is significant for two reasns. One is that methods for dealing with symmetry used in physics acquire important meanings in ML, as our discussion on the loss function, generalization and conservation laws show. The second reason is for scientific machine learning. Physicists usually use the simplest forms for the loss function, similar to the MSE loss above. Many areas of physics suffer from this because these ansatze diverge in many cases or fit the data poorly. Hence, more complex L-conv loss functions can potentially serve as more advanced ansatze for solving physics problems.

References

Anselmi, F., Evangelopoulos, G., Rosasco, L., and Poggio, T. Symmetry-adapted representation learning. *Pattern Recognition*, 86:201–208, 2019.

Belavin, A. A., Polyakov, A. M., and Zamolodchikov, A. B. Infinite conformal symmetry in two-dimensional quantum field theory. *Nuclear Physics B*, 241(2):333–380, 1984.

Benton, G., Finzi, M., Izmailov, P., and Wilson, A. G. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

Bogatskiy, A., Anderson, B., Offermann, J., Roussi, M., Miller, D., and Kondor, R. Lorentz group equivariant neural network for particle physics. In *International Conference on Machine Learning*, pp. 992–1002. PMLR, 2020.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Cohen, T. and Welling, M. Learning the irreducible representations of commutative lie groups. In *International Conference on Machine Learning*, pp. 1755–1763, 2014.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999, 2016a.

Cohen, T., Weiler, M., Kicanaoglu, B., and Welling, M. Gauge equivariant convolutional networks and the icosahedral cnn. In *International Conference on Machine Learning*, pp. 1321–1330. PMLR, 2019a.

Cohen, T. S. and Welling, M. Steerable cnns. 2016b.

Cohen, T. S., Geiger, M., Köhler, J., and Welling, M. Spherical cnns. In *International Conference on Learning Representations*, 2018.

Cohen, T. S., Geiger, M., and Weiler, M. A general theory of equivariant cnns on homogeneous spaces. In *Advances in Neural Information Processing Systems*, pp. 9142–9153, 2019b.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Finzi, M., Stanton, S., Izmailov, P., and Wilson, A. G. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. *arXiv preprint arXiv:2002.12880*, 2020.

Finzi, M., Welling, M., and Wilson, A. G. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv preprint arXiv:2104.09459*, 2021.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International Conference on Machine Learning*, pp. 1263–1272. PMLR, 2017.

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

Hartford, J., Graham, D., Leyton-Brown, K., and Ravanbakhsh, S. Deep models of interactions across sets. In *International Conference on Machine Learning*, pp. 1909–1918. PMLR, 2018.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

Kondor, R., Lin, Z., and Trivedi, S. Clebsch–gordan nets: a fully fourier space spherical convolutional neural network. In *Advances in Neural Information Processing Systems*, pp. 10117–10126, 2018.

Kunin, D., Sagastuy-Brena, J., Ganguli, S., Yamins, D. L., and Tanaka, H. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *arXiv preprint arXiv:2012.04728*, 2020.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, J. M., Chow, B., Chu, S.-C., Glickenstein, D., Guenther, C., Isenberg, J., Ivey, T., Knopf, D., Lu, P., Luo, F., et al. Manifolds and differential geometry. *Topology*, 643:658, 2009.

Maron, H., Litany, O., Chechik, G., and Fetaya, E. On learning sets of symmetric elements. In *International Conference on Machine Learning*, pp. 6734–6744. PMLR, 2020.

Polyakov, A. M. *Gauge fields and strings*. Routledge, 2018.

Rao, R. P. and Ruderman, D. L. Learning lie groups for invariant visual perception. In *Advances in neural information processing systems*, pp. 810–816, 1999.

Ravanbakhsh, S. Universal equivariant multilayer perceptrons. *arXiv preprint arXiv:2002.02912*, 2020.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2892–2901. JMLR. org, 2017.

Schaub, M. T., Benson, A. R., Horn, P., Lippner, G., and Jadbabaie, A. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, 2020.

Sohl-Dickstein, J., Wang, C. M., and Olshausen, B. A. An unsupervised algorithm for learning lie group transformations. *arXiv preprint arXiv:1001.1027*, 2010.

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

Weiler, M., Geiger, M., Welling, M., Boomsma, W., and Cohen, T. S. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, pp. 10381–10392, 2018a.

Weiler, M., Hamprecht, F. A., and Storath, M. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 849–858, 2018b.

Weinberg, S. *The quantum theory of fields*, volume 3. Cambridge university press, 1995.

Whitaker, E. On the functions which are represented by the expansion of interpolating theory. In *Proc. Roy. Soc. Edinburgh*, volume 35, pp. 181–194, 1915.

Worrall, D. E., Garbin, S. J., Turmukhambetov, D., and Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Zhou, A., Knowles, T., and Finn, C. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]